



IntechOpen

Data Service Outsourcing and Privacy Protection in Mobile Internet

*Authored by Zhen Qin, Erqiang Zhou, Yi Ding,
Yang Zhao, Fuhu Deng and Hu Xiong*



Data Service Outsourcing and Privacy Protection in Mobile Internet

*Authored by Zhen Qin, Erqiang Zhou,
Yi Ding, Yang Zhao, Fuhu Deng and Hu Xiong*

Published in London, United Kingdom



IntechOpen





Supporting open minds since 2005



Data Service Outsourcing and Privacy Protection in Mobile Internet

<http://dx.doi.org/10.5772/intechopen.79901>

Authored by Zhen Qin, Erqiang Zhou, Yi Ding, Yang Zhao, Fuhu Deng and Hu Xiong

Contributors

Zhen Qin, Erqiang Zhou, Yi Ding, Yang Zhao, Fuhu Deng and Hu Xiong

© The Editor(s) and the Author(s) 2018

The rights of the editor(s) and the author(s) have been asserted in accordance with the Copyright, Designs and Patents Act 1988. All rights to the book as a whole are reserved by INTECHOPEN LIMITED. The book as a whole (compilation) cannot be reproduced, distributed or used for commercial or non-commercial purposes without INTECHOPEN LIMITED's written permission. Enquiries concerning the use of the book should be directed to INTECHOPEN LIMITED rights and permissions department (permissions@intechopen.com).

Violations are liable to prosecution under the governing Copyright Law.



Individual chapters of this publication are distributed under the terms of the Creative Commons Attribution - NonCommercial 4.0 International which permits use, distribution and reproduction of the individual chapters for non-commercial purposes, provided the original author(s) and source publication are appropriately acknowledged. More details and guidelines concerning content reuse and adaptation can be found at <http://www.intechopen.com/copyright-policy.html>.

Notice

Statements and opinions expressed in the chapters are those of the individual contributors and not necessarily those of the editors or publisher. No responsibility is accepted for the accuracy of information contained in the published chapters. The publisher assumes no responsibility for any damage or injury to persons or property arising out of the use of any materials, instructions, methods or ideas contained in the book.

First published in London, United Kingdom, 2018 by IntechOpen

eBook (PDF) Published by IntechOpen, 2019

IntechOpen is the global imprint of INTECHOPEN LIMITED, registered in England and Wales, registration number: 11086078, The Shard, 25th floor, 32 London Bridge Street

London, SE19SG - United Kingdom

Printed in Croatia

British Library Cataloguing-in-Publication Data

A catalogue record for this book is available from the British Library

Additional hard and PDF copies can be obtained from orders@intechopen.com

Data Service Outsourcing and Privacy Protection in Mobile Internet

Authored by Zhen Qin, Erqiang Zhou, Yi Ding, Yang Zhao, Fuhu Deng and Hu Xiong
p. cm.

Print ISBN 978-1-78984-335-4

Online ISBN 978-1-78984-336-1

eBook (PDF) ISBN 978-1-83881-840-1

We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

3,800+

Open access books available

116,000+

International authors and editors

120M+

Downloads

151

Countries delivered to

Our authors are among the
Top 1%

most cited scientists

12.2%

Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



Meet the authors



Zhen Qin is currently an associate professor in the School of Information and Software Engineering, University of Electronic Science and Technology of China (UESTC). He received his PhD degree from UESTC in 2012. He was a visiting scholar in the Department of Electrical Engineering and Computer Science at Northwestern University. His research interests include network measurement, mobile social networks, and wireless sensor networks.



Erqiang Zhou received his PhD degree with the School of Computing from Dublin Institute of Technology, Dublin, Ireland, in 2011. Erqiang is currently an associate professor with the School of Information and Software Engineering. His research interests include question answering, opinion mining, and sentiment analysis.



Yi Ding received his PhD degree with the school of Computer Science from the Dublin Institute of Technology, Dublin, Ireland, in 2012, where he is currently an associate professor with the School of Information and Software Engineering. His research interests include machine learning and deep learning.



Yang Zhao received his PhD degree with the School of Computer Science and Engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 2009, where he is currently an associate professor with the School of Information and Software Engineering. His research interests include information security and applications.



Fuhu Deng is currently a lecturer in the School of Information and Software Engineering, University of Electronic Science and Technology of China. He received his PhD degree from Dublin Institute of Technology in 2014. His research interests include wireless LAN network and bandwidth estimation.



Hu Xiong received his PhD degree with the School of Computer Science and Engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 2009, where he is currently an associate professor with the School of Information and Software Engineering. His research interests include cryptographic protocols and network security.

Contents

Preface	XIII
Section 1 Setting the Stage	1
Chapter 1 Why Outsourcing and Privacy Protection Matter for Data Service in the Mobile Internet	3
Chapter 2 Foundations	16
Section 2 Outsourcing for Data Services in Mobile Internet	47
Chapter 3 Information Retrieval	49
Chapter 4 Classical Machine Learning	93
Chapter 5 Deep Learning	119
Section 3 Privacy Preserving for Data Services in Mobile Internet	155
Chapter 6 Attribute-Based Encryption for Flexible and Fine-Grained Access Control	157
Chapter 7 Proxy Re-Encryption for Secure Access Delegation	193

Preface

The data of mobile Internet have the characteristics of large scale, variety of patterns, complex association and so on. On the one hand, it needs efficient data processing model to provide support for data services, and on the other hand, it needs certain computing resources to provide data security services. Due to the limited resources of mobile terminals, it is impossible to complete large-scale data computation and storage. However, outsourcing to third parties may cause some risks in user privacy protection.

This monograph focuses on key technologies of data service outsourcing and privacy protection in mobile Internet, including the existing methods of data analysis and processing, the fine-grained data access control through effective user privacy protection mechanism, and the data sharing in the mobile Internet, which provide technical support for improving various service applications based on mobile Internet. The financial support from the National Natural Science Foundation of China under grants No.61672135, the Sichuan Science-Technology Support Plan Program under grants No.2018GZ0236, No.2017FZ0004 and No.2016JZ0020, the National Science Foundation of China - Guangdong Joint Foundation under grants No.U1401257 are also greatly appreciated.

Needless to say, any errors and omissions that still remain are totally our own responsibility and we would greatly appreciate the feedbacks about them from the readers. Thus, please send your comments to “qinzhen@uestc.edu.cn” and put “Data Service Outsourcing and Privacy Protection in Mobile Internet” in the subject line.

Zhen Qin
University of Electronic Science and Technology of China
March, 2018

Section 1

Setting the Stage

Why Outsourcing and Privacy Protection Matter for Data Service in the Mobile Internet

1.1. Mobile Internet

With the continuous development of science and technology, the mobile Internet (MI) has become an important part of our daily lives. It is a product of the convergence of mobile communication technologies and Internet technologies, which has a wide range of network coverage, convenience, and instant features. Smartphone is the main carrier of the mobile Internet. Through smartphones, people can make voice calls, send text messages, and take video communications as well as surf the Internet at any time (including browsing the web, sending and receiving e-mails, watching movies, etc.). In other words, using smartphones, users are more capable of creating, implementing, and managing novel and efficient mobile applications. Nowadays, the MI has become an important portal and major innovation platform for Internet businesses and an important hub for the exchange of information resources among various social media, business service companies, and new application software.

1.1.1 Mobile Internet

The mobile Internet, defined as the platform where the user's wireless access to the digitized contents of the Internet via mobile devices, consists of the following:

- Internet service provider is an organization that provides services such as Internet access, Internet transit, domain name registration, web hosting, Usenet service, and collocation.
- Users can get different services from the mobile Internet such as points of interest services, music services, fitness services, and mobile commerce services by using mobile applications.

In the MI, mobile devices, which include smartphones, tablets, PCs, e-books, MID, etc., enable users to stay connected with not only the network service providers through LTE and NFC techniques but also the physically close peers through short-range wireless communication techniques. In fact, each individual user has a personalized behavior pattern, which can influence the reliability and efficiency of the network in the MI. Moreover, the MI has a wide range of applications such as wireless body area networks (WBANs), mobile e-commerce, mobile social networks, mobile ad hoc networks, etc. The common scenarios of the MI will be shown in **Figure 1.1**.

1.1.2 Application

The mobile Internet offers a huge number of applications to the users. Most of these applications have many unique features or special functions, but the main



Figure 1.1.
The common scenarios of the MI.

technology of the applications remains the same as other services. The popular applications in the mobile Internet are briefly reviewed below.

1.1.2.1 Mobile social networks

Social networking has already become an important integral part of our daily lives, enabling us to contact our friends and families. Nowadays, the pervasive use of the mobile Internet and the social connections fosters a promising mobile social network (MSN) where reliable, comfortable, and efficient computation and communication tools are provided to improve the quality of our work and life.

MSN is social networking where individuals with similar interests converse and connect with one another through their mobile phone and/or tablet. Much like web-based social networking, mobile social networking occurs in virtual communities [1].

Similar to many online social networking sites, such as Facebook and Twitter, there are just as many social networks on mobile devices. They offer vast number of functions including multimedia posts, photo sharing, and instant messaging. Most of these mobile applications offer free international calling and texting capabilities. Today, social networking applications are not just for the social aspect but are frequently used for professional aspects as well, such as LinkedIn, which is still constantly growing. Along with sharing multimedia posts and instant messaging, social networks are commonly used to connect immigrants in a new country. While the thought of moving to a new country may be intimidating for many, social media can be used to connect immigrants of the same land together to make assimilation a little less stressful [2].

1.1.2.2 Mobile ad hoc networks

Due to the limitation of the applications in the wire communication, the need of the extension of applications via the mobile Internet becomes necessary. The

growth of laptops and 802.11/Wi-Fi wireless networks has made mobile ad hoc networks (MANETs) a popular research topic since the mid-1990s.

A mobile ad hoc network (MANET), also known as wireless ad hoc network [3] or ad hoc wireless network, is continuously self-configuring, infrastructure-less network of mobile devices connected using wireless channels [4]. Each device in a MANET is free to move independently in any direction and will therefore change its links to other devices frequently. Such networks may operate by themselves or may be connected to the larger Internet. They may contain one or multiple and different transceivers between nodes. This results in a highly dynamic, autonomous topology [5].

MANETs can be used in many applications, ranging from sensors for environment, vehicular ad hoc communications, road safety, smart home, peer-to-peer messaging, air/land/navy defense, weapons, robots, etc. A typical instance is health monitoring, which may include heart rate, blood pressure, etc. [6]. This can be constant, in the case of a patient in a hospital or event driven in the case of a wearable sensor that automatically reports your location to an ambulance team in case of an emergency. Animals can have sensors attached to them in order to track their movements for migration patterns, feeding habits, or other research purposes [7]. Another example is used in disaster rescue operations. Sensors may also be attached to unmanned aerial vehicles (UAVs) for surveillance or environment mapping [8]. In the case of autonomous UAV-aided search and rescue, this would be considered an event mapping application, since the UAVs are deployed to search an area but will only transmit data back when a person has been found.

1.1.2.3 Mobile payment

Since the rapid economic growth, the methods of payment have changed a lot all over the world. Given by the development of the mobile Internet, the mobile payment becomes more pervasive in Asian countries, especially in China.

Mobile payment (also referred to as mobile money, mobile money transfer, and mobile wallet) generally refers to payment services operated under financial regulation and performed from or via a mobile device. Instead of paying with cash, check, or credit cards, a consumer can use a mobile to pay for a wide range of services and digital or hard goods.

Although the concept of using non-coin-based currency systems has a long history [9], it is only recently that the technology to support such systems has become widely available.

1.1.3 Mobile Internet and user behavior

Since entering the twenty-first century, the China's Internet has developed rapidly. In particular, the mobile Internet further enhances this speed. We soon discovered that today's Internet is used in e-commerce shopping, map navigation, mobile payment, stock financing, online booking, government office, recharge, and other aspects that are completely different from the original. It seems that everything can be done on the Internet. You can rely on your phone instead of taking it out. In the mobile Internet, users and mobile terminals are closely related. The accumulation of user behavior in the mobile Internet is of great value. Smartphones and other mobile terminals are no longer simple communication tools but become mobile terminals that integrate communication, computing, and sensing functions. It can continuously generate and collect various data information [10]. Various sensing devices on the mobile terminal can sense the surrounding environment and generate various data information; at the same time, the mobile terminal serves as

an entrance for the user to enter the mobile Internet and can collect data information generated by other users. By combining various data related to mobile users, user behaviors can be modeled and predicted, and important statistical data for specific services can be extracted. For example, mobile users' GPS information and historical travel habits can be used to generate traffic statistics and travel advice [11]. It can be seen that, based on the data information generated in the mobile Internet, we can satisfy the different requirements of different users in different scenarios in the first time by judging the user information needs and accurately sorting the information categories, thereby providing users with personalized high-quality information and improving users' service experience.

The relevant data show that in the 3 years of rapid development of the mobile Internet, the total amount of data and information generated by humans has exceeded 400 years in the past. The global data volume has reached 1.8 ZB in 2011, and it is expected that the global data volume will reach 35 ZB by 2020. However, with the accumulation of data information, the amount of data available for analysis and mining continues to increase, and the entropy of information (i.e., the ratio of total information to valuable information) tends to increase. The increasing amount of data information makes information overload, information redundancy, and information search become new issues in the mobile Internet era. Search engines and recommendation systems provide very important technical means for solving these problems. When users search for information on the Internet, the search engine performs information matching in the background of the system with the keywords from the users and displays the results for users. However, if the user cannot accurately describe the keywords they need, then the search engine is powerless. Unlike search engines, the recommendation system does not require users to provide explicit requirements but simulates the user's interests by analyzing their historical behavior, thereby actively recommending to users' information that meets their interests and needs. In recent years, e-commerce has been booming, and the dominant position of the recommendation system in the Internet has become increasingly apparent. However, in the mobile Internet, data are not only more and more large scale but also have a variety of models, complex relationships, and reliability uncertainties. Diversified models make the data content difficult to understand. The complexity of the association makes the data difficult to be effectively identified. The uncertainty of credibility makes it difficult to determine the authenticity of the data. Such data features make the perception, expression, understanding, and calculation of the data face new challenges. The complexity of space-time complexity in the traditional computing model greatly limits our ability to design data-efficient computing models for the mobile Internet. How to quantify, define, and extract the essential characteristics and internal correlation of data in the mobile Internet, then study its internal mechanism, and provide support for the recommendation service is an urgent problem to be solved.

1.1.4 Behavior and data

At the same time, due to the characteristics of large scale, diverse models, and complex relationships in data in the mobile Internet, data analysis and processing processes have certain requirements for computing resources, while mobile terminals such as smartphones have limited computing resources (such as computational processing capabilities, etc.). Therefore, analysis and processing of the mobile Internet data is often done on devices such as third-party data processing servers that have sufficient computing resources. However, users and mobile terminals in the mobile Internet are closely related, and their data contains a large amount of user privacy information. This will bring a series of security issues. Since the users

facing the outsourced data application system are not a specific user, there may be any users who are connected to the Internet. These users can submit data query requests to outsourcing. Different users have different requirements for outsourcing. It is precisely because of this particularity that the outsourcing system may be completely open to any external potential users. It is not enough to protect the privacy of outsourced data only through system-level access control, cryptography, and other technical means. According to statistics from the Beijing Zhongguancun police station, telecommunication fraud reported for the whole year of 2012 accounted for 32% of the cases filed, which is the highest proportion of the types of crimes. Fraudulent often uses six kinds of methods:

- The impersonation of an individual or a circle of friends after the disclosure of information, such as criminals posing as frauds by public security organs, postal services, telecommunications, banks, social security staff, or friends and relatives, accounts for 42% of the total number of fraud cases.
- After the leakage of shopping information, the seller is guilty of fraud.
- The winning fraud is caused by the leakage of telephone, QQ, or e-mail.
- The false recruitment information is received after job information leakage.
- The Internet dating fraud after the leakage of dating information.
- Kidnapping fraud after the disclosure of family information. It can be seen that many third parties have disclosed users' personal information to varying degrees.

When people realized that they wanted to protect their privacy and tried to hide their actions, they did not think that their actions were already on the Internet; especially, different social locations in the network generate many data footprints. This kind of data has the characteristics of being cumulative and relevant. The information of a single place may not reveal the privacy of the user, but if a lot of people's behaviors are gathered from different independent locations, his privacy will be exposed because there has been enough information about him. This hidden data exposure is often unpredictable and controllable by individuals. The leakage of personal privacy information has caused panic among some users, which has negatively affected the sharing of data resources in the mobile Internet. The issue of privacy protection has become increasingly important.

As a carrier of personal information, mobile smart terminals are characterized by mobility, diversity, and intelligence. The various types of applications attached to them present complex types, their execution mechanisms are not disclosed, a large amount of data, etc. These factors are superimposed and cause them to face much greater threats than before. In terms of presentation, the security issues of mobile smart terminals are mainly concentrated on the following aspects: firstly, illegally disseminating content; secondly, maliciously absorbing fees; thirdly, user privacy is stolen; and fourthly, mobile terminal viruses and illegal brushing cause the black screen, system crashes, and other issues. The following is an example of the mobile phone malicious code and user privacy theft. Due to the mobile malicious code problem caused by the negligence of the design and implementation process of mobile smart terminal operating systems, more or less system vulnerabilities may be exploited by malicious code. In particular, with the continuous maturation of the mobile Internet hacking technology, malicious code for mobile smart terminals poses a great threat to their security. Mobile malicious code is a

destructive malicious mobile terminal program. It usually uses SMS, MMS, e-mail, and browsing websites through mobile communication networks or uses infrared and Bluetooth to transfer between mobile smart terminals. At present, the highest market share of mobile smart terminal operating system is Google Android operating system, followed by Apple iOS operating system and Microsoft Windows Phone operating system. As smart terminal operating systems become more and more uniform, the spread of malicious code has been accelerated, and it has evolved from a simple suction to sophisticated fraud and hooliganism. In addition, with the increase in the binding of mobile terminals and bank accounts, such as mobile terminals and third-party payment, the generation and spread of mobile phone viruses are accelerating toward the direction of traffic consumption, malicious deductions, and private information theft. In particular, with the rapid development of the Android-based operating system, mobile smart terminals using the operating system have gradually become the main targets of hacker attacks. In the first half of 2015, there were 5.967 million new Android virus data packets nationwide, an increase of 17.41% year on year, and the number of infected users reached 140 million, an increase of 58% from the same period last year. The total number of mobile payment users reached 11.455 million. How to outsource the data storage in the mobile Internet to a third-party server securely under the premise of ensuring the privacy of users and the realization of fine-grained access control of mobile Internet data is an urgent problem to be solved.

1.2. Research topics of the mobile Internet

1.2.1 Ontology-based information retrieval

Information retrieval is an important way of information exchange between users and data, which have been studied by many researches. Traditional ways of information retrieval are always based on keyword index and retrieval document. But the data on the mobile Internet have four features: large scale, complicated modules, diversified data format, and incongruous data sources. So, there is always a problem that different keywords can be the same meaning. The results of the search can be inaccurate or incomplete.

To deal with this situation, paper [12] gives us the definition of ontology-based information retrieval. This model includes the four main processes of an IR system: indexing, querying, searching, and ranking. However, as opposed to traditional keyword-based IR models, in this approach, the query is expressed in terms of an ontology-based query language (SPARQL), and the external resources used for indexing and query processing consist of an ontology and its corresponding KB. The indexing process is equivalent to a semantic annotation process. Instead of creating an inverted index where the keywords are associated with the documents where they appear, in the case of the ontology-based IR model, the inverted index contains semantic entities (meanings) associated with the documents where they appear. The relation or association between a semantic entity and a document is called annotation. The overall retrieval process consists of the following steps:

- The system takes as input a formal SPARQL query.
- The SPARQL query is executed against a KB, returning a list of semantic entities that satisfy the query. This process is purely Boolean (i.e., based on an exact match), so that the returned instances must strictly hold all the conditions of the formal query.

- The documents that are annotated (indexed) with the above instances are retrieved, ranked, and presented to the user. In contrast to the previous phase, the document retrieval phase is based on an approximate match, since the relation between a document and the concepts that annotate it has an inherent degree of fuzziness.

The steps listed above are described in more detail in the following subsections, from indexing to query processing, document retrieval, and ranking.

In this model, it is assumed that a KB has been built and associated with the information sources (the document base), by using one or several domain ontologies that describe concepts appearing in a document text. The concepts and instances in the KB are linked to the documents by means of explicit, non-embedded annotations of the documents. In this model, keywords appearing in a document are assigned weights reflecting the fact that some words are better at discriminating between documents than others. Similarly, in this system, annotations are assigned weights that reflect the discriminative power of instances with respect to the documents. And, the weight d_x of an instance x for a document d is computed as

$$d_x = \frac{freq_{x,d}}{\max_y freq_{y,d}} \cdot \log \frac{|D|}{n_x} \quad (1.1)$$

where $freq_{x,d}$ is the number of occurrences in d of the keywords attached to x and D is the set of all documents in the search space.

The query execution returns a set of tuples that satisfy the SPARQL query. Then, extract the semantic entities from those tuples, and access the semantic index to collect all the documents in the repository that are annotated with these semantic entities. Once the list of documents is formed, the search engine computes a semantic similarity value between the query and each document, using an adaptation of the classic vector space IR model. Each document in the search space is represented as a document vector where each element corresponds to a semantic entity. The value of an element is the weight of the annotation between the document and the semantic entity, if such annotation exists, and zero otherwise. The query vector is generated weighting the variables in the SELECT clause of the SPARQL query. For testing purposes, the weight of each variable of the query was set to 1, but in the original model, users are allowed to manually set this weight according to their interest. Once the vectors are constructed, the similarity measure between a document d and the query q is computed as

$$sim(d, q) = \frac{d \times q}{|d| \cdot |q|} \quad (1.2)$$

If the knowledge in the KB is incomplete (e.g., there are documents about travel offers in the knowledge source, but the corresponding instances are missing in the KB), the semantic ranking algorithm performs very poorly: SPARQL queries will return less results than expected, and the relevant documents will not be retrieved or will get a much lower similarity value than they should. As limited as might be, keyword-based search will likely perform better in these cases. To cope with this, the ranking function combines the semantic similarity measure with the similarity measure of a keyword-based algorithm. Combine the output of search engines, and compute the final score as

$$\lambda \cdot sim(d, q) + (1 - \lambda) \cdot ksim(d, q) \quad (1.3)$$

where $ksim$ is computed by a keyword-based algorithm and $\lambda \in [0, 1]$.

A lot of information retrieval models have been put forward to improve the performance of information retrieval, and some of the combinations of these technologies have been proven to be effective. But faced with the rich data in the MI, the great improvement needs to be done for getting the search more intelligent and efficient.

1.2.2 Deep learning

Machine learning is a field of computer science that gives computer systems the ability to “learn” (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed. It learns from the external input data through algorithm analysis to obtain the regularity for recognition and judgment. With the development of machine learning, it is classified as shallow learning and deep learning and plays an important role in the field of artificial intelligence. Due to the large-scale data, multiple model, and complex correlation, traditional shallow learning algorithms (such as the support vector machine algorithm [13]) are still subject to certain restrictions in dealing with complex classifications [14] even if widely used.

Deep learning is a new field in the study of machine learning; the motive is to establish and simulate human brain analysis neural network to learn. It mimics the human brain to explain the mechanism of data, such as images, sound, and text. As with machine learning methods, deep machine learning methods also have supervised learning and unsupervised learning. Supervised learning algorithms experience a data set containing features, but each example is also associated with a label or target. For example, the iris data set is annotated with the species of each iris plant. A supervised learning algorithm can study the iris data set and learn to classify iris plants into three different species based on their measurements. Unsupervised learning algorithms experience a data set containing many features and then learn useful properties of the structure of this data set. In the context of deep learning, we usually want to learn the entire probability distribution that generated a data set, whether explicitly as in density estimation or implicitly for tasks like synthesis or deionizing. Some other unsupervised learning algorithms perform other roles, like clustering, which consists of dividing the data set into clusters of similar examples. The learning model established under different learning frameworks is very different. For example, the convolutional neural network (CNN) is a kind of machine learning model of deep learning under the supervision of the deep place letter (deep belief networks, referred to as DBNs), which is a kind of machine learning model in the case of unsupervised learning.

The concept of deep learning was proposed in 2006 by Hinton [15]. Based on the deep belief network (DBN), the algorithm is proposed to solve the optimization problem of the deep structure, and then the deep structure of multilayer automatic encoder is proposed. In addition, the convolution neural network proposed by Lecun et al. [16] is the first real multilayer structure learning algorithm, which uses the space relative relation to reduce the number of parameters to improve the training performance. Deep learning is a method of representing learning in machine learning. Observations (such as an image) can be represented in a variety of ways, such as vectors for each pixel's strength value, or more abstracted into a range of edges, specific shapes, etc. It is easier to learn a task from an instance using some specific presentation methods (e.g., face recognition or facial expression recognition). The advantage of deep learning is to replace the manual acquisition feature with non-supervised or semi-supervised feature learning and hierarchical feature extraction.

Deep learning architectures such as deep neural networks, deep belief networks, and recurrent neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, and drug design [13], where they have produced results comparable to and in some cases superior [14] to human experts.

The work [17] firstly proposed a novel context-dependent (CD) model for large vocabulary speech recognition (LVSR). This model is a pre-trained deep neural network-hidden Markov model (DNN-HMM) hybrid architecture that trains the DNN to produce a distribution over senones (tied triphone states) as its output. Although the experiments show that CD-DNN-HMMs provide dramatic improvements in recognition accuracy, many issues remain to be resolved. First, although CD-DNN-HMM training is asymptotically quite scalable, in practice, it is quite challenging to train CD-DNN-HMMs on tens of thousands of hours of data. Second, highly effective speaker and environment adaptation algorithms for DNN-HMMs must be found, ideally ones that are completely unsupervised and integrated with the pre-training phase. Third, the training in this study used the embedded Viterbi algorithm, which is not optimal. In addition, the study views the treatment of the time dimension of speech by DNN-HMM and GMM-HMMs alike as a very crude way of dealing with the intricate temporal properties of speech. Finally, although Gaussian RBMs can learn an initial distributed representation of their input, they still produce a diagonal covariance Gaussian for the conditional distribution over the input space given the latent state (as diagonal covariance GMMs also do). Ji et al. [18] developed a 3D CNN model for action recognition. This model construct features from both spatial and temporal dimensions by performing 3D convolutions. The developed deep architecture generates multiple channels of information from adjacent input frames and performs convolution and subsampling separately in each channel. The final feature representation is computed by combining information from all channels. Evaluated by the TRECVID and the KTH data sets, the results show that the 3D CNN model outperforms compared with the methods on the TRECVID data, while it achieves competitive performance on the KTH data, demonstrating its superior performance in real-world environments. Furthermore, Cho et al. [19] proposed a novel neural network model called RNN encoder-decoder that is able to learn the mapping from a sequence of an arbitrary length to another sequence, possibly from a different set, of an arbitrary length. The encoder and decoder of the proposed model are jointly trained to maximize the conditional probability of a target sequence given a source sequence. The performance of a statistical machine translation system is empirically found to improve by using the conditional probabilities of phrase pairs computed by the RNN encoder-decoder as an additional feature in the existing log-linear model. But one approach that was not investigated is to replace the whole or a part of the phrase table by letting the RNN encoder-decoder propose target phrases. A multi-context deep learning framework for saliency detection is proposed in [20]. Different pre-training strategies are investigated to learn the deep model for saliency detection, and a task-specific pre-training scheme for the presented multi-context deep model is designed. Tested by the contemporary deep models in ImageNet image classification challenge, their effectiveness in saliency detection is investigated.

Although deep learning has shown great success in various applications such as objection recognition [21] and speech recognition [17], especially language modeling [22], paraphrase detection [23], and word embedding extraction [24], the work implemented in the MI is still worthy to investigate.

1.2.3 Outsourced attribute-based encryption system

Attribute-based encryption (ABE) is a promising cryptographic primitive, which has been widely applied to design fine-grained access control system recently. Data access control has been evolving in the past 30 years, and various techniques have been developed to effectively implement fine-grained access control [25], which allows flexibility in specifying differential access rights of individual users. However, traditional access control systems are mostly designed for in-house services and depend greatly on the system itself to enforce authorization policies. Thus, they cannot be applied in cloud computing because users and cloud servers are no longer in the same trusted domain. For the purpose of helping the data owner impose access control over data stored on untrusted cloud servers, a feasible consideration would be encrypting data through certain cryptographic primitives but disclosing decryption keys only to authorized users. One critical issue of this branch of approaches is how to achieve the desired security goals without introducing high complexity of key management and data encryption. Existing works resolve this issue either by introducing a per-file access control list (ACL) for fine-grained access control or by categorizing files into several filegroups for efficiency. As the system scales, however, the ACL-based scheme would introduce an extremely high complexity which could be proportional to the number of system users. The filegroup-based scheme, on the other hand, is just able to provide coarse-grained access control of data.

To provide fine-grained access control over encrypted data, a novel public-key primitive, namely, attribute-based encryption (ABE) [26], is introduced in the cryptographic community, which enables public-key-based one-to-many encryption. In ABE system, users' keys and ciphertexts are labeled with sets of descriptive attributes and access policies, respectively, and a particular key can decrypt a ciphertext only if the associated attributes and policy are matched. Though ABE is a promising primitive to design fine-grained access control system in cloud computing, there are several challenges remained in the application of ABE.

- One of the main drawbacks of ABE is that the computational cost in decryption phase grows with the number of attributes specified in the access policy. The drawback appears more serious for resource-constrained users such as mobile devices and sensors. Therefore, one challenge is how to reduce the decryption complexity of ABE such that it can be applied to fine-grained access control for users with resource-constrained devices.
- Beyond decryption, generating users' private key in existing ABE schemes also requires a great quantity of modular exponentiations. Furthermore, the revocation of any single user in existing ABE requires key update at authority for remaining users who share his/her attributes. All of these heavy tasks centralized at the authority side would make it become the efficiency bottleneck in the whole access control system. Therefore, another challenge is how to reduce the key-issuing complexity of ABE such that scalable access control can be supported.

To improve the practical application level of the attribute-based encryption system, researchers began to attempt to introduce outsourced computing to the attribute-based encryption system. In 2011, Green et al. [27] proposed a scheme that outsourced the decryption operation in the attribute-based cryptosystem to a third party in order to improve the client's operational efficiency and enable the attribute-based cryptosystem to run on devices with limited computational resources such as

smart cards. The ciphertext length in this scheme increases linearly with the number of access policy attributes. In [27], the third party uses the conversion key sent by the user to convert the encrypted ciphertext with the higher decryption cost into the ElGamal-encrypted ciphertext with the lower decryption cost and transmits the converted ciphertext back to users. And then, users can use their own decryption keys to complete the decryption locally. Li et al. [28] simplified the client's computational complexity in the attribute-based cryptosystem from the perspective of outsourced cryptographic operations. After outsourcing cryptographic operations, the user only needs to perform constant exponentiation operations for encryption regardless of the number of attributes and can encrypt any access policy. Li et al. [29] proposed an attribute-based encryption scheme that supports outsourced key generation and outsourced decryption operations. Meanwhile, the scheme proposed a fine-grained access control system based on this scheme. Lai et al. [30] pointed out that the solution proposed by the user in [27] validates the results returned by the third party and that the security of the solution depends on the random oracle model. On the basis of [27], the literature [30] proposes a verifiable outsourced attribute-based encryption scheme. By adding redundant information, users can verify the conversion results returned by third parties to ensure the accuracy of conversions. The security of the new program is based on the standard model. Li et al. [31] proposed an attribute-based encryption scheme that can both reduce attribute authority and user computing burden and also support user authentication, on the basis of [27, 28, 31] and so on. By outsourcing part of the key generation and decryption operations to different service providers, the attribute authority and the user's local computing burden are greatly reduced. Lin et al. [32] pointed out that the method of adding redundant information in [30] can not only achieve the purpose of verifying the returned result, but also increases the computational load and communication burden of the system. The literature [32] proposes an outsourced attribute-based encryption scheme that uses a hybrid encryption system and a commitment mechanism to complete user authentication. The length of the ciphertext and the calculation of cryptographic operations in this scheme are only half that of the scheme mentioned in [30], and the security of the scheme is based on the standard model. Ma et al. [33] presented the concept of innocent proof for the first time. The user who holds the private key cannot report cloud server operation errors and proposed an attribute-based encryption scheme that supports both outsourced encryption and decryption operations and verifiable and innocent proofs. Mao et al. [34] proposed an attribute-based encryption scheme that verifies outsource decryption, and experiments show that the scheme is more concise and less computational than cryptographic schemes proposed in [30] but still with the number of attributes linearly related. Xu et al. [35] proposed a circuit-based ciphertext-policy-based attribute-based hybrid encryption system to ensure data integrity in cloud computing environments and implement fine-grained access control and verifiable delegation mechanisms.

It can be known from the existing work that currently existing secure outsourced attribute-based encryption schemes cannot achieve a fixed ciphertext length, and these solutions not only improve the computational efficiency but also increase the communication load of mobile devices. However, the work of achieving fine-grained access control of data in the MI and promoting the sharing of data in the MI is still worthy to investigate.

1.2.4 Proxy re-encryption system

Since cloud users do not fully trust cloud computing service providers (CSPs), they do not want any unauthorized personnel (including cloud service providers) to

view their own data. This cannot rely on administrative constraints or ethics. The rules to be solved can only adopt technical means to ensure that cloud service providers and other non-authorized personnel cannot obtain any uploading cloud data from the customer, and any unauthorized changes to the customer data in the cloud will be accepted by the customers. Therefore, user data usually needs to be encrypted on the client before being transmitted to the cloud. However, in the cloud computing environment, there are a large number of data sharing scenarios. When data sharing is performed, if decryption is performed in the cloud, there is a risk of leakage of user data, which is not completely trusted by the cloud computing service provider for the user. The situation is impossible. If you need the user to re-encrypt the file, this is very inconvenient for the user. Therefore, the proxy re-encryption (PRE) scheme for distributed data storage can well solve this problem.

Proxy re-encryption (PRE), initially introduced by Blaze et al. [36], enables a semi-trusted proxy to transform a ciphertext encrypted under the public key of delegator into another ciphertext under the public key of delegatee without leaking the underlying encrypted messages or private keys of delegator/delegatee to the proxy. This special kind of public-key encryption seems to be an optimal candidate to ensure the security of sharing data in cloud computing. Supposedly, the data owner (say, Alice) intends to share the sensitive data stored in the cloud with another granted user (say, Bob). It is desirable that the requested data can be accessed by nobody other than Bob. Inspired by the primitive of PRE, Alice can encrypt the sensitive data under her own public key before uploading the shared data to the semi-trusted cloud. After receiving the request of data sharing from Bob, Alice generates a proxy re-encryption key using her own private key and Bob's public key and sends this proxy re-encryption key to the semi-trusted cloud server. Equipped with this proxy re-encryption key, cloud server can transform the ciphertext encrypted under the public key of Alice into an encryption under the public key of Bob. By utilizing the PRE primitive, the transformed ciphertext can only be decrypted by Bob, whereas the cloud server is unable to learn the plaintext or private keys of Alice or Bob. Finally, Bob can download and decrypt the requested data with his own private key. In this way, the costly burden of secure data sharing can be uploaded to the semi-trusted cloud server with abundant resources.

In 2005, Ateniese et al. [37] proposed the first unidirectional PRE scheme. Similar to the scheme of Blaze et al. [36], both of the schemes are secure only against chosen-plaintext attacks (CPA). In 2007, Canetti et al. [38] designed a bidirectional PRE scheme with chosen-ciphertext security. In 2008, Libert et al. [39] introduced a replayable chosen-ciphertext secure (RCCA) unidirectional PRE scheme. Since then, various PRE schemes have been proposed in the literature (e.g., [40–44]).

PRE can be extended in the context of identity-based encryption. In 2007, Green and Ateniese [45] proposed the first identity-based proxy re-encryption (IBPRE) scheme, which is CCA secure in the random oracle model, where hash functions are assumed to be fully random. Chu and Tzeng [30] constructed a CCA secure IBPRE scheme in the standard model. After that, many identity-based proxy re-encryption (IBPRE) schemes have been proposed, such as [47–53].

However, among all of the aforementioned schemes, the semi-trusted proxy can use a given re-encryption key to transform all the ciphertexts of a delegator into those of a delegatee. But in reality, the delegator does not want to transform all of his data for the delegatee. Therefore, type-based PRE [54] and conditional PRE (CPRE) [55, 56] were proposed, in which the proxy can only fulfill ciphertext conversion conditionally. Later, Liang et al. [57, 58] proposed two IBCPRE schemes with CCA secure in the standard model. However, He et al. [59] presented the

security analysis to show that their schemes only achieve CPA security. In 2016, He et al. [60] proposed an efficient identity-based conditional proxy re-encryption (IBCPRE) scheme with CCA secure in the random oracle model.

PRE can be extended in the attribute-based setting. Attribute-based proxy re-encryption (ABPRE) can effectively increase the flexibility of data sharing. In 2009, Liang et al. [61] first defined the notion of ciphertext-policy ABPRE (CP-ABPRE), where each ciphertext is labeled with a set of descriptive conditions and each re-encryption key is associated with an access tree that specifies which type of ciphertexts the proxy can re-encrypt, and they presented a concrete scheme supporting and gates with positive and negative attributes. After that, several CP-ABPRE schemes (e.g., [62]) with more expressive access policy were proposed. In 2011, Fang et al. [63] proposed a key-policy ABPRE (KP-ABPRE) scheme in the random oracle model, whereby ciphertext encrypted with conditions W can be re-encrypted by the proxy using the CPRE key under the access structure TT if and only if $T(W)T(W) = 1$. More recent ABPRE systems can be seen in [64–66].

In 2016, Lee et al. [67] proposed a searchable hierarchical CPRE (HCPRE) scheme for cloud storage services, and cloud service provider is able to generate a hierarchical key, but the re-encryption key generation algorithm also requires the private keys of the delegator and delegatee.

So far, the proxy re-encryption has been extensively investigated. However, the concrete proxy re-encryption schemes with different properties still need to be proposed and applied to the special environment for the MI.

1.3. Organization

The core material of this book consists of the following:

- Chapters 1 and 2, discussing the importance of outsourcing and privacy protection for data service in the mobile Internet and the basics of data service outsourcing and privacy protection in the mobile Internet
- Chapters 3–5, introducing information retrieval, classical machine learning, and deep learning
- Chapters 6 and 7, introducing attribute-based encryption for flexible and fine-grained access control and motivating proxy re-encryption for secure access delegation

Foundations

The purpose of this chapter is to offer a brief review of all the necessary cryptographic concepts needed in the following chapters. We start with the mathematical backgrounds and associated hard problems. The other aspect of this chapter is some facts regarding the classical symmetric cryptography and public-key cryptosystem. Finally, a concise description of provable security will be presented in the rest of this chapter. We remark that this introduction is by no means exhaustive such that the approaches used to speed up pairing computations or select suitable parameters of elliptic curve are really beyond the scope of this book.

2.1. Mathematical concepts and properties

2.1.1 Concepts from number theory

2.1.1.1 Primes and divisibility

Let \mathbb{Z} be the set of integers. We say that a divides b (denoted as $a|b$) if there exists an integer c satisfying $ac = b$ for $a, b, c \in \mathbb{Z}$. If a does not divide b , we write $a \nmid b$. Despite that this definition makes sense even when one or more of these integers are negative or zero, we are only interested in the case when a, b, c are all positive. A direct observation is that if $a|b$ and $a|c$, then $a|(xb + yc)$ for any $x, y \in \mathbb{Z}$.

If $a|b$ and a is positive, a is regarded as a divisor or factor of b . Furthermore, a is also called a *nontrivial* factor of b in case $a \notin \{1, b\}$. A positive integer $p (> 1)$ is considered as *prime* if it has no nontrivial factors, i.e., it has only two divisors: 1 and p itself. A positive integer greater than 1 is called *composite* if this integer is not a prime number. It is the convention to know that “1” is neither prime nor composite.

According to the fundamental theorem of arithmetic, every integer greater than 1 can be expressed *uniquely* as a product of primes. Namely, any positive integer $n > 1$ can be written as $n = \prod_i p_i^{e_i}$ such that $\{p_i\}$ are distinct primes and $e_i \geq 1$ for all i ; furthermore, the $\{p_i\}$ and $\{e_i\}$ are uniquely determined up to ordering.

Proposition 2.1

Let a and b be two integers and $b > 0$. Then there exist unique integers q, r satisfying $a = qb + r$ and $0 \leq r < b$.

c is known as the *greatest common divisor* of two nonnegative integers a and b (written as $\gcd(a, b)$) if c is the largest integer satisfying $c|a$ and $c|b$. It is noted that $\gcd(0, 0)$ are not defined, whereas $\gcd(b, 0) = \gcd(0, b) = b$. The notion of greatest common divisor also makes sense when either or both of a, b are negative, but we will never need this; therefore, when we write $\gcd(a, b)$, we always assume that $a, b \geq 0$. Note that if p is prime, then $\gcd(a, p)$ is either equal to 1 or p . a and b are called *relatively prime* if $\gcd(a, b) = 1$.

Proposition 2.2

Let a, b be two positive integers. Then there exist $x, y \in \mathbb{Z}$ such that $ax + by = \gcd(a, b)$. Furthermore, $\gcd(a, b)$ is the smallest positive integer that can be expressed in this manner.

Proof. Let I denote the set $\{ax + by\}$ where x, y are chosen from \mathbb{Z} . Note that I certainly contains some positive integers since at least $a, b \in I$ such that $a = a \times 1 + b \times 0$ and $b = a \times 0 + b \times 1$. Let d denote the smallest positive integer in I . We claim that $d = \gcd(a, b)$ due to the fact that d can be represented as $d = ax + by$ for some $x, y \in \mathbb{Z}$ (recall that $d \in I$). In addition, to show that $d = \gcd(a, b)$, we must show that $d|a$ and $d|b$ and that d is the largest integer with this property. In fact, we can show that d divides every element in I . To prove this, take arbitrary $c \in I$ such that $c = ax' + by'$ with $x', y' \in \mathbb{Z}$. According to Proposition 2.1, we have $c = qd + r$ with q and r being integers and $0 \leq r < d$. Then

$$r = c - qd = ax' + by' - q(ax + by) = (x' - qx)a + (y' - qy)b \in I. \quad (2.1)$$

There is a contradiction between $r \neq 0$ and the choice of d as the smallest positive integer in I . Thus, it is obvious that $r = 0$ and hence $d|c$.

Since both a and b are elements in I , the above shows that $d|a$ and $d|b$. Suppose there exists $d' > d$ such that $d'|a$ and $d'|b$. Then $d'|ax + by$; since the latter is equal to d , this means $d'|d$, which is impossible if d' is larger than d . It is concluded that d is the largest integer dividing both a and b and hence $d = \gcd(a, b)$.

Given a and b , the *Euclidean algorithm* can be used to compute $\gcd(a, b)$ in polynomial time. The *extended Euclidean algorithm* can be used to compute the coefficients x, y (as in Proposition 2.1) in polynomial time as well. Interested readers can refer to [68] for more details.

Proposition 2.3

If $c|ab$ and $\gcd(a, c) = 1$, then $c|b$. In particular, if p is prime and $p|ab$, then either $p|a$ or $p|b$.

Proof. It is easy to observe that $c \times \alpha = ab$ for some integer α since $c|ab$. From $\gcd(a, c) = 1$ then, according to the previous proposition, there exist integers x, y such that $1 = ax + cy$. Multiplying both sides by b , we obtain

$$b = axb + cyb = abx + cyb = c \times \alpha x + cyb = c \cdot (ax + yb). \quad (2.2)$$

Since $(ax + yb)$ is an integer, it follows that $c|b$.

The second part of this proposition follows from the fact that if $p \nmid a$, then $\gcd(a, p) = 1$.

Proposition 2.4

If $p|N, q|N$, and $\gcd(p, q) = 1$, then $pq|N$.

Proof. We can see $pa = N$ and $qb = N$ from $p|N$ and $q|N$ and obtain $1 = px + qy$ from Proposition 2.2, where α, β, x, y are all integers. Multiplying both sides of the last equation by N , we obtain $N = pxN + qyN = pxq\beta + qyp\alpha = pq(x\beta + y\alpha)$. Thus, $pq|N$.

2.1.1.2 Modular arithmetic

Let a, b, N be integers with $N > 1$. Let $[a \bmod N]$ denote the remainder of $a \in \mathbb{Z}$ upon division by N . In more detail, according to Proposition 2.1, there exist unique q and r such that $a = qN + r$ and $0 \leq r < N$, and $[a \bmod N]$ is defined as this remainder r . Namely, $0 \leq [a \bmod N] < N$. The process of mapping a to $[a \bmod N]$ is called *reduction modulo N*.

a and b are regarded as congruent modulo N (written as $a = b \bmod N$) if $[a \bmod N] = [b \bmod N]$, that is to say, the remainder when a is divided by N is equivalent to the remainder when b is divided by N . In this manner, $a = b \bmod N$ if and only if $N|(a - b)$. Note that $a = [b \bmod N]$ implies $a = b \bmod N$, but not vice versa. For example, $36 = 18 \bmod 18$ but $36 \neq [18 \bmod 18] = 0$.

We remark that congruence modulo N is an equivalence relation, i.e., it is reflexive ($a = a \pmod N$ for all a), symmetric ($a = b \pmod N$ implies $b = a \pmod N$), and transitive (if $a = b \pmod N$ and $b = c \pmod N$, then $a = c \pmod N$). Congruence modulo N also obeys the standard rules of arithmetic with respect to addition, subtraction, and multiplication, so if $a = a' \pmod N$ and $b = b' \pmod N$, then $(a + b) = (a' + b') \pmod N$ and $ab = a'b' \pmod N$. Thus, the calculations of congruence modulo N can be simplified by “reducing and then adding/multiplying” instead of “adding/multiplying and then reducing.”

Example 1. *Let us compute $[3193015 \times 590702 \pmod{100}]$. Since $3193015 = 15 \pmod{100}$ and $590702 = 2 \pmod{100}$, we have*

$$\begin{aligned} 3193015 \times 590702 &= [3193015 \pmod{100}] \cdot [590702 \pmod{100}] \pmod{100} \\ &= 15 \times 2 = 30 \pmod{100}. \end{aligned}$$

The cost of alternate approach to derive the answer (viz., computing the product 3193015×590702 and then reducing the answer modulo 100) is much more expensive.

Different from addition, subtraction, and multiplication, the division operation in congruence modulo N has not been involved. That is to say, if $a = a' \pmod N$ and $b = b' \pmod N$, then it is not necessarily true that $a/b = a'/b' \pmod N$; in fact, the expression “ $a/b \pmod N$ ” is not, in general, defined well. As a specific example related to the division, $ab = cb \pmod N$ does not necessarily imply that $a = c \pmod N$.

Example 2. *Take $N = 12$. Then $3 \times 3 = 9 = 3 \times 7 \pmod{12}$, but $3 \not\equiv 7 \pmod{12}$.*

However, a meaningful notion of division has also been defined for congruence modulo N . If for a given integer a there exists an integer a^{-1} such that $a \times a^{-1} = 1 \pmod N$, a^{-1} is viewed as a (multiplicative) inverse of a modulo N , and a is considered as invertible modulo N . It is obvious to observe that if α is a multiplicative inverse of a modulo N , then so is $[\alpha \pmod N]$; furthermore, if α' is another multiplicative inverse, then $[\alpha \pmod N] = [\alpha' \pmod N]$. We can simply let a^{-1} denote the unique multiplicative inverse of a that lies in the range $\{0, \dots, N - 1\}$ if a is invertible.

In this way, division by a modulo N is defined as multiplication by a^{-1} modulo N if a is invertible modulo N (i.e., c/a is defined as $c \times a^{-1} \pmod N$). We stress that division by a is only defined when a is invertible. If $c \times a = b \times a \pmod N$ and a are invertible, then we may divide each side of the equation by a (or, equivalently, multiply each side by a^{-1}) to obtain

$$(c \times a) \times a^{-1} = (b \times a) \times a^{-1} \pmod N \Rightarrow c = b \pmod N. \quad (2.3)$$

We see that in this case, division works “as expected” by adopting the idea of invertible integers. The natural question is that which integers are invertible modulo of a given modulus N . To answer this question fully, Proposition 2.2 is used in the following proposition:

Proposition 2.5

Let a, N be integers with $N > 1$. Then a is invertible modulo if and only if $\gcd(a, N) = 1$.

Proof. Assume a is invertible modulo N , and let b denote its inverse. It is evident that $a \neq 0$ since $0 \times b = 0 \pmod N$ regardless of the value of b . Since $a \times b = 1 \pmod N$, the definition of congruence modulo N implies that $a \times b - 1 = \alpha \times N$ for some $\alpha \in \mathbb{Z}$. Equivalently, $b \times a - \alpha \times N = 1$. According to Proposition 2.2, this implies $\gcd(a, N) = 1$.

Conversely, if $\gcd(a, N) = 1$, then according to Proposition 2.2, there exist integers x, y such that $ax + Ny = 1$. Reducing each side of this equation, modulo N gives $ax = 1 \pmod N$, and it is easy to see that x is a multiplicative inverse of a .

Example 3. Let $N = 13$ and $a = 10$. Then $10 \times 4 + (-3) \times 13 = 1$, and so $4 = [4 \bmod 13]$ is the inverse of 10. One can verify that $10 \times 4 = 1 \bmod 13$.

2.1.2 Concepts from abstract algebra

2.1.2.1 Group theory

Assume \mathbb{G} be a set. A *binary operation* \circ defined over the set \mathbb{G} (written as $g \circ h$ if $g, h \in \mathbb{G}$) is simply a function that takes as input two elements of \mathbb{G} and outputs another element in the set \mathbb{G} . The formal definition of a *group* is described as follows:

Definition 2.1. A group is a set \mathbb{G} pairing with a binary operation \circ such that:

Closure law: For all $g, h \in \mathbb{G}$, the result of $g \circ h$ still remains in the set \mathbb{G} .

Identity law: There exists an identity element $e \in \mathbb{G}$ such that for all $g \in \mathbb{G}$, $e \circ g = g = g \circ e$.

Inverse law: For all $g \in \mathbb{G}$, there exists an element $h \in \mathbb{G}$ such that $g \circ h = e = h \circ g$. Such an h is called an inverse element of g .

Associative law: For all $g_1, g_2, g_3 \in \mathbb{G}$, $(g_1 \circ g_2) \circ g_3 = g_1 \circ (g_2 \circ g_3)$.

In this way, the set \mathbb{G} along with the binary operation \circ is defined as a group. When the set \mathbb{G} has a finite number of elements, (\mathbb{G}, \circ) is viewed as a finite group, and $|\mathbb{G}|$ denotes the order of the group, that is, the number of elements in \mathbb{G} .

If $g \circ h = h \circ g$ for all $g, h \in \mathbb{G}$, this group is called abelian group or commutative group. In this book, we will always deal with finite and abelian (commutative) groups.

If (\mathbb{G}, \circ) is a group, a set $\mathbb{H} \subseteq \mathbb{G}$ pairing with the operation \circ is called a subgroup of (\mathbb{G}, \circ) if the set \mathbb{H} forms a group under the involved operation \circ . To check that (\mathbb{H}, \circ) is a subgroup, we need to verify closure, existence of identity and inverses, and associativity according to Definition 2.1. (In fact, associativity is inherited automatically from the group (\mathbb{G}, \circ) .) Every group (\mathbb{G}, \circ) always has the trivial subgroups (\mathbb{G}, \circ) and $(\{e\}, \circ)$. (\mathbb{H}, \circ) is called a *strict* subgroup of (\mathbb{G}, \circ) if $\mathbb{H} \neq \mathbb{G}$.

Associativity implies that the notation of long expression $g_1 \circ g_2 \circ \dots \circ g_n$ without parentheses is unambiguous since it does not matter in what order we perform the operation \circ .

It is easy to see that the identity element in a group (\mathbb{G}, \circ) is *unique*, and thus we can therefore refer to the identity element of a group. One can also demonstrate that each element g of a group has a unique inverse element.

In general, the abstract notation \circ will not be used to denote the group operation directly. Either the *additive* notation or the *multiplicative* notation will be used instead depending on the involved group. In case the additive notation has been adopted, the group operation applied to two elements g, h in the group is denoted as $g + h$; the identity element is denoted as “0,” and the inverse element of g is denoted as $-g$. In case the multiplicative notation has been employed, the group operation applied to g, h in the group is denoted as $g \times h$ or simply gh ; the identity element is denoted as “1,” and the inverse element of g is denoted as g^{-1} . As in the case of multiplication modulo N , we also define division by g as multiplication by g^{-1} (i.e., $[(h/g) \bmod N]$ is defined as $[(h \times g^{-1}) \bmod N]$). Finally, we remark that this does not imply that the group operation corresponds to the addition or multiplication operation for integers. Instead, this merely serves as useful general notation.

Example 4. A set may be formed as a group under one operation, but not another operation. For example, the set of integers \mathbb{Z} forms an abelian group under the addition operation: the identity element is “0,” and every integer g has inverse identity $-g$. On the other hand, it does not form a group under multiplication operation since, for example, the multiplicative inverse of the integer “0” does not make sense.

Example 5. The set of complex numbers \mathbb{C} is not a group under multiplication, since “0” does not have a multiplicative inverse. The set of nonzero complex numbers, however, is an abelian group under multiplication with identity “1.”

Example 6. Let $N \geq 2$ be an integer. The set $\{0, \dots, N - 1\}$ with respect to addition modulo N is an abelian group of order N : closure is obvious; associativity and commutativity follow from the fact that the integers satisfy these properties; the identity element is 0; and, since $a + (N - a) = 0 \pmod N$, it follows that the inverse element of any element a is $[(N - a) \pmod N]$. We denote this group by $(\mathbb{Z}_N, +)$. (Occasionally, the notion \mathbb{Z}_N will also be used to denote the set $\{0, \dots, N - 1\}$ without involving any particular operation.)

The “cancellation law” for groups has been demonstrated in the following lemma.

Lemma 2.1 Let (\mathbb{G}, \times) be a group and $a, b, c \in \mathbb{G}$. If $a \times c = b \times c$, then $a = b$. In particular, if $a \times c = c$, then a is the identity element in \mathbb{G} .

Proof. We know $a \times c = b \times c$. Multiplying both sides by the unique inverse element c^{-1} of the element c , we obtain $a = b$. In detail

$$a = a \times 1 = a \times (c \times c^{-1}) = (a \times c) \times c^{-1} = (b \times c) \times c^{-1} = b \times (c \times c^{-1}) = b \times 1 = b. \quad (2.4)$$

Group exponentiation: It is often useful to be able to describe the group operation applied n times to a fixed element g , where n is a positive integer. As for the additive operation, we demonstrate this as $n \cdot g$ or ng , that is,

$$ng = n \cdot g \stackrel{\text{def}}{=} \underbrace{g + \dots + g}_n. \quad (2.5)$$

Note that n is an integer, while g is a group element. So ng does not represent the group operation applied to n and g (indeed, we are working in a group where the group operation is written additively). However, the operation “behaves fortunately as it should,” for example, if $g \in \mathbb{G}$ and n, n' are integers, then $(ng) + (n'g) = (n + n')g$, $n(n'g) = (nn')g$ and $1 \cdot g = g$. In an abelian group \mathbb{G} with $g, h \in \mathbb{G}$, $(ng) + (nh) = n(g + h)$.

As for the multiplicative operation, we demonstrate application of the group operation n times to an element g by g^n . That is,

$$g^n \stackrel{\text{def}}{=} \underbrace{g \times \dots \times g}_n. \quad (2.6)$$

The familiar rules of exponentiation follow: $g^n \times g^{n'} = g^{n+n'}$, $(g^n)^{n'} = g^{nn'}$, and $g^1 = g$. Also, if \mathbb{G} is a commutative group and $g, h \in \mathbb{G}$, then $g^n \cdot h^n = (gh)^n$.

The above notation can be extended to the case when n is zero or a negative integer in the natural way. Generally, we leave g^r undefined if r is not an integer. As for additive operation, we have $0 \cdot g \stackrel{\text{def}}{=} 0$ and $(-n) \cdot g \stackrel{\text{def}}{=} n \cdot (-g)$ such that n is a positive integer. (Note that in the equation “ $0 \cdot g = 0$,” the “0” on the left-hand side is the integer 0, while the “0” on the right-hand side is the identity element in the group.) As one would expect, it can be shown that $(-n) \cdot g = -(ng)$. As for multiplicative notation, $g^0 \stackrel{\text{def}}{=} 1$ and $g^{-n} \stackrel{\text{def}}{=} (g^{-1})^n$. Again, as expected, one can show that $g^{-n} = (g^n)^{-1}$.

Theorem 2.1

If (\mathbb{G}, \times) is a finite group with the order $m = |\mathbb{G}|$, then for any element $g \in \mathbb{G}$, $g^m = 1$.

Proof. We prove the theorem only for the commutative group (\mathbb{G}, \times) (despite it holds for any finite group). Fix arbitrary $g \in \mathbb{G}$, and let g_1, \dots, g_m be the elements of \mathbb{G} . We claim that

$$g_1 \times g_2 \times \cdots \times g_m = (g \times g_1) \times (g \times g_2) \times \cdots \times (g \times g_m). \quad (2.7)$$

It is noted that $g \times g_i = g \times g_j$ implies $g_i = g_j$ according to the cancelation law shown in Lemma 2.1. Thus m elements in parentheses on the right-hand side of the displayed equation are pairwise different from each other. By considering that there are exactly m elements in \mathbb{G} , the m elements being multiplied together on the right-hand side are simply all elements of \mathbb{G} in some permuted order. The order in which all elements of the group are multiplied does not matter due to the fact that the group (\mathbb{G}, \times) is commutative, and thus the result of the right-hand side is identical to the result of left-hand side.

Fueled by the fact that (\mathbb{G}, \times) is commutative, all occurrences of the element g can be pulled out, and we can obtain

$$g_1 \times g_2 \times \cdots \times g_m = (g \times g_1) \times (g \times g_2) \times \cdots \times (g \times g_m) = g^m \times (g_1 \times g_2 \times \cdots \times g_m). \quad (2.8)$$

Once again by the cancelation law in the group, it is obvious that $g^m = 1$.

Corollary 2.1

If (\mathbb{G}, \times) is a finite group with the order $m = |\mathbb{G}| > 1$, then for any element $g \in \mathbb{G}$ and any integer i , $g^i = g^{[i \bmod m]}$.

Proof. According to Proposition 2.1, i can be expressed as $qm + r$, where q, r are integers and r can be demonstrated as $[i \bmod m]$. By Theorem 2.1,

$$g^i = g^{qm+r} = g^{qm} \times g^r = 1^q \times g^r = g^r \quad (2.9)$$

holds.

Example 7. As for the additive operation, the above corollary means that if g is an element in a group with order m , then $i \cdot g = [i \bmod m] \cdot g$. As an example, consider the group \mathbb{Z}_{25} of order $m = 25$, and take $g = 13$. The corollary can be instantiated as

$$303 \cdot 13 = [303 \bmod 25] \cdot 13 = 3 \cdot 13 = 13 + 13 + 13 = 39 = 14 \bmod 25. \quad (2.10)$$

Corollary 2.2

Let (\mathbb{G}, \times) be a finite group with order $m = |\mathbb{G}| > 1$. Let $e > 0$ be an integer, and define the function $f_e : \mathbb{G} \rightarrow \mathbb{G}$ by $f_e(g) = g^e$. If $\gcd(e, m) = 1$, then f_e is a permutation. Furthermore, if $d = [e^{-1} \bmod m]$, then f_d is the inverse of f_e .

Proof. According to Proposition 2.5, e is invertible modulo m due to the fact that $\gcd(e, m) = 1$. To show that f_d is the inverse of f_e , for any $g \in \mathbb{G}$, we have

$$f_d(f_e(g)) = f_d(g^e) = (g^e)^d = g^{ed} = g^{[ed \bmod m]} = g^1 = g. \quad (2.11)$$

2.1.2.2 Group (\mathbb{Z}_N^*, \times)

As mentioned before, the set $\mathbb{Z}_N = \{0, \dots, N - 1\}$ pairing with the addition operation modulo N can be regarded as a group. One natural problem is that whether the set $\{0, \dots, N - 1\}$ associated with the multiplication operation modulo N can be viewed as a group or not. It is obvious that “1” can be considered as the identity element for the multiplication modulo N . However, not every element in this set is invertible associated with the multiplication modulo N , for example, the multiplicative inverse element of “0” obviously does not make sense. What make matters worse, if $N = 8$, then the elements “2,” “4,” and “6” are not invertible by exhaustively trying every possible elements in $\{0, \dots, 7\}$. Thus, it is necessary to identify which elements in $\{0, \dots, N - 1\}$ are invertible modulo N . Depending on Proposition 2.5, the element $a \in \{0, \dots, N - 1\}$ is invertible if and only if $\gcd(a, N) = 1$. It is also

easy to observe that the inverse element of a resides in the range $\{0, \dots, N - 1\}$. This results in the definition of the following set for $N > 1$:

$$\mathbb{Z}_N^* \stackrel{\text{def}}{=} \{a \in \{1, \dots, N - 1\} \mid \gcd(a, N) = 1\}. \quad (2.12)$$

In other words, \mathbb{Z}_N^* includes integers in the set $\{1, \dots, N - 1\}$ that are relatively prime to N .

Based on the discussion above, the identity element and inverse element associated with each element can be found in \mathbb{Z}_N^* . Furthermore, the commutativity and associativity features inherit from \mathbb{Z}_N directly. To discuss the feature of closure, let $a, b \in \mathbb{Z}_N^*$ and $c = [a \times b \bmod N]$, and then assume $c \notin \mathbb{Z}_N^*$. This implies that $\gcd(c, N) \neq 1$, and so a prime p exists such that $p \mid N$ and $p \mid c$. From $c = [a \times b \bmod N]$, we see that $a \times b = qN + c$ for some integer q and thus $p \mid a \times b$. Based on Proposition 2.3, we obtain $p \mid a$ or $p \mid b$ and, thus, contradict either $\gcd(a, N) = 1$ or $\gcd(b, N) = 1$ (recall that $a, b \in \mathbb{Z}_N^*$). In short, the set \mathbb{Z}_N^* with respect to the multiplication operation modulo N forms a group.

Proposition 2.6

\mathbb{Z}_N^* is a commutative group pairing with the multiplication operation modulo N when $N > 1$ is an integer.

Let $\phi(N)$ denote the order of the group (\mathbb{Z}_N^*, \times) such that $\phi(N) \stackrel{\text{def}}{=} |\mathbb{Z}_N^*|$ and ϕ is called the *Euler phi function*. To compute the value of $\phi(N)$, we first consider the case when N is prime, i.e., $N = p$. In this case, all elements in $\{1, \dots, p - 1\}$ are relatively prime to p , and so $\phi(p) = |\mathbb{Z}_p^*| = p - 1$. We then consider the case $N = p \times q$ such that p, q are distinct primes. We see that either $p \mid a$ or $q \mid a$ if an integer $a \in \{1, \dots, N - 1\}$ is not relatively prime to N . Note that a cannot be divided by both p and q simultaneously since this would imply $pq \mid a$ and contradict $a < N (= p \times q)$. The elements in $\{1, \dots, N - 1\}$ that can be divided by p are exactly the $(q - 1)$ elements $p, 2p, 3p, \dots, (q - 1)p$, and the elements that can be divided by q are exactly the $(p - 1)$ elements $q, 2q, \dots, (p - 1)q$. Thus, the number of elements that are neither divisible by p or q is therefore given by

$$N - 1 - (q - 1) - (p - 1) = p \times q - p - q + 1 = (p - 1)(q - 1). \quad (2.13)$$

That is to say, $\phi(N) = (p - 1)(q - 1)$ when N is the product of two distinct primes p and q .

Theorem 2.2

Let $N = \prod_i p_i^{e_i}$, where the $\{p_i\}$ are distinct primes and $e_i \geq 1$. Then $\phi(N) = N \times \prod_i (1 - \frac{1}{p_i})$.

Example 8. Take $N = 24 = 3 \cdot 2^3$. Then $\mathbb{Z}_{24}^* = \{1, 5, 7, 11, 13, 17, 19, 23\}$ and $|\mathbb{Z}_N^*| = 8 = 24 \times (1 - \frac{1}{2}) \times (1 - \frac{1}{3}) = \phi(24)$. The inverse identity of 5 in \mathbb{Z}_N^* is 5 itself, since $5 \times 5 = 25 = 1 \bmod 24$.

Until now, the set \mathbb{Z}_N^* under the multiplication operation modulo N is shown to be a group with order $\phi(N)$. According to the Theorem 2.1, we get the following theorem directly:

Theorem 2.3

Take arbitrary $N > 1$ and $a \in \mathbb{Z}_N^*$; then we obtain

$$a^{\phi(N)} = 1 \bmod N. \quad (2.14)$$

For the specific case when $N = p$ is prime and $a \in \{1, \dots, p - 1\}$, we have

$$a^{p-1} = 1 \pmod{p}. \tag{2.15}$$

By Corollary 2.2, we obtain the following theorem easily:

Theorem 2.4

Let $N > 1$ be an integer. For integer $e > 0$, we define $f_e : \mathbb{Z}_N^* \rightarrow \mathbb{Z}_N^*$ as $f_e(x) = x^e \pmod{N}$. If e is relatively prime to $\phi(N)$, then f_e is a permutation. Moreover, if $d = [e^{-1} \pmod{\phi(N)}]$, then f_d is the inverse of f_e .

Definition 2.2. A function $f : \mathbb{G} \rightarrow \mathbb{H}$ is said to be an isomorphism from a group $(\mathbb{G}, \circ_{\mathbb{G}})$ to another group $(\mathbb{H}, \circ_{\mathbb{H}})$ if (1) f is a bijective mapping, and (2) for all $g_1, g_2 \in \mathbb{G}$, we have $f(g_1 \circ_{\mathbb{G}} g_2) = f(g_1) \circ_{\mathbb{H}} f(g_2)$. If these properties hold, then we say the group $(\mathbb{G}, \circ_{\mathbb{G}})$ and the group $(\mathbb{H}, \circ_{\mathbb{H}})$ are isomorphic and write this as $\mathbb{G} \cong \mathbb{H}$.

Group isomorphisms: An isomorphism from a group to another group provides an alternate and equivalent approach to think about the structure of groups. For example, if the group $(\mathbb{G}, \circ_{\mathbb{G}})$ is finite and $\mathbb{G} \cong \mathbb{H}$, then the group $(\mathbb{H}, \circ_{\mathbb{H}})$ must be finite and have the same order as \mathbb{G} . Also, if there exists an isomorphism f from a group $(\mathbb{G}, \circ_{\mathbb{G}})$ to another group $(\mathbb{H}, \circ_{\mathbb{H}})$, then f^{-1} is an isomorphism from $(\mathbb{H}, \circ_{\mathbb{H}})$ to $(\mathbb{G}, \circ_{\mathbb{G}})$.

Example 9. The bijective mapping f where $f(n) = 2n$ is an isomorphism from the group $(\mathbb{Z}, +)$ to the group $(\mathbb{Z}_{2n}, +)$ because $f(a + b) = 2(a + b) = 2a + 2b = f(a) + f(b)$, where \mathbb{Z}_{2n} denotes the set of even integers and $+$ is the addition operation for the integers.

Example 10. The bijective mapping f where $f(n) = 3^n$ is an isomorphism from the group $(\mathbb{Z}_6, +)$ to the group (\mathbb{Z}_7^*, \times) , where $+$ and \times denote the addition operation modulo 6 and the multiplication operation modulo 7. On the one hand, $3^0 = 1, 3^1 = 3, 3^2 = 2, 3^3 = 6, 3^4 = 4, \text{ and } 3^5 = 5$. On the other hand, $f(2 + 3) = 3^{(2+3)} = 3^2 \times 3^3 = f(2) \times f(3)$.

2.1.2.3 Chinese remainder theorem

Theorem 2.5

Let n_1, n_2, \dots, n_k be integers that are pairwise relatively prime, i.e., $\gcd(n_i, n_j) = 1$ for $i \neq j$. Then there exists a unique solution modulo of the product $n = n_1 \times n_2 \times \dots \times n_k$ to the following system of congruences:

$$\begin{aligned} x &\equiv a_1 \pmod{n_1} \\ x &\equiv a_2 \pmod{n_2} \\ &\vdots \\ x &\equiv a_k \pmod{n_k} \end{aligned} \tag{2.16}$$

The solution to the system of congruences shown in Theorem 2.5 can be calculated as

$$x = \sum_{i=1}^k a_i N_i M_i \pmod{n} \tag{2.17}$$

where

$$N_i = \frac{n}{n_i} \tag{2.18}$$

and

$$M_i = N_i^{-1} \pmod{n_i}. \tag{2.19}$$

The solution can also be represented in a slightly different way which makes it easier to be understood. Namely, we can rewrite Eq. (2.1) as

$$x = \sum_{i=1}^k a_i \cdot e_i \pmod n \quad (2.20)$$

such that

$$e_i \equiv \begin{cases} 1 & (\pmod{n_i}) \\ 0 & (\pmod{n_j}), j \neq i \end{cases} \quad (2.21)$$

So the solution of the Chinese remainder theory can be regarded essentially as an integer version of Lagrange interpolation, where a polynomial is created according to k points by calculating a similar set of coefficients that are either 0 or 1 and thus enforce the desired behavior at the given points.

Example 11. Consider the following system of congruences:

$$\begin{aligned} x &\equiv 2 \pmod{3} = a_1 \pmod{n_1} \\ x &\equiv 3 \pmod{5} = a_2 \pmod{n_2}. \\ x &\equiv 2 \pmod{7} = a_3 \pmod{n_3} \end{aligned} \quad (2.22)$$

According to the solution of the Chinese remainder theorem, we see that

$$\begin{aligned} n &= n_1 \times n_2 \times n_3 = 3 \times 5 \times 7 = 105 \\ N_1 &= \frac{n}{n_1} = \frac{105}{3} = 35 \\ N_2 &= \frac{n}{n_2} = \frac{105}{5} = 21 \\ N_3 &= \frac{n}{n_3} = \frac{105}{7} = 15 \\ M_1 &= N_1^{-1} \pmod{n_1} = 2^{-1} \pmod{3} = 2 \\ M_2 &= N_2^{-1} \pmod{n_2} = 3^{-1} \pmod{5} = 1 \\ M_3 &= N_3^{-1} \pmod{n_3} = 2^{-1} \pmod{7} = 1 \end{aligned} \quad (2.23)$$

so that

$$\begin{aligned} x &= (a_1 \times N_1 \times M_1 + a_2 \times N_2 \times M_2 + a_3 \times N_3 \times M_3) \pmod{105} \\ &= (2 \times 35 \times 2 + 3 \times 21 \times 1 + 2 \times 15 \times 1) \pmod{105} \\ &= 23 \pmod{105}. \end{aligned} \quad (2.24)$$

In this example we can also think of the solution as finding integers e_1 , e_2 , and e_3 such that

$$\begin{aligned} x &= (2 \times e_1 + 3 \times e_2 + 2 \times e_3) \pmod{105} \\ e_1 = 70 &\equiv \begin{cases} 70 & (\pmod{3}) \\ 0 & (\pmod{5}) \\ 0 & (\pmod{7}) \end{cases} \end{aligned} \quad (2.25)$$

$$e_2 = 21 \equiv \begin{cases} 0 & (\text{mod } 3) \\ 21 & (\text{mod } 5) \\ 0 & (\text{mod } 7) \end{cases}$$

and

$$e_3 = 15 \equiv \begin{cases} 0 & (\text{mod } 3) \\ 0 & (\text{mod } 5) \\ 15 & (\text{mod } 7) \end{cases}. \quad (2.26)$$

2.1.2.4 Cyclic groups and generators

Let (\mathbb{G}, \times) be a finite group with order m . For arbitrary $g \in \mathbb{G}$, consider the set

$$\langle g \rangle \stackrel{\text{def}}{=} \{g^0, g^1, \dots\}. \quad (2.27)$$

According to Theorem 2.1, $g^m = 1$. If $i \leq m$ denotes the smallest positive integer for which $g^i = 1$, then the above sequence repeats after i terms (i.e., $g^i = g^0$, $g^{i+1} = g^1$, ..., $g^{2i} = g^0$, etc.), and so

$$\langle g \rangle = \{g^0, \dots, g^{i-1}\}. \quad (2.28)$$

It is obvious to see that $\langle g \rangle$ contains exactly i elements since if $g^j = g^k$ with $0 \leq j < k < i$, then $g^{k-j} \neq 1$ thanks to the choice of i .

It is easy to observe that $(\langle g \rangle, \times)$ can be regarded as a subgroup of (\mathbb{G}, \times) , which is generated by the element g . If the order of the subgroup $(\langle g \rangle, \times)$ is i , then i is called the *order* of the element g .

Definition 2.3. If (\mathbb{G}, \times) is a finite group and g is randomly chosen from \mathbb{G} , then the order of element g is defined as the smallest positive integer i such that $g^i = 1$.

Proposition 2.7

If \mathbb{G} is a finite group, and $g \in \mathbb{G}$ is an element with order i , then for any integer x , we have $g^x = g^{[x \text{ mod } i]}$.

Proof. The proof of this proposition is similar to the proof of Corollary 2.1, and thus we omit it here.

Proposition 2.8

If \mathbb{G} is a finite group and $g \in \mathbb{G}$ is an element with order i , then $g^x = g^y$ if and only if $x = y \text{ mod } i$.

Proof. If $x = y \text{ mod } i$, then $[x \text{ mod } i] = [y \text{ mod } i]$, and according to the previous proposition, we directly have

$$g^x = g^{[x \text{ mod } i]} = g^{[y \text{ mod } i]} = g^y. \quad (2.29)$$

On the other hand, from $g^x = g^y$, we can easily obtain $g^{x'} = g^{y'}$ from the previous proposition, where $x' = [x \text{ mod } i]$ and $y' = [y \text{ mod } i]$. We in turn get $g^{x'}(g^{y'})^{-1} = g^{x'-y'} = 1$. If $x' \neq y'$, the difference $x' - y'$ is then a nonzero integer smaller than i since both x' and y' are smaller than i . There is contradiction between the fact that i is the order of the element g and the fact that $x' - y'$ is a nonzero integer smaller than i . Therefore, we obtain $x = y'$ directly.

The identity element of any group (\mathbb{G}, \circ) , features with the order 1, can generate the group $(\langle e \rangle, \circ)$. Furthermore, identity element is known as the only

element with order 1. On the other hand, if an element $g \in \mathbb{G}$ features with the order m (where m denotes the order of the group (\mathbb{G}, \circ)) can be found, then $\langle g \rangle = \mathbb{G}$. In other words, \mathbb{G} can be generated by the element g and considered as a cyclic group. Here, g is called a generator of \mathbb{G} . (Different from the identity element, a cyclic group may have multiple generators.) If g is a generator of cyclic group (\mathbb{G}, \circ) , then, by definition, every element $h \in \mathbb{G}$ can be derived by computing g^x for some $x \in \{0, \dots, m-1\}$.

Proposition 2.9

Let (\mathbb{G}, \circ) be a finite group with order m , and the element $g \in \mathbb{G}$ features with the order i . Then $i|m$.

Proof. By Theorem 2.1, $g^m = 1$. According to Proposition 2.7, we have $g^m = g^{[m \bmod i]}$ in case the element g features with the order i . If $i \nmid m$, then $i \stackrel{\text{def}}{=} [m \bmod i]$ is a nonzero integer smaller than i such that $g^i = 1$. This contradicts the fact that i is the order of the element g .

Corollary 2.3

If (\mathbb{G}, \circ) is a group with prime order p , then this group is cyclic. Furthermore, all elements of \mathbb{G} other than the identity element are generators of (\mathbb{G}, \circ) .

Proof. Based on Proposition 2.9, the only possible orders of elements in the group (\mathbb{G}, \circ) are 1 and p . Only the identity element features with the order 1, and so all other elements own order p and can generate the original group.

In addition to the groups of prime order, the additive group \mathbb{Z}_N , for $N > 1$, provides another example of a cyclic group, whereas the element 1 offers the function of generator.

Theorem 2.6

If p is prime, then $(\mathbb{Z}_p^{*}, \times)$ forms a cyclic group.

The proof of this theorem is outside the scope of this book, and the interesting reader can refer to for more details.

Example 12. Consider the group $(\mathbb{Z}_{11}^{*}, \times)$, which is cyclic by the previous theorem. We have $\langle 10 \rangle = \{1, 10\}$, and so 10 is not a generator. However,

$$\langle 2 \rangle = \{1, 2, 4, 8, 5, 10, 9, 7, 3, 6\} = \mathbb{Z}_{11}^{*} \tag{2.30}$$

and so 2 is a generator of \mathbb{Z}_{11}^{*} .

Given a cyclic group \mathbb{G} with order q , we can represent this group by a generator $g \in \mathbb{G}$ as $\mathbb{G} = \{g^0, g^1, \dots, g^{q-1}\}$. In other words, each element $h \in \mathbb{G}$ can be expressed as $h = g^x$ such that $x \in \mathbb{Z}_q$. Correspondingly, x is called the discrete logarithm of h with respect to the generator g such that $x = \log_g h$. Note that the logarithms in this case are regarded as “discrete” because these logarithm values range over a finite range, as opposed to “standard” logarithms from calculus whose values are within an infinite set.

The discrete logarithm problem in a cyclic group \mathbb{G} with respect to a given generator g is to calculate x such that $g^x = h$ with the input of a random element $h \in \mathbb{G}$. Formally speaking, \mathcal{G} is a polynomial-time algorithm with the input 1^n to output a cyclic group \mathbb{G} with order q and a generator $g \in \mathbb{G}$. In addition, the group operation in \mathbb{G} is required to be computed efficiently (i.e., in time polynomial in n). Consider the following experiment between a given algorithm \mathcal{A} and the algorithm \mathcal{G} :

The discrete logarithm experiment $DLog_{\mathcal{A}, \mathcal{G}}(n) \leftarrow$

1. Given the parameter n , the algorithm \mathcal{G} outputs (\mathbb{G}, q, g) , where \mathbb{G} denotes a cyclic group with order q and g is a generator of \mathbb{G} such that $||g|| = \#$.
2. Choose $h \in \mathbb{G}$ randomly.

3. Given $\{\mathbb{G}, q, g, h\}$, the task of \mathcal{A} is to output $x \in \mathbb{Z}_q$.

4. This experiment outputs 1 if $g^x = h$ and 0 otherwise.

Definition 2.4. The discrete logarithm problem is said to be hard relative to the algorithm \mathcal{G} if, for all probabilistic, polynomial-time algorithms \mathcal{A} , the probability $\Pr[DL\text{Log}_{\mathcal{A}, \mathcal{G}}(n) = 1]$ is negligible.

By way of example, groups of the form \mathbb{Z}_p^* , where p is prime number, offer one family of cyclic groups in which the discrete logarithm problem is believed to be hard [69, 70].

2.1.3 Elliptic curve groups

Different from \mathbb{Z}_p^* , another interesting class of groups, which is used widely in cryptographic applications, is those consisting of points on elliptic curves. Despite elliptic curve groups being very crucial in practical construction of cryptographic primitives, our treatment of such groups in this book is rather minimal and sacrifices generality in favor of simplicity. The basic reason about this approach is due to the fact that most cryptographic primitives based on elliptic curve groups can be understood and investigated by treating the underlying group in a completely generic manner without involving any particular group used to instantiate the primitive. Namely, cryptographic primitives can be built on arbitrary cyclic groups, and the security of these schemes can be proven as long as the related computational problem in the underlying group is believed to be hard no matter how the group is actually instantiated. Therefore, mathematical background needed for a deeper understanding of elliptic curve groups is beyond the scope of this book. Interested readers can refer to [71] for more details.

Define \mathbb{Z}_p as $\{0, \dots, p - 1\}$ for $p \geq 5$ be a prime. Despite the elliptic curves can in fact be defined over arbitrary (finite or infinite) fields, only the case such that $p \neq 2$ or 3 is taken into consideration to eliminate the additional complications. The elliptic curve E over \mathbb{Z}_p represents the set of points (x, y) defined by the equation $y^2 = x^3 + ax + b \pmod p$ with $a, b \in \mathbb{R}\mathbb{Z}_p$ and with the discriminant $4a^3 + 27b^2 \neq 0 \pmod p$ (this condition ensures that the equation $x^3 + ax + b = 0 \pmod p$ has no repeated roots). This set of points on \mathbb{Z}_p along with a distinguished point \mathcal{O} at infinity $\{(x, y) | x, y \in \mathbb{Z}_p \wedge E(x, y) = 0\} \cup \{\mathcal{O}\}$ outlines the curve E . The elements of the set $E(\mathbb{Z}_p)$ are called the points on the elliptic curve E defined by $y^2 = x^3 + ax + b \pmod p$, and \mathcal{O} is called the “point at infinity.”

Example 13. An element $y \in \mathbb{Z}_p^*$ is called a quadratic residue modulo p if there exists an $x \in \mathbb{Z}_p^*$ such that $x^2 = y \pmod p$, while x is regarded as a square root of y in this case. According to [68], every quadratic residue modulo p that has exactly two square roots for $p > 2$ is prime.

Let $f(x)$ be a function such that $f = x^3 + x + 6$ and consider the curve $E : y^2 = f(x) \pmod{11}$. Each value of x for which $f(x)$ is a quadratic residue modulo 11 yields two points on the curve; a value of x for which $f(x) = 0 \pmod{11}$ gives one point on the curve $E(\mathbb{Z}_{11})$. The points on the curve $E(\mathbb{Z}_{11})$ (shown in **Table 2.1**) can be identified and explained as follows:

- $f(0) = 6 \pmod{11}$, a quadratic nonresidue modulo 11.
- $f(1) = 8 \pmod{11}$, a quadratic nonresidue modulo 11.

x	0	1	2	3	4	5	6	7	8	9	10
$x^3 + x + 6 \pmod{11}$	6	8	5	3	8	4	8	4	9	7	4
Quadratic residue or nonresidue modulo 11?	×	×	✓	✓	×	✓	×	✓	✓	×	✓
y			4	5		2		2	3		2
			7	6		9		9	8		9

Table 2.1.

The points on the elliptic curve $y^2 \equiv x^3 + x + 6 \pmod{11}$.

- $f(2) = 5 \pmod{11}$, a quadratic residue modulo 11 with square roots 4 and 7. This yields the points $(2, 4), (2, 7) \in E(\mathbb{Z}_{11})$.
- $f(3) = 3 \pmod{11}$, a quadratic residue modulo 11 with square roots 5 and 6. This yields the points $(3, 5), (3, 6) \in E(\mathbb{Z}_{11})$.
- $f(4) = 8 \pmod{11}$, a quadratic nonresidue modulo 11.
- $f(5) = 4 \pmod{11}$, a quadratic residue modulo 11 with square roots 2 and 9. This yields the points $(5, 2), (5, 9) \in E(\mathbb{Z}_{11})$.
- $f(6) = 8 \pmod{11}$, a quadratic nonresidue modulo 11.
- $f(7) = 4 \pmod{11}$, a quadratic residue modulo 11 with square roots 2 and 9. This yields the points $(7, 2), (7, 9) \in E(\mathbb{Z}_{11})$.
- $f(8) = 9 \pmod{11}$, a quadratic residue modulo 11 with square roots 3 and 8. This yields the points $(8, 3), (8, 8) \in E(\mathbb{Z}_{11})$.
- $f(9) = 7 \pmod{11}$, a quadratic nonresidue modulo 11.
- $f(10) = 4 \pmod{11}$, a quadratic residue modulo 11 with square roots 2 and 9. This yields the points $(10, 2), (10, 9) \in E(\mathbb{Z}_{11})$.

Including the point \mathcal{O} at infinity, there are 7 points in $E(\mathbb{Z}_{11})$.

A common approach to conceptualize $E(\mathbb{Z}_p)$ intuitively is to observe the graph with equation $y^2 = x^3 + ax + b$ over the reals (rather than the equation $y^2 = x^3 + ax + b \pmod{p}$) as in **Figure 2.1**. Despite that this figure does not correspond exactly to $E(\mathbb{Z}_p)$ because $E(\mathbb{Z}_p)$ consists only a finite number of points (recall that \mathbb{Z}_p is a finite set), while there are an infinite number of solutions to the same equation over the real numbers, the picture offers insightful intuition. In such figure, the point at infinity \mathcal{O} can be viewed as sitting at the top of the y -axis and lying on every vertical line.

Observations from **Figure 2.2** demonstrate that every line intersecting with the elliptic curve $E(\mathbb{Z}_p)$ intersects this curve in exactly three points. In case the elliptic line is tangent to the curve at the point P , the point P is counted twice, and the point \mathcal{O} is also counted (when the line is vertical). A binary operation (called “addition”) can be defined from this fact and denoted by “+” on points of $E(\mathbb{Z}_p)$ as follows:

- The point \mathcal{O} is defined as the identity element such that $P + \mathcal{O} = \mathcal{O} + P = P$ for all $P \in E(\mathbb{Z}_p)$
- We obtain

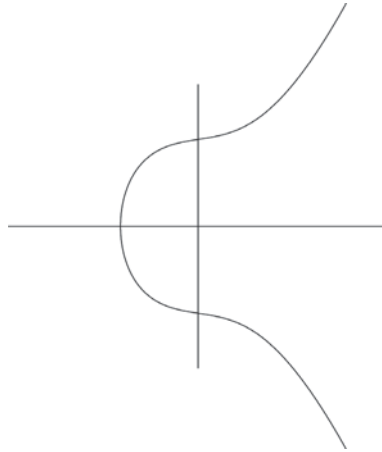


Figure 2.1.
 Graph of the elliptic curve $y^2 = x^3 + x + 6$ such that $\Delta > 0$.

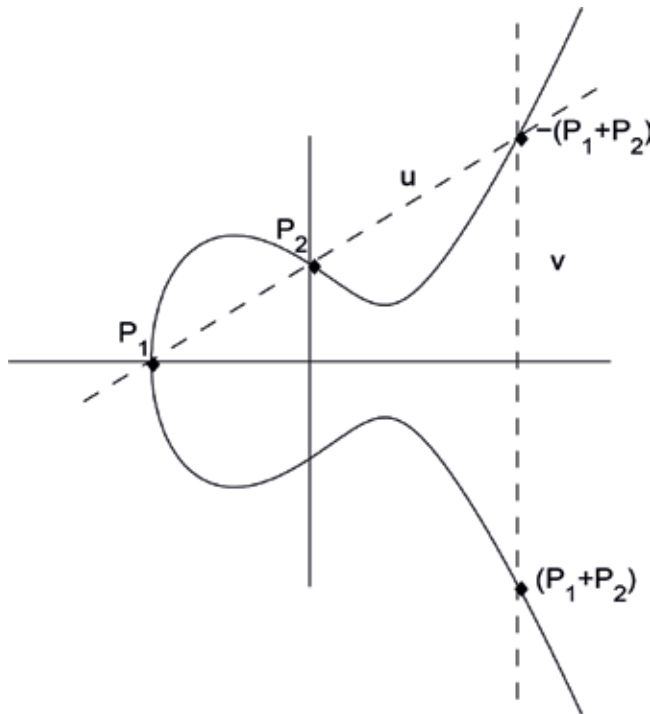


Figure 2.2.
 Addition operations on points on an elliptic curve.

$$P_1 + P_2 + P_3 = \mathcal{O}. \quad (2.31)$$

for P_1, P_2, P_3 are colinear points on E . (Note that the ordering of P_1, P_2, P_3 has been disregarded in the sense that the addition operation is commutative for all points and associative for colinear points.)

Negation. On input a point P , the negation of P is defined as the point $-P$ such that $P + (-P) = \mathcal{O}$. If $P = \mathcal{O}$, it is easy to see that $-P = \mathcal{O}$. Otherwise, due to the fact that $P + (-P) + \mathcal{O} = (P + (-P)) + \mathcal{O} = \mathcal{O} + \mathcal{O} = \mathcal{O}$ and Eq. (2.2), it is evident

that the negation of P corresponds to the third point on the line passing through P and \mathcal{O} or, in other words, the vertical line passing through P . According to **Figure 2.2**, $-P$ is simply the reflection of P in the x -axis, that is, if $P = (x, y)$, then $-P = -(x, y) = (x, -y)$.

Addition of points: To evaluate the sum $P_1 + P_2$ for two arbitrary points $P_1, P_2 \neq \mathcal{O}$ on the elliptic curve E , we can draw the line through P_1, P_2 (if $P_1 = P_2$ then draw the line tangent to E at P_1) and identify the third point of intersection P_3 of this line with the curve E . From Eq. (2.2), we can directly derive $P_1 + P_2 = -P_3$. If $P_3 = \mathcal{O}$, then $P_1 + P_2 = -\mathcal{O} = \mathcal{O}$. Otherwise, if the third point of intersection of the line through P_1 and P_2 is the point $P_3 = (x, y) \neq \mathcal{O}$, then

$$P_1 + P_2 = -P_3 = (x, -y). \quad (2.32)$$

It is straightforward but tedious to carry out the addition law concretely. Assume $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ are two points in $E(\mathbb{Z}_p)$ with $P_1, P_2 \neq \mathcal{O}$ and E is represented by the equation $y^2 = x^3 + ax + b \pmod{p}$. The slope of the line through these points can be computed as

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} \pmod{p}, & P_1 \neq P_2 \\ \frac{3x_1^2 + a}{2y_1} \pmod{p}, & P_1 = P_2 \end{cases}. \quad (2.33)$$

The line passing through P_1 and P_2 can be outlined by the equation

$$y = \lambda \cdot (x - x_1) + y_1 \pmod{p}. \quad (2.34)$$

To find the third point of intersection of this line with E , substitute Eq. (2.3) into $y^2 = x^3 + ax + b \pmod{p}$ to obtain

$$(\lambda \cdot (x - x_1) + y_1)^2 = x^3 + ax + b \pmod{p}. \quad (2.35)$$

The values of x that satisfy this equation are x_1, x_2 and $[\lambda^2 - x_1 - x_2 \pmod{p}]$. The first two solutions correspond to the original points P_1 and P_2 , while the third is the x -coordinate of the third point of intersection P_3 . The y -value corresponding to this third value of x is $y = [\lambda \cdot (x - x_1) + y_1 \pmod{p}]$. That is, $P_3 = (x_3, y_3)$ where

$$x_3 = (\lambda^2 - x_1 - x_2) \pmod{p} \quad y_3 = [\lambda \cdot (x_1 - x_3) - y_1] \pmod{p}. \quad (2.36)$$

It is straightforward that the set of points $E(\mathbb{Z}_p)$ pairing with the addition law defined above forms an abelian group. All the necessary properties have almost been shown: closure under addition law depends on the fact (the proof is omitted due to the space limit) that any line intersecting E intersects this curve in exactly three points, \mathcal{O} serves as the identity element, inverse element of each point on $E(\mathbb{Z}_p)$ can also be found in $E(\mathbb{Z}_p)$, and commutativity of addition law rests on Eq. (2.2). The final property to verify is associativity, which the interested reader can check themselves with tedious calculation.

According to **Table 2.1**, there are seven points in the set $E(\mathbb{Z}_{11})$ with the curve $E : y^2 = x^3 + x + 6 \pmod{11}$ such that

$$E(\mathbb{Z}_{11}) = \{\mathcal{O}, (2, 4), (2, 7), (3, 5), (3, 6), (5, 2), (5, 9), (7, 2), (7, 9), (8, 3), (8, 8), (10, 2), (10, 9)\}. \quad (2.37)$$

Suppose $P = (2, 7)$, $2P = P + P$ can be computed as follows:

First, the slope of the line can be calculated as follows:

$$\lambda = \frac{3 \times 2^2 + 1}{2 \times 7} \pmod{11} = \frac{2}{3} \pmod{11} = 8. \tag{2.38}$$

After that, we obtain

$$\begin{aligned} x_3 &= (8^2 - 2 - 2) \pmod{11} = 5 \\ y_3 &= [8 \times (2 - 5) - 7] \pmod{11} = 2. \end{aligned} \tag{2.39}$$

Therefore, $2P = (5, 2)$. Similarly, we can also get $3P = (8, 3)$, $4P = (10, 2)$, $5P = (3, 6)$, $6P = (7, 9)$, $7P = (7, 2)$, $8P = (3, 5)$, $9P = (10, 9)$, $10P = (8, 8)$, $11P = (5, 9)$, $12P = (2, 4)$, and $13P = \mathcal{O}$.

It is easy to see that $E(\mathbb{Z}_{11})$ along with the addition operation forms a commutative group with the generator $P = (2, 7)$.

Similar to \mathbb{Z}_p^* , elliptic curve groups provide another family of cyclic groups in which the discrete logarithm problem is believed to be hard [71].

2.1.4 Bilinear pairing

Let \mathbb{G} be a cyclic multiplicative group with the generator g and the prime order p and \mathbb{G}_T be another multiplicative cyclic group of the same order p . A bilinear pairing refers to a map $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ featured with the following properties:

1. Bilinearity: For all $g_1, g_2 \in \mathbb{G}$, and $a, b \in \mathbb{Z}_p^*$, $\hat{e}(g_1^a, g_2^b) = \hat{e}(g_1, g_2)^{ab}$.
2. Nondegeneracy: There exist $g_1, g_2 \in \mathbb{G}$ such that $\hat{e}(g_1, g_2) = 1_{\mathbb{G}_T}$.
3. Computability: For all $g_1, g_2 \in \mathbb{G}$, there is an efficient algorithm to compute $\hat{e}(g_1, g_2)$.

The modified Weil pairing [77] or Tate pairing [72] defined on the elliptic curves can be adopted to implement such an admissible bilinear map.

2.2. Public-key cryptography

The discussion thus far has been very general. We now show some concrete examples of concrete public-key encryption and signature schemes in the traditional PKI and ID-PKC settings, respectively. As mentioned in [73], public-key encryption enables two parties to communicate with each other securely without any shared secret information in advance. Let us call the sender Alice and the receiver Bob. Bob first selects the key pair (pk_B, sk_B) and sends the public key pk_B to Alice over any channel but keeps the private key sk_B secure and secret. Before sending the sensitive message m to Bob, Alice generates the ciphertext c by performing the encryption algorithm $c = Enc_{pk_B}(m)$ under Bob's public key. To recover the message m , Bob decrypts the ciphertext c by carrying out the inverse decryption $Dec_{sk_B}(c)$ with his own private key sk_B . According to [74], digital signature allows a user to generate a signature on the given message with secret key sk in such a way that any other party who knows this user's corresponding public key pk can ensure that the message indeed originated from this user and has not been altered in any way. Let us call the sender Alice and the receiver Bob. The sender

Alice first generates a public/secret key pair (pk_A, sk_A) and then distributes pk_A in some reliable way to the other users in the system while keeping sk_A secret. When the message m originated from Alice needs to be authenticated, Alice can generate a digital signature σ on the message m using her private key sk_A ; the pair (m, σ) will be then sent. Anyone who knows the public key pk_A can verify the authenticity of m by checking whether $\text{Vrfy}_{pk}(m, \sigma) \stackrel{?}{=} 1$ or not. In this section, we have chosen to focus on the most well-known and long-standing public-key primitives in this section.

Observing that all existing constructions of public-key encryption and signature schemes are proven secure under some assumptions associated with the hardness of solving some mathematical problems. Therefore, the number-theoretic hard problems related to these constructions have been first reviewed here.

Definition 2.5. (RSA problem) Let $N = p \times q$, where p and q are two distinct odd prime numbers. Let e be a random prime number such that $\text{gcd}(e, (p-1) \times (q-1)) = 1$, and let \mathbb{G} be a cyclic subgroup of \mathbb{Z}_N^* . Given (N, e) and $y \in \mathbb{G}$, the RSA problem consists of finding $z \in \mathbb{Z}_N$ such that $y = z^e$.

Definition 2.6. (Discrete logarithm (DL) problem) Let \mathbb{G} be a finite cyclic group and g be a generator of \mathbb{G} . Given a random element $h \in \mathbb{G}$, the DL problem consists of computing the exponent x such that $h = g^x$.

Definition 2.7. (Weil Diffie-Hellman (WDH) problem) Given a generator P of \mathbb{G} and a triple $\langle aP, bP, cP \rangle$ for random $a, b, c \in \mathbb{Z}_p^*$, the WDH problem is to compute $\hat{e}(P, P)^{abc} \in \mathbb{G}_T$.

Definition 2.8. (Decision Diffie-Hellman (DDH) problem) Given a generator P of \mathbb{G} and a tuple $\langle aP, bP, cP \rangle$ for $a, b, c \in \mathbb{Z}_p^*$, the DDH problem is to decide whether $c \stackrel{?}{=} ab \pmod q$ holds or not.

Definition 2.9. (Computational Diffie-Hellman (CDH) problem) Given a generator P of \mathbb{G} and a tuple $\langle aP, bP \rangle$ for $a, b \in \mathbb{Z}_p^*$, the CDH problem is to compute abP .

2.2.1 Public-key encryption algorithms

2.2.1.1 Framework of public-key encryption

Definition 2.10. A public-key encryption scheme (see **Figure 2.3**) consists of the following three algorithms:

1. Gen: Given a security parameter 1^n , this algorithm outputs a pair of public and secret keys (pk, sk) .
2. Enc: On input a public key pk and a message m from some underlying plaintext space, this algorithm outputs a ciphertext c , which can be denoted as $c = \text{Enc}_{pk}(m)$.
3. Dec: On input a private key sk and a ciphertext c , this algorithm outputs a message m or a special symbol \perp to denote failure, which can be written as $m = \text{Dec}_{sk}(c)$.

We make the consistency constraint that if for every n , every (pk, sk) generated by $\text{Gen}(1^n)$, every message m in the appropriate underlying plaintext space, and $c = \text{Enc}_{pk}(m)$, then $\text{Dec}_{sk}(c) = m$.

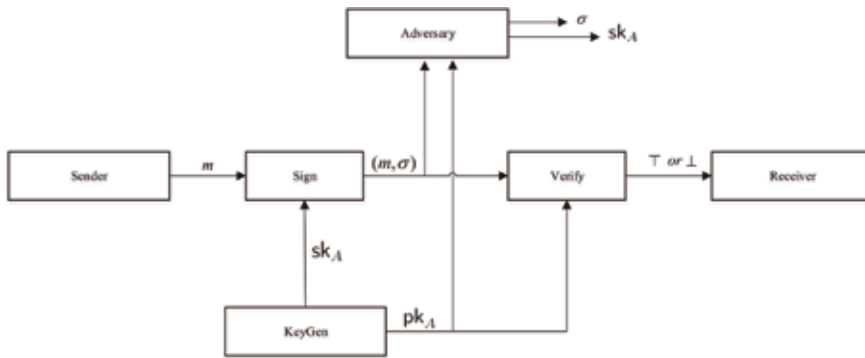


Figure 2.3.
 Intuition of public-key encryption.

2.2.1.2 RSA encryption scheme

The RSA encryption scheme [75] is the most widely used public-key encryption scheme, and its security depends on the intractability of the RSA problem. Concretely, the RSA encryption scheme consists of the following three algorithms:

Gen: To create the public key and the corresponding secret key, the following steps are performed:

1. Randomly choose two prime numbers p and q such that $|p| \approx |q|$.
2. Compute $N = p \times q$.
3. Compute $\phi(N) = (p - 1) \times (q - 1)$.
4. Randomly choose an integer $e < \phi(N)$ such that $\gcd(e, \phi(N)) = 1$, and compute the integer d such that $e \times d \equiv 1 \pmod{\phi(N)}$.
5. Publish (N, e) as the public key, and keep d as the corresponding private key. Note that p, q and $\phi(N)$ will be destroyed safely.

Enc: Before sending a sensitive message $m < N$, the sender creates the ciphertext $c = m^e \pmod{N}$. (Note that the plaintext message space is in fact \mathbb{Z}_N^* .)

Dec: To decrypt the ciphertext c , the corresponding receiver computes $m = c^d \pmod{N}$.

2.2.1.3 ElGamal encryption scheme

The ElGamal encryption scheme [76] is constructed based on the DL problem. Concretely, the ElGamal encryption scheme consists of the following three algorithms:

Gen: To create the public key and the corresponding secret key, the following steps are performed:

1. Choose a random multiplicative generator g of \mathbb{Z}_p^* such that p is a random prime number.
2. Pick a random number $1 \leq x \leq p - 2$ as the private key.

3. Compute the corresponding public key by $y = g^x \pmod p$.
4. Publish (p, g, y) as the public key, and keep x as the corresponding private key.

Enc: Before sending a sensitive message $m \in \mathbb{Z}_p^*$, the sender picks $1 \leq k \leq p - 2$ and computes the ciphertext as follows:

$$c_1 = g^k \pmod p, c_2 = y^k \pmod p. \quad (2.40)$$

Dec: To decrypt the ciphertext (c_1, c_2) , the following steps are performed:

$$m = c_2 / c_1^x \pmod p. \quad (2.41)$$

2.2.1.4 Framework of ID-based encryption scheme

Definition 2.11. An ID-based encryption scheme (see **Figure 2.4**) consists of the following three algorithms:

Setup: On input a security parameter k , this algorithm is performed by the PKG to create the system parameters params and the master secret key master-key such that params will be distributed to all users in the system and master-key will be kept secret by PKG itself.

Extract: On input params , master-key , and an arbitrary identity $ID \in \{0, 1\}^*$, this algorithm is performed by the PKG to generate a private key d_{ID} . After that, d_{ID} will be sent to the corresponding user secretly.

Enc: On input params , ID , and m , this algorithm outputs a ciphertext c .

Dec: On input params , ID , c , and the corresponding private key d_{ID} , this algorithm recovers m .

2.2.1.5 Boneh-Franklin IBE

The first practical identity-based encryption scheme is constructed by Boneh and Franklin [77] based on bilinear pairings, and its security depends on the WDH problem. First the basic Boneh-Franklin IBE scheme which is not secure against an adaptive chosen-ciphertext attack¹ is presented. The only reason for describing the basic scheme is to improve the readability of the full scheme. After that, the full scheme extends the basic scheme to achieve security against an adaptive chosen-ciphertext attack.

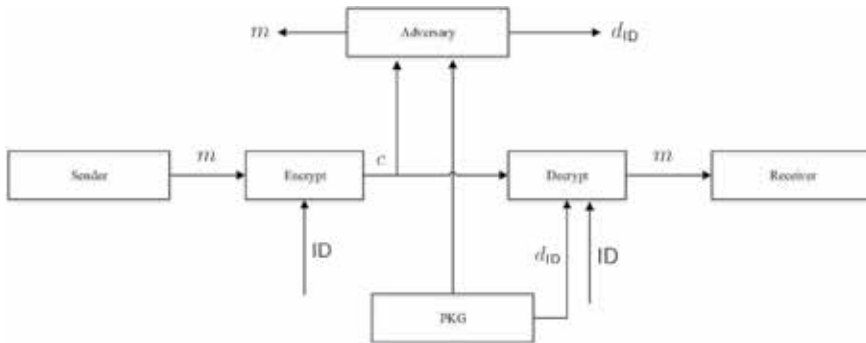


Figure 2.4. Intuition of ID-based encryption.

¹ The detail of adaptive chosen-ciphertext attack can be found in Section 2.3.

Boneh-Franklin IBE (basic scheme): Concretely, the basic Boneh-Franklin IBE scheme consists of the following four algorithms:

Setup: The algorithm works as follows:

1. Let $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a symmetric bilinear pairing defined in Section 2.1.4.
2. Pick a random $s \in \mathbb{Z}_q^*$ and set $P_{pub} = sP$.
3. Choose two cryptographic hash functions $H_1: \{0, 1\}^* \rightarrow \mathbb{G}^*$ and $H_2 : \mathbb{G}_T \rightarrow \{0, 1\}^n$.

The system parameters $params = \{P, P_{pub}, H_1, H_2\}$ are published to everyone in the system, while master-key $s \in \mathbb{Z}_q^*$ is kept secret by PKG.

Extract: On input a string $ID \in \{0, 1\}^*$, this algorithm is performed by the PKG as follows:

1. Compute $d_{ID} = s \cdot H_1(ID)$ as the private key associated with the identity ID .
2. Send d_{ID} to the user ID secretly.

Encrypt: To encrypt m under the public-key ID , r is chosen from \mathbb{Z}_q^* , and the ciphertext is generated as follows:

$$c = \langle rP, m \oplus H_2(g_{ID}^r) \rangle \text{ where } g_{ID} = \hat{e}(H_1(ID), P_{pub}) \in \mathbb{G}_T. \quad (2.42)$$

Decrypt: Given a ciphertext $c = \langle U, V \rangle$ encrypted using the public-key ID . The plaintext message can be recovered as follows:

$$V \oplus H_2(\hat{e}(d_{ID}, U)) = m. \quad (2.43)$$

Boneh-Franklin IBE (full scheme): Concretely, the full Boneh-Franklin IBE scheme consists of the following four algorithms:

Setup and Extract: The algorithms are the same as those in the above basic scheme. In addition, two additional hash function $H_3: \{0, 1\}^n \times \{0, 1\}^n \rightarrow \mathbb{Z}_q^*$ and $H_4: \{0, 1\}^n \rightarrow \{0, 1\}^n$ are required.

Encrypt: To encrypt m under the public-key ID , choose a random $\gamma \in \{0, 1\}^n$, set $r = H_3(\gamma, m)$, and generate the ciphertext as follows:

$$C = \langle rP, \gamma \oplus H_2(g_{ID}^r), m \oplus H_4(\gamma) \rangle \text{ where } g_{ID} = \hat{e}(H_1(ID), P_{pub}). \quad (2.44)$$

Decrypt: Let $\langle U, V, W \rangle$ be a ciphertext encrypted using the public-key ID . Given a ciphertext $c = \langle U, V, W \rangle$ encrypted using the public-key ID . The plaintext message can be recovered as follows:

1. Compute $V \oplus H_2(\hat{e}(d_{ID}, U)) = \gamma$.
2. Compute $W \oplus H_4(\gamma) = m$.
3. Set $r = H_3(\gamma, m)$. Test that $U = rP$. If not, reject the ciphertext.
4. Output m .

2.2.2 Signature algorithms

2.2.2.1 Framework of digital signature

Definition 2.12. A signature scheme (see **Figure 2.5**) consists of the following three algorithms:

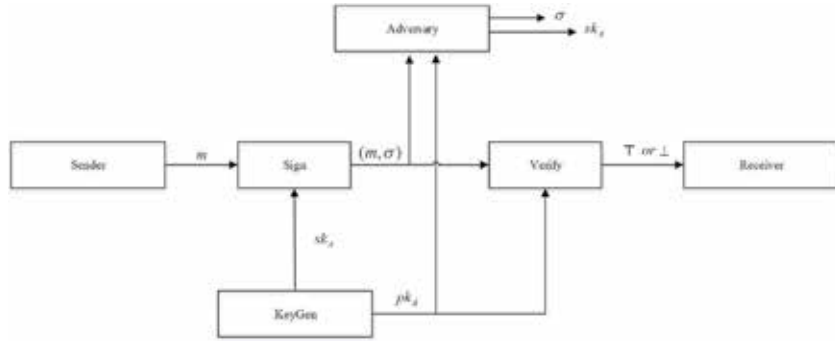


Figure 2.5.
Intuition of digital signature.

Gen: Given a security parameter 1^n , this algorithm outputs a pair of public and secret keys (pk, sk) .

Sign: On input a private key sk and a message m from some underlying message space, this algorithm outputs a signature σ , which can be written as $\sigma = \text{Sign}_{sk}(m)$.

Verify: On input a public key pk , a message m , and a signature σ , this algorithm returns 1 for accept or 0 for reject.

We make the consistency constraint that if for every n , every (pk, sk) generated by Gen, and every message m , it holds that

$$\text{Verify}_{pk}(m, \text{Sign}_{sk}(m)) = 1. \quad (2.45)$$

2.2.2.2 RSA signature scheme

The RSA cryptosystems [75] may be used to provide both encryption and digital signature. The RSA digital signature is described as follows:

Gen: To create the public key and the corresponding secret key, the following steps are performed:

1. Randomly choose two prime numbers p and q such that $|p| \approx |q|$.
2. Compute $N = p \times q$.
3. Compute $\phi(N) = (p - 1) \times (q - 1)$.
4. Randomly choose an integer $e < \phi(N)$ such that $\text{gcd}(e, \phi(N)) = 1$, and compute the integer d such that $e \times d \equiv 1 \pmod{\phi(N)}$.
5. Publish (N, e) as the public key, and keep d as the corresponding private key. Noted that p, q and $\phi(N)$ will be destroyed safely.

Sign: To create a signature of message $m \in \mathbb{Z}_N^*$, the signer creates $\sigma = m^d \pmod{N}$.

Verify: Given a message-signature pair (m, s) , this algorithm outputs 1 if $m = s^e \pmod{N}$ and returns 0 otherwise.

2.2.2.3 ElGamal signature scheme

The ElGamal cryptosystems [75] may be used to provide both encryption and digital signature. The ElGamal digital signature is described as follows:

Gen: To create the public key and the corresponding secret key, the following steps are performed:

1. Choose a random multiplicative generator g of \mathbb{Z}_p^* such that p is a random prime number.
2. Pick a random number $1 \leq x \leq p - 2$ as the private key.
3. Compute the corresponding public key by $y = g^x \pmod{p}$.
4. Publish (p, g, y) as the public key, and keep x as the corresponding private key.

Sign: To create a signature of message $m \in \mathbb{Z}_p^*$, the signer picks a random number $1 \leq k \leq p - 2$ and generates a signature (r, s) such that

$$r = g^k \pmod{p}, s = k^{-1}(m - x \times r) \pmod{p - 1}. \quad (2.46)$$

Verify: Given a message-signature pair $(m, (r, s))$, this algorithm outputs 1 if $y^r \times r^s = g^m \pmod{p}$ and returns 0 otherwise.

2.2.2.4 Schnorr signature scheme

Many variations of the basic ElGamal signature scheme have been proposed, and Schnorr signature scheme [78] is one well-known variant of the ElGamal scheme. The ElGamal digital signature is described as follows:

Gen: The system parameters are generated as follows:

1. Choose two prime numbers p and q such that $q|p - 1$.
2. Choose an element $g \in \mathbb{Z}_p^*$ of order q .
3. Select a cryptographic hash function $H: \{0, 1\}^* \rightarrow \mathbb{Z}_q$.
4. Select a random number $x \in {}_R\mathbb{Z}_q$ as the secret key and compute $y = g^{-x} \pmod{p}$ as the public key. Publish (p, q, g, y, H) as the public key and keep x as the corresponding secret key.

The parameters (p, q, g, H) are publicized for use by system-wide users.

Sign: To create a signature on message $m \in \{0, 1\}^*$, the signer picks a random number k from \mathbb{Z}_q at random and computes a signature such that $r = g^k \pmod{p}$, $e = H(m||r)$, and $s = k + x \times e \pmod{q}$.

Verify: Given a message-signature pair $(m, (e, s))$, this algorithm outputs 1 if $r^s = g^s \times y^e \pmod{p}$ and $e' = H(m||r')$ and returns 0 otherwise.

2.2.2.5 Digital signature standard

In 1991, a digital signature algorithm (DSA) has been presented by the US National Institute of Standards and Technology (NIST) and later been called the Digital Signature Standard (DSS) [79] by the US Federal Information Processing Standards (FIPS 186). DSS is regarded as the first digital signature scheme recognized by the US government and is described as follows:

Gen: The system parameters are generated as follows:

1. Select a prime number q such that $2^{159} < q < 2^{160}$.
2. Choose t such that $0 \leq t \leq 8$, and select a prime number p where $2^{511+64t} < p < 2^{512+64t}$ and $q|(p-1)$.
3. Select a generator g of the unique cyclic group with order q in \mathbb{Z}_p^* .
 - a. Select an element $h \in \mathbb{Z}_p^*$ and compute $g = h^{(p-1)/q} \bmod p$.
 - b. If $g = 1$ then go to previous step.
4. Select a random integer x such that $1 \leq x \leq q-1$ and compute $y = g^x \bmod p$.
5. (p, q, g, y) is published as the public key, while x is kept secret as the private key of the corresponding user.

Sign: To create a signature on message $m \in \{0, 1\}^*$, the following steps are performed.

1. Select a random secret integer k such that $0 < k < q$.
2. Compute $r = (g^k \bmod p) \bmod q$.
3. Compute $k^{-1} \bmod q$.
4. Compute $s = k^{-1}\{h(m) + xr\} \bmod q$.
5. (r, s) is returned as the signature on the message m .

Verify: Given a message-signature pair $(m, (r, s))$, the following steps are performed to check the validity of the signature.

1. Compute $u_1 = s^{-1} \cdot h(m) \bmod q$ and $u_2 = r \cdot s^{-1} \bmod q$.
2. Output 1 if $r = (g^{u_1} y^{u_2} \bmod p) \bmod q$ and return 0 otherwise.

2.2.2.6 Framework of ID-based signature scheme

An ID-based encryption scheme (see **Figure 2.6**) consists of the following three algorithms:

Setup: On input a security parameter k , this algorithm is performed by the PKG to create the system parameters params and the master secret key master-key such that params will be distributed to all users in the system and master-key will be kept secret by PKG itself.

Extract: On input params , master-key , and an arbitrary identity $ID \in \{0, 1\}^*$, this algorithm is performed by the PKG to generate a private key d_{ID} . After that, d_{ID} will be sent to the corresponding user secretly.

Sign: Given a message m , an identity ID , a private key d_{ID} , and params , this algorithm generates the signature σ on the message m under the identity ID .

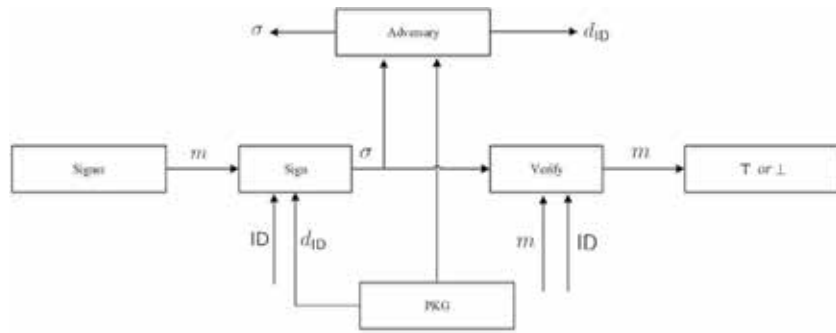


Figure 2.6.
 Intuition of ID-based signature.

Verify: Given a signature σ , a message m , an identity ID , and params, this algorithm returns 1 for accept or 0 for reject.

2.2.2.7 Cha-Cheon IBS

Inspired by [77], an ID-based signature scheme has been proposed by Cha and Cheon [80] using gap Diffie-Hellman (GDH) groups, where GDH group is a cyclic group where the CDH problem is hard, but the DDH problem is easy. The Cha-Cheon identity-based signature scheme is described as follows:

Setup: Let $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a symmetric bilinear pairing defined in Section 2.1.4. Choose a generator P from the group \mathbb{G} , pick a random $s \in \mathbb{Z}_p^*$, set $P_{pub} = sP$, and choose cryptographic hash functions $H_1: \{0, 1\}^* \times G \rightarrow \mathbb{Z}_p^*$ and $H_2: \{0, 1\}^* \rightarrow \mathbb{G}$. The system parameter (P, P_{pub}, H_1, H_2) is published, while the master secret key s is kept secret by PKG.

Extract: Given an identity ID , this algorithm computes $d_{ID} = sH_2(ID)$ and sends it to the user associated with identity ID secretly.

Sign: Given a secret key d_{ID} and a message m , pick a random number $r \in \mathbb{Z}_p^*$ and output a signature $\sigma = (U, V)$ such that $U = rH_2(ID)$, $h = H_1(m, U)$, and $V = (r + h)d_{ID}$.

Verify: To verify a signature $\sigma = (U, V)$ on a message m under an identity ID , this algorithm outputs 1 if $\hat{e}(U + hH_2(ID), P_{pub}) = \hat{e}(P, V)$ and returns 0 otherwise, where $h = H_1(m, U)$.

2.2.2.8 Bellare-Neven-Namprempre IBS

Bellare et al. [81] proposed the first pairing-free identity-based signature scheme based on the DL problems. The Bellare-Neven-Namprempre identity-based signature scheme is described as follows:

Setup: Generate an elliptic curve E over a finite field \mathbb{Z}_p , a prime q dividing the number of points on E , and a curve point P of order q , pick a random $x \in \mathbb{Z}_q^*$, and compute $P_{pub} = xP$. Choose two cryptographic hash functions $H_1, H_2: \{0, 1\}^* \rightarrow \mathbb{Z}_p$. $\langle \mathbb{Z}_p, E, q, P, P_{pub}, H_1, H_2 \rangle$ is published as the system parameter, and x is kept secret as the master secret key by PKG.

Extract: Given an identity ID , PKG picks a random $r \in \mathbb{Z}_q^*$ and computes $R = rP$, $c = H_1(ID, R)$, and $s = r + cx \text{ mod } q$. (R, s) is sent to the user associated with ID secretly.

Sign: Given a secret key pair (R, s) associated with ID and a message m , pick a random $t \in \mathbb{Z}_q^*$ and compute $T = tP$, $h = H_2(ID, m, R, T)$, and

$$z = t + hs \pmod{q}. \quad (2.47)$$

The resulting signature is $\sigma = (R, T, z)$.

Verify: To verify a signature $\sigma = (R, T, z)$ of a message m for an identity ID, compute $h = H_2(ID, m, R, T)$ and $c = H_1(ID, R)$ and check whether

$$zP = T + h(R + cP_{pub}) \quad (2.48)$$

holds or not.

2.3. Provable security

Provable security, which involves the exact definitions, precise assumptions, and rigorous security proof of cryptographic primitives, is regarded as the methodology unavoidable in designing, analyzing, and evaluating new primitives [82–84]. The basic reason for the necessity of provable security originates from the fact that the security of a cryptographic primitive cannot be checked in the same way that software is typically checked. Without a proof that no adversary with enough computational resources can break the primitive, we can only depend on our intuition about the security of the mentioned primitive. Unfortunately, history has already demonstrated that intuition in cryptography and information security is disastrous in the sense that countless examples of unproven schemes or schemes only with heuristic security proof were broken (sometimes immediately or sometimes years after being published or deployed).

Instead of indiscreetly assuming a given cryptographic primitive is secure, provable security assumes that some mathematical problems are hard to solve and then to prove that the given primitive is secure provided this assumption. Concretely, the proof proceeds by presenting an explicit reduction showing how to construct an efficient algorithm \mathcal{C} that succeeds in solving certain computational tasks that was assumed to be hard from any efficient adversary \mathcal{A} that succeeds in breaking the primitive with nonnegligible probability. The steps of such a proof can be outlined in a high level as follows:

We first assume that some mathematical problem X cannot be solved by any polynomial-time algorithm except with negligible probability. To prove that some cryptographic primitive Π is secure, the following steps are performed.

Initial: Assume an efficient adversary \mathcal{A} attempts to attack the cryptographic primitive Π and denotes \mathcal{A} 's success probability by ε .

1. An efficient adversary \mathcal{C} is constructed by employing adversary \mathcal{A} as a subroutine to solve problem X . Given some input instance of mathematical problem X , the algorithm \mathcal{C} will simulate an execution of Π for the adversary \mathcal{A} such that:
 - a. The view of \mathcal{A} when it is invoked as a subroutine by \mathcal{C} should be identical to the view of \mathcal{A} when it interacts with the real primitive Π itself.
 - b. Furthermore, if \mathcal{A} succeeds in breaking the execution of Π which is being simulated by \mathcal{C} , this should enable \mathcal{C} to solve the given instance of X with inverse polynomial probability $\frac{1}{p}$.

2. From Step 1, it is obvious that if ϵ is not negligible, then the problem X can be solved by \mathcal{C} with nonnegligible probability $\frac{\epsilon}{p}$, which contradicts the initial assumption. Therefore, we conclude that, given the assumption associated with X , no efficient algorithm \mathcal{A} can succeed with nonnegligible probability ϵ ; in other words, Π is proven to be computationally secure formally.

2.3.1 Public-key encryption

As for the security model of public-key encryption, we begin our definitional treatment by considering the case of an eavesdropping adversary who observes a single ciphertext that it wishes to crack. In other words, the eavesdropping adversary does not have any further interaction with the sender or the receiver after receiving a (single) ciphertext. Intuitively speaking, the security model for the public-key encryption is depicted by the definition of indistinguishability. Particularly, consider an experiment $\text{PubK}_{\mathcal{A}, \Pi}^{\text{eav}}(n)$ between an eavesdropping adversary \mathcal{A} and a corresponding simulator/challenger \mathcal{C} , in which \mathcal{A} outputs two messages m_0 and m_1 with equal length and in turn is given an encryption of one of these messages randomly chosen by \mathcal{C} using a randomly generated key. According to the definition of indistinguishability, a public-key encryption scheme Π is said to be secure if, in this experiment, no adversary \mathcal{A} can distinguish which message is encrypted better than a naive guess. We now give the formal definition of $\text{PubK}_{\mathcal{A}, \Pi}^{\text{eav}}(n)$ as follows:

Given a public-key encryption scheme $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ and an adversary \mathcal{A} , the eavesdropping experiment $\text{PubK}_{\mathcal{A}, \Pi}^{\text{eav}}(n)$ is defined as follows:

1. The algorithm $\text{Gen}(1^n)$ is performed by \mathcal{C} to calculate keys (pk, sk) .
2. After receiving the public key pk \mathcal{A} outputs a pair of messages (m_0, m_1) such that $|m_0| = |m_1|$.
3. After receiving (m_0, m_1) , \mathcal{C} chooses a random bit $b \in \{0, 1\}$ and sends to \mathcal{A} a ciphertext $c = \text{Enc}_{pk}(m_b)$. Here, c is called the challenge ciphertext.
4. After receiving c , \mathcal{A} outputs a bit b' and wins the experiment if $b' = b$.

Definition 2.13. A public-key encryption scheme $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ is regarded to achieve indistinguishability against an eavesdropper if for all probabilistic, polynomial-time adversaries \mathcal{A} , the advantage of \mathcal{A} in the eavesdropping experiment $\text{PubK}_{\mathcal{A}, \Pi}^{\text{eav}}(n)$, defined as $\text{Adv}_{\Pi}^{\text{ind-eav}}(\mathcal{A}) = |2\Pr[b' = b] - 1|$, is negligible.

2.3.1.1 Security against chosen-plaintext attacks

In view of the fact that a relatively weak attacker who only passively eavesdrops on the communication has been modeled in the eavesdropping experiment, a more powerful type of adversarial attacker who is allowed to query encryptions of multiple messages in an adaptive manner should be considered to simulate the attacker's capabilities in the real world. Thus, a chosen-plaintext attack (CPA) has been defined to enable the adversary \mathcal{A} to interact freely with an encryption oracle to encrypt messages of \mathcal{A} 's choice.

Specifically, consider the following experiment defined for public-key encryption scheme $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$; the CPA indistinguishability experiment $\text{PubK}_{\mathcal{A}, \Pi}^{\text{cpa}}(n)$ between the adversary \mathcal{A} who can mount chosen-plaintext attack and a corresponding simulator/challenger \mathcal{C} is defined as follows:

Initial: The algorithm $\text{Gen}(1^n)$ is performed by \mathcal{C} to calculate keys (pk, sk) .

Phase 1: After receiving the public key pk , \mathcal{A} issues a sequence of queries to the oracle $\text{Enc}_{pk}(\cdot)$ adaptively.

Challenge: After deciding Phase 1 is over, \mathcal{A} outputs a pair of messages $(m_0, m_1) \leftarrow$ with equal length. \mathcal{C} now chooses a random bit $b \in \{0, 1\}$ and sends to \mathcal{A} a ciphertext $c \leftarrow \text{Enc}_{pk}(m_b)$. Here, c is called the challenge ciphertext.

Phase 2: The adversary \mathcal{A} now issues a second sequence of queries to the oracle $\text{Enc}_{pk}(\cdot)$ adaptively again as in Phase 1.

Guess: After receiving c , \mathcal{A} outputs a bit b' and wins the experiment if $b' = b$.

Definition 2.14. A public-key encryption scheme $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ is regarded to achieve indistinguishability against chosen-plaintext attacks if for all probabilistic, polynomial-time adversaries \mathcal{A} , the advantage of \mathcal{A} in the experiment $\text{PubK}_{\mathcal{A}, \Pi}^{\text{cpa}}(n)$, defined as $\text{Adv}_{\Pi}^{\text{ind-cpa}}(\mathcal{A}) = |2\Pr[b' = b] - \frac{1}{2}|$, is negligible.

2.3.1.2 Security against chosen-ciphertext attacks

To strengthen the adversary's capabilities further, a third type of adversary, which is more powerful than the former two types of adversaries and can mount chosen-ciphertext attack [85, 86], should be taken into consideration. In this attack model, the adversary is not only allowed to encrypt any messages of its choice as in the chosen-plaintext attack model but also enabled the adversary to decrypt any ciphertexts of its choice. Therefore, the chosen-ciphertext attack is regarded as the most powerful attack associated to the public-key encryption so far.

Specifically, consider the following experiment defined for public-key encryption scheme $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$; the CCA indistinguishability experiment $\text{PubK}_{\mathcal{A}, \Pi}^{\text{cca}}(n) \leftarrow$ (shown in **Figure 2.7**) between the adversary \mathcal{A} who can mount chosen-ciphertext attack and a corresponding simulator/challenger \mathcal{C} is defined as follows:

Initial: The algorithm $\text{Gen}(1^n)$ is performed by \mathcal{C} to calculate keys (pk, sk) .

Phase 1: After receiving the public key pk , \mathcal{A} issues a sequence of queries to the encryption oracle $\text{Enc}_{pk}(\cdot)$ and the decryption oracle $\text{Dec}_{sk}(\cdot)$ adaptively.

Challenge: After deciding Phase 1 is over, \mathcal{A} outputs a pair of messages $(m_0, m_1) \leftarrow$ with equal length. \mathcal{C} now chooses a random bit $b \in \{0, 1\}$ and sends to \mathcal{A} a ciphertext $c \leftarrow \text{Enc}_{pk}(m_b)$. Here, c is called the challenge ciphertext.

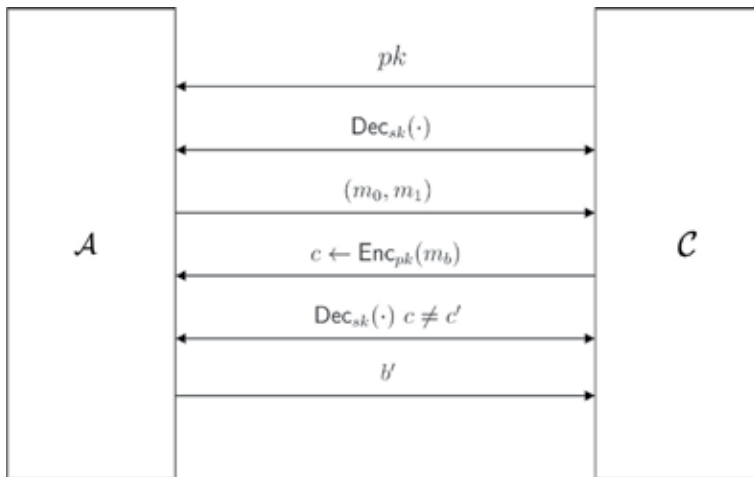


Figure 2.7. CCA experiment $\text{PubK}_{\mathcal{A}, \Pi}^{\text{cca}}(n)$ for public-key encryption.

Phase 2: After receiving c , \mathcal{A} now issues a second sequence of queries to the oracles $\text{Enc}_{pk}(\cdot)$ and $\text{Dec}_{sk}(\cdot)$ adaptively again as in Phase 1 with the restriction that the challenged ciphertext cannot be queried to the decryption oracle.

Guess: \mathcal{A} outputs a bit b' and wins the experiment if $b' = b$.

Definition 2.15. A public-key encryption scheme $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ is regarded to achieve indistinguishability against chosen-ciphertext attacks if for all probabilistic, polynomial-time adversaries \mathcal{A} , the advantage of \mathcal{A} in the experiment $\text{PubK}_{\mathcal{A}, \Pi}^{\text{cca}}(n)$, defined as $\text{Adv}_{\Pi}^{\text{ind-cca}}(\mathcal{A}) = |2\Pr[b' = b] - 1|$, is negligible.

2.3.2 ID-based encryption

Different from the public-key encryption in the traditional PKI, the definition of chosen-ciphertext security must be strengthened a bit in the ID-PKC due to the fact that the attacker might already own the private keys of users ID_1, \dots, ID_n of her choice when she attacks another public key ID . To ensure the security of encryption scheme in the ID-based environment [77], the definition of chosen-ciphertext security must enable the attacker to obtain the private key associated with any identity of her choice (other than the challenged public key ID being attacked).

2.3.2.1 Security against chosen-ciphertext and identity attacks

Specifically, consider the following experiment defined for ID-based encryption scheme $\Pi = (\text{Setup}, \text{Extract}, \text{Encrypt}, \text{Decrypt})$; the CCA indistinguishability experiment $\text{IBE}_{\mathcal{A}, \Pi}^{\text{cca}, \text{cida}}(n)$ (shown in **Figure 2.8**) between the adversary \mathcal{A} who can mount chosen-ciphertext attack and a corresponding simulator/challenger \mathcal{C} is defined as follows:

Initial: The algorithm Setup is performed by the challenger \mathcal{C} to generate the system parameters params , which will be forwarded to the adversary \mathcal{A} .

Phase 1: \mathcal{A} can perform a polynomially bounded number of queries in an adaptive manner as follows:

1. Upon receiving an identity ID_i , \mathcal{C} runs private key extraction oracle $\text{Extract}(\cdot)$ on ID_i and forwards the associated private key to \mathcal{A} .

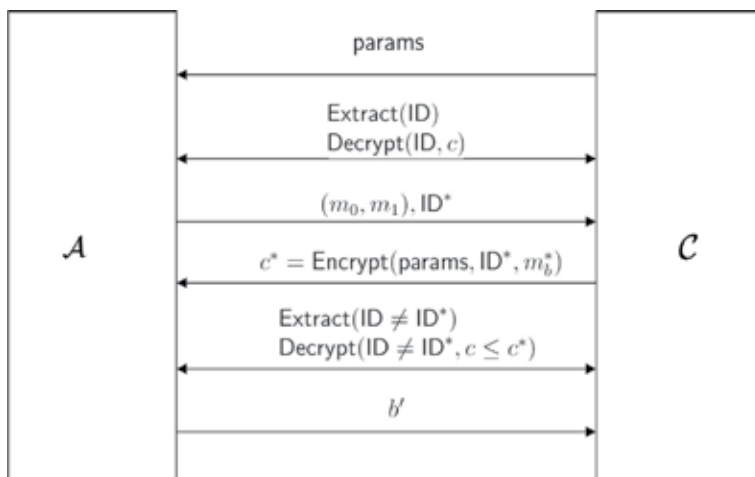


Figure 2.8. CCA experiment $\text{IBE}_{\mathcal{A}, \Pi}^{\text{cca}, \text{cida}}(n)$ for ID-based encryption.

2. Upon receiving a tuple (ID_i, c_i) , \mathcal{C} runs decryption oracle $Decrypt(\cdot)$ on (ID_i, c_i) and sends the result to \mathcal{A} .

Challenge: After deciding Phase 1 is over, adversary submits two plaintexts (m_0, m_1) with equal length and an identity ID^* to \mathcal{C} such that the identity ID^* must not have sent to the private key extraction oracle $Extract(\cdot)$ in Phase 1. After receiving (m_0, m_1) , \mathcal{C} selects a random bit $b \in \{0, 1\}$, sets $c = Encrypt(params, ID^*, m_b)$, and forwards c to the adversary as the challenge ciphertext.

Phase 2: This is identical to Phase 1 with the restriction that:

1. \mathcal{A} must not have sent ID^* to the private key extraction oracle $Extract(\cdot)$.
2. \mathcal{A} must not have sent (ID^*, c) to the decryption oracle $Decrypt(\cdot)$.

Guess: \mathcal{A} outputs a bit b' and wins the experiment if $b' = b$.

Definition 2.16. An ID-based encryption scheme $\Pi = (Setup, Extract, Encrypt, Decrypt)$ is regarded to achieve indistinguishability against chosen-ciphertext attacks if for all probabilistic, polynomial-time adversaries \mathcal{A} , the advantage of \mathcal{A} in the experiment $IBE_{\mathcal{A}, \Pi}^{cca, cida}(n)$, defined as $Adv_{\Pi}^{ind-cca-ibe}(\mathcal{A}) = |2Pr[b' = b] - 1|$, is negligible.

2.3.3 Digital signature

To capture the adversary's power in the digital signature, a security property called existential unforgeability against a chosen message attack [87] should be achieved in the sense that the existential unforgeability means that the adversary should not be able to generate a valid signature on any message, and the chosen message attack means that the adversary is able to obtain signatures on any messages it wishes during its attack. We now give the formal definition of existential unforgeability against a chosen message attack as follows:

2.3.3.1 Security against chosen message attacks

Let $\Pi = (Gen, Sign, Verify)$ be a signature scheme, and consider the following experiment $Sig - forge_{\mathcal{A}, \Pi}^{cma}(n)$ (shown in **Figure 2.9**) between an adversary \mathcal{A} and a corresponding challenger/simulator \mathcal{C} as follows:

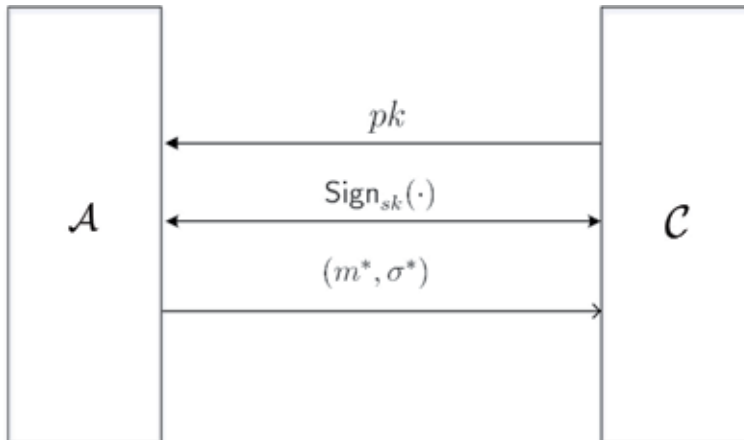


Figure 2.9. Unforgeability experiment $Sig - forge_{\mathcal{A}, \Pi}^{cma}(n)$ for signature.

Initial: The algorithm Gen is run by \mathcal{C} to obtain keys (pk, sk) such that the public key pk is forwarded to the adversary \mathcal{A} , whereas the private key sk is kept secret by \mathcal{C} itself.

Attack: After receiving the public key pk , \mathcal{A} is given to access the signing oracle $\text{Sign}_{sk}(\cdot)$ which returns a signature $\text{Sign}_{sk}(m)$ for any message m of \mathcal{A} 's choice.

Forgery: \mathcal{A} outputs a message and signature pair (m^*, σ^*) . \mathcal{A} wins this experiment if $\text{Verify}_{pk}(m^*, \sigma^*) = 1$ with the restriction that σ^* has not been queried to the oracle $\text{Sign}_{sk}(\cdot)$.

Definition 2.17. A signature scheme $\Pi = (\text{Gen}, \text{Sign}, \text{Vrfy})$ is said to achieve existentially unforgeability under an adaptive chosen message attack if for all probabilistic polynomial-time adversaries \mathcal{A} , the advantage of \mathcal{A} wins the experiment $\text{Sig} - \text{forge}_{\mathcal{A}, \Pi}^{cma}(n)$ is negligible.

2.3.4 ID-based signature

The security model of existential unforgeability under an adaptive chosen message attack in the traditional PKI can be extended to the identity-based environment naturally [80, 81]. We now give the formal definition of existential unforgeability against a chosen message attack for the ID-based signature as follows:

2.3.4.1 Security against chosen message and identity attacks

Let $\Pi = (\text{Setup}, \text{Extract}, \text{Sign}, \text{Verify})$ be a signature scheme, and consider the following experiment $\text{IBS} - \text{forge}_{\mathcal{A}, \Pi}^{cma, cida}(n)$ (shown in **Figure 2.10**) between an adversary \mathcal{A} and a corresponding challenger/simulator \mathcal{C} as follows:

Initial: The algorithm Setup is performed by the challenger \mathcal{C} to generate the system parameters params , which will be forwarded to the adversary \mathcal{A} .

Attack: \mathcal{A} can perform a polynomially bounded number of queries in an adaptive manner as follows:

- Upon receiving an identity ID_i , \mathcal{C} runs private key extraction oracle $\text{Extract}(\cdot)$ on ID_i and forwards the associated private key to \mathcal{A} .

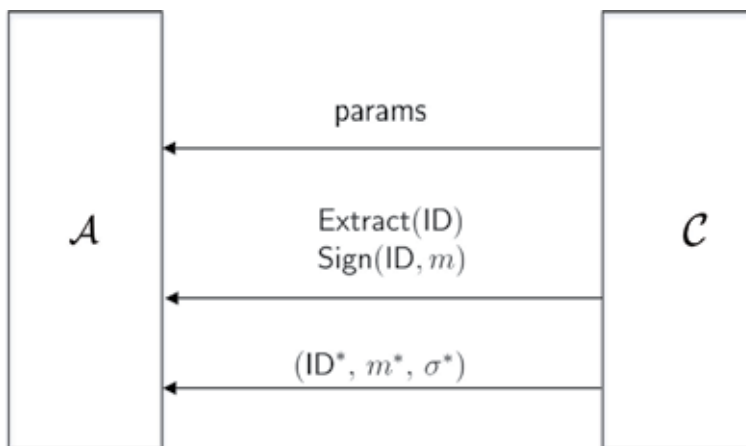


Figure 2.10. Unforgeability experiment $\text{IBS} - \text{forge}_{\mathcal{A}, \Pi}^{cma, cida}(n)$ for ID-based signature.

- Upon receiving a tuple (ID, m) , \mathcal{C} runs signing oracle $Sign(\cdot)$ on (ID, m) and forwards the result to \mathcal{A} .

Forgery: \mathcal{A} outputs a message m^* , an identity ID^* , and a signature σ^* . \mathcal{A} is said to succeed in the experiment $IBS - forge_{\mathcal{A}, \Pi}^{cma, cida}(n)$ if the following requirements are satisfied:

1. $Verify(\text{params}, ID^*, m^*, \sigma^*) = 1$.
2. ID^* has not been queried to the private key extraction oracle $Extract(\cdot)$.
3. (ID^*, m^*) has not been queried to the signing oracle $Sign(\cdot)$.

Definition 2.18. An ID-based signature scheme $\Pi = (Setup, Extract, Sign, Verify)$ is said to achieve existentially unforgeability under an adaptive chosen message attack if for all probabilistic polynomial-time adversaries \mathcal{A} , the advantage of \mathcal{A} wins the experiment $IBS - forge_{\mathcal{A}, \Pi}^{cma, cida}(n)$ is negligible.

Section 2

Outsourcing for Data
Services in Mobile Internet

Information Retrieval

3.1. Introduction

Information retrieval is a very simple concept with everyone having practical experience in its use. The scenario of a user having an information need, translating that into a search statement and executing that search to locate the information, has become ubiquitous to everyday life. The Internet has become a repository of any information a person needs, replacing the library as a more convenient research tool. An information retrieval system is a system that ingests information, transforms it into searchable format, and provides an interface to allow a user to search and retrieve information. The most obvious example of an information retrieval system is Google, and the English language has even been extended with the term “Google it” to mean search for something.

So, everyone has had an experience with information retrieval systems, and with a little thought, it is easy to answer the question: does it work? Everyone who has used such systems has experienced the frustration that is encountered when looking for certain information. Given the massive amount of intellectual effort that is going into the design and evolution of a “Google” or other search systems, the question comes to mind why is it so hard to find what you are looking for.

One of the goals of this chapter is to explain the practical and theoretical issues associated with information retrieval that makes the design of information retrieval systems one of the challenges of our time. The demand for and expectations of users to quickly find any information they need continues to drive both the theoretical analysis and development of new technologies to satisfy that need. To scope the problem, one of the first things that need to be defined is “information.” Twenty-five years ago, information retrieval was totally focused on textual items. That was because almost all of the “digital information” of value was in textual form. In today’s technical environment, most people carry with them most of the time the capability to create images and videos of interest, that is, the cell phone. This has made modalities other than text to become as common as text. That is coupled with the Internet Web sites that allow and are designed for ease of the use of uploading and storing those modalities which more than justify the need to include other than text as part of the information retrieval problem. There are a lot of parallelisms between the information processing steps for text, images, audio, and video. Although maps are another modality that could be included, they will only be generally discussed.

In general, information that will be considered in information retrieval systems includes text, images, audio, and video. The term “item” shall be used to define a specific information object. This could be a textual document, a news item from an RSS feed, an image, a video program, or an audio program. It is useful to make a distinction between the original items from what is processed by the information retrieval system as the basic indexable item. The original item will always be kept for display purposes, but a lot of preprocessing can occur on it during the process of creating the searchable index. The term “item” will refer to the original object. On occasion the term document will be used when the item being referred to is a textual item.

An information retrieval system is the hardware and software that facilitates a user in finding the information the user needs. Hardware is included in the

definition because specialized hardware is needed to transform certain modalities into digital processing format (e.g., encoders that translate composite video to digital video). As the detailed processing of items is described, it will become clear that an information retrieval system is not a single application but is composed of many different applications that work together to provide the tools and functions needed to assist the users in answering their questions. The overall goal of an information retrieval system is to minimize the user overhead in locating the information of value. Overhead from a user's perspective can be defined as the time it takes to locate the needed information. The time starts when a user starts to interact with the system and ends when they have found the items of interest. Human factors play significantly in this process. For example, most users have a short threshold on frustration waiting for a response. That means in a commercial system on the Internet, the user is more satisfied with a response less than 3 s than a longer response that has more accurate information. In internal corporate systems, users are willing to wait a little longer to get results, but there is still a trade-off between accuracy and speed. Most users would rather have the faster results and iterate on their searches than allowing the system to process the queries with more complex techniques providing better results. All of the major processing steps are described for an information retrieval system, but in many cases, only a subset of them are used on operational systems because users are not willing to accept the increase in response time.

The evolution of information retrieval systems has been closely tied to the evolution of computer processing power. Early information retrieval systems were focused on automating the manual indexing processes in libraries. These systems migrated the structure and organization of card catalogs into structured databases. They maintained the same Boolean search query structure associated with the database that was used for other database applications. This was feasible because all of the assignment of terms to describe the content of a document was done by professional indexers. In parallel, there was also academic research work being done on small data sets that considered how to automate the indexing process making all of the text of a document part of the searchable index. The only place that large systems designed to search on massive amounts of text were available was in government and military systems. As commercial processing power and storage significantly increased, it became more feasible to consider applying the algorithms and techniques being developed in the universities to commercial systems. In addition, the creation of the original documents also was migrating to digital format so that they were in a format that could be processed by the new algorithms. The largest change that drove information technologies to become part of everyone's experience was the introduction and growth of the Internet. The Internet became a massive repository of unstructured information, and information retrieval techniques were the only approach to effectively locate information on it. This changed the funding and development of search techniques from a few government-funded efforts to thousands of new ideas being funded by venture capitalists moving the more practical implementation of university algorithms into commercial systems.

3.2. Objectives of information retrieval system

The general objective of an information retrieval system is to minimize the time it takes for a user to locate the information they need. The goal is to provide the information needed to satisfy the user's question. Satisfaction does not necessarily mean finding all information on a particular issue. It means finding sufficient information that the user can proceed with whatever activity initiated. This is very important because it does explain some of the drivers behind the existing search

systems and suggests that precision is typically more important than recalling all possible information. For example, a user looking for a particular product does not have to find the names of everyone that sells the product or every company that manufactures the product to meet their need of getting that product. Of course, if they did have total information, then it is possible that they could have gotten it cheaper, but in most cases, the consumer will never know what they missed. The concept that a user does not know how much information they missed explains why in most cases the precision of a search is more important than the ability to recall all possible items of interest; the user never knows what they missed, but they can tell if they are seeing a lot of useless information in the first few pages of search results. That does not mean that finding everything on a topic is not important to some users. If you are trying to make decisions on purchasing a stock or a company, then finding all the facts about that stock or company may be critical to prevent a bad investment. Missing the one article talking about the company being sued and possibly going bankrupt could lead to a very painful investment. But providing comprehensive retrieval of all items that are relevant to a user's search can have the negative effect of information overload on the user. In particular there is a tendency for important information to be repeated in many items on the same topic. Thus, trying to get all information makes the process of reviewing and filtering out redundant information very tedious. The better a system is in finding all items on a question (Recall), the more important techniques to present aggregates of that information become.

From the user's perspective, time is the important factor that they use to gage the effectiveness of information retrieval. Except for users that do information retrieval as a primary aspect of their job (e.g., librarians and research assistants), most users have very little patience for investing extensive time in finding information they need. They expect interactive response from their searches with replies within 3 C4s at the most. Instead of looking through all the hits to see what might be of value, they will only review the first one, and at most second pages before deciding, they need to change their search strategy. These aspects of the human nature of searchers have had a direct effect on the commercial Web sites and the development of commercial information retrieval. The times that are candidates to be minimized in an information retrieval system are the time to create the query, the time to execute the query, the time to select what items returned from the query the user wants to review in detail, and the time to determine if the returned item is of value. The initial research in information retrieval focused on the search as the primary area of interest. But to meet the user's expectation of fast response and to maximize the relevant information returned require optimization in all of these areas. The time to create a query used to be considered outside the scope of technical system support. But systems such as Google know what is in their database and what other users have searched on; so as you type a query, they provide hints on what to search on. This vocabulary browse capability helps the user in expanding the search string and helps in getting better precision.

In information retrieval, the term "relevant" is used to represent an item containing the needed information. In reality, the definition of relevance is not a binary classification but a continuous function. Items can exactly match the information need or partially match the information need. From a user's perspective, "relevant" and needed are synonymous. From a system perspective, information could be relevant to a search statement (i.e., matching the criteria of the search statement) even though it is not needed/relevant to user (e.g., the user already knew the information or just read it in the previous item reviewed).

Relevant documents are those that contain some information that helps answer the user's information need. Nonrelevant documents do not contain any useful

information. Using these definitions, the two primary metrics used in evaluating information retrieval systems can be defined. They are Precision and Recall:

$$\begin{aligned} \text{Precision} &= \frac{\text{Number_Retrieved_Relevant}}{\text{Number_Total_Retrieved}} \\ \text{Recall} &= \frac{\text{Number_Retrieved_Relevant}}{\text{Number_Possible_Relevant}} \end{aligned} \quad (3.1)$$

Number_Possible_Relevant is the number of relevant items in the database, *Number_Total_Retrieved* is the total number of items retrieved from the query, and *Number_Retrieved_Relevant* is the number of items retrieved that are relevant to the user's search need.

Precision is the factor that most users understand. When a user executes a search and has 80% precision, it means that four out of five items that are retrieved are of interest to the user. From a user's perspective, the lower the precision the more likely the user is wasting his resource (time) looking at nonrelevant items. From a metric perspective, the precision figure is across all of the "hits" returned from the query. But in reality most users will only look at the first few pages of hit results before deciding to change their query strategy. Thus, what is of more value in commercial systems is not the total precision but the precision across the first 20C50 hits. Typically, in a weighted system where the words within a document are assigned weights based upon how well they describe the semantics of the document, precision in the first 20C50 items is higher than the precision across all the possible hits returned. But when comparing search systems, the total precision is used.

Recall is a very useful concept in comparing systems. It measures how well a search system is capable of retrieving all possible hits that exist in the database. Unfortunately, it is impossible to calculate except in very controlled environments. It requires in the denominator the total number of relevant items in the database. If the system could determine that number, then the system could return them. There have been some attempts to estimate the total relevant items in a database, but there are no techniques that provide accurate enough results to be used for a specific search request. In Chapter 3.3 on "Information Retrieval Evaluation," techniques that have been used in evaluating the accuracy of different search systems will be described. But it is not applicable in the general case.

3.3. Classic information retrieval

In this section we briefly present the three classic models in information retrieval, namely, the Boolean, the vector, and the probabilistic models.

3.3.1 Basic concepts

The classic models in information retrieval consider that each document is described by a set of representative keywords called index terms. An index term is simply a (document) word whose semantics helps in remembering the document's main themes. Thus, index terms are used to index and summarize the document contents. In general, index terms are mainly nouns because nouns have meaning by themselves and, thus, their semantics is easier to identify and to grasp. Adjectives, adverbs, and connectives are less useful as index terms because they work mainly as complements. However, it might be interesting to consider all the distinct words in a document collection as index terms. We postpone a discussion on the problem of how to generate index terms, where the issue is covered in detail.

Given a set of index terms for a document, we notice that not all terms are equally useful for describing the document contents. In fact, there are index terms which are simply vaguer than others. Deciding on the importance of a term for summarizing the contents of a document is not a trivial issue. Despite this difficulty, there are properties of an index term which are easily measured and which are useful for evaluating the potential of a term as such. For instance, consider a collection with a hundred thousand documents. A word which appears in each of the 100,000 documents is completely useless as an index term because it does not tell us anything about which documents the user might be interested in. On the other hand, a word which appears in just five documents is quite useful because it narrows down considerably the space of documents which might be of interest to the user. Thus, it should be clear that distinct index terms have varying relevance when used to describe document contents. This effect is captured through the assignment of numerical weights to each index term of a document.

Let k_i be an index term, d_j be a document, and $w_{i,j} \geq 0$ be a weight associated with the pair (k_i, d_j) . This weight quantifies the importance of the index term for describing the document semantic contents.

Definition 3.1. Let t be the number of index terms in the system and k_i be a generic index term. $K = k_1, \dots, k_t$ is the set of all index terms. A weight $w_{i,j} > 0$ is associated with each index term k_i of a document d_j . For an index term which does not appear in the document text, $w_{i,j} = 0$. With the document d_j is associated an index term vector \vec{d}_j represented by $\vec{d}_j = (w_{1,j}, w_{2,j}, \dots, w_{t,j})$. Further, let g_i be a function that returns the weight associated with the index term k_i in any t -dimensional vector (i.e., $g_i(\vec{d}_j) = w_{i,j}$).

As we later discuss, the index term weights are usually assumed to be mutually independent. This means that knowing the weight $w_{i,j}$ associated with the pair (k_i, d_j) tells us nothing about the weight $w_{i+1,j}$ associated with the pair (k_{i+1}, d_j) . This is clearly a simplification because occurrences of index terms in a document are not uncorrelated. Consider, for instance, that the terms computer and network are used to index a given document which covers the area of computer networks. Frequently, in this document, the appearance of one of these two words attracts the appearance of the other. Thus, these two words are correlated and their weights could reflect this correlation. While mutual independence seems to be a strong simplification, it does simplify the task of computing index term weights and allows for fast ranking computation. Furthermore, taking advantage of index term correlations for improving the final document ranking is not a simple task. In fact, none of the many approaches proposed in the past has clearly demonstrated that index term correlations are advantageous (for ranking purposes) with general collections. Therefore, unless clearly stated otherwise, we assume mutual independence among index terms. In Chapter 5, we discuss modern retrieval techniques which are based on term correlations and which have been tested successfully with particular collections. These successes seem to be slowly shifting the current understanding toward a more favorable view of the usefulness of term correlations for information retrieval systems.

The above definitions provide support for discussing the three classic information retrieval models, namely, the Boolean, the vector, and the probabilistic models, as we now do.

3.3.2 Boolean model

The Boolean model is a simple retrieval model based on set theory and Boolean algebra. Since the concept of a set is quite intuitive, the Boolean model provides a

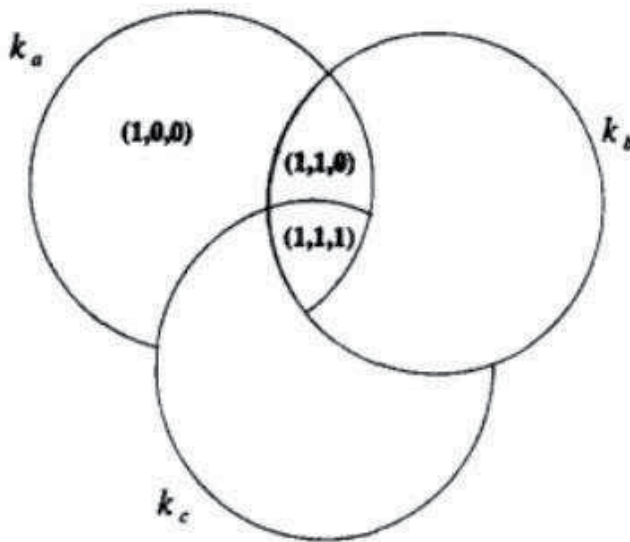


Figure 3.1.
The three conjunctive components for the query $[q = k_a \wedge (k_b \vee \neg k_c)]$.

framework which is easy to grasp by a common user of an IR system. Furthermore, the queries are specified as Boolean expressions which have precise semantics. Given its inherent simplicity and neat formalism, the Boolean model received great attention in the past years and was adopted by many of the early commercial bibliographic systems (**Figure 3.1**).

Unfortunately, the Boolean model suffers from major drawbacks. First, its retrieval strategy is based on a binary decision criterion (i.e., a document is predicted to be either relevant or nonrelevant) without any notion of a grading scale, which prevents good retrieval performance. Thus, the Boolean model is in reality much more a data (instead of information) retrieval model. Second, while Boolean expressions have precise semantics, frequently it is not simple to translate an information need into a Boolean expression. In fact, most users find it difficult and awkward to express their query requests in terms of Boolean expressions. The Boolean expressions actually formulated by users often are quite simple (see Chapter 10 for a more thorough discussion on this issue). Despite these drawbacks, the Boolean model is still the dominant model with commercial document database systems and provides a good starting point for those new to the field.

The Boolean model considers that index terms are present or absent in a document. As a result, the index term weights are assumed to be all binary, that is, $w_{i,j} \in \{0, 1\}$. A query q is composed of index terms linked by three connectives: *not*, *and*, and *or*. Thus, a query is essentially a conventional Boolean expression which can be represented as a disjunction of conjunctive vectors (i.e., in *disjunctive normal form* (DNF)). For instance, the query $[q = k_a \wedge (k_b \vee \neg k_c)]$ can be written in disjunctive normal form as $[d_{dnf}^{\rightarrow} = (1, 1, 1) \vee (1, 1, 0) \vee (1, 0, 0)]$, where each of the components is a binary weighted vector associated with the tuple (k_a, k_b, k_c) . These binary weighted vectors are called the conjunctive components of q_{dnf}^{\rightarrow} . **Figure 2.3** illustrates the three conjunctive components for the query q .

Definition 3.2. For the Boolean model, the index term weight variables are all binary, that is, $w_{i,j} \in \{0, 1\}$. A query q is a conventional Boolean expression. Let q_{dnf}^{\rightarrow} be

the disjunctive normal form for the query q . Further, let \vec{q}_{cc} be any of the conjunctive components of \vec{q}_{dnf} . The similarity of a document d_j to the query q is defined as

$$\begin{cases} 1 & \text{if } \exists \vec{q}_{cc} | (\vec{q}_{cc} \in \vec{q}_{dnf}) \wedge (\forall k_i, g_i(\vec{d}_j) = g_i(\vec{q}_{cc})) \\ 0 & \text{otherwise} \end{cases} \quad (3.2)$$

If $\text{sim}(d_j, q) = 1$, then the Boolean model predicts that the document d_j is relevant to the query q (it might not be). Otherwise, the prediction is that the document is not relevant.

The Boolean model predicts that each document is either relevant or nonrelevant. There is no notion of a partial match to the query conditions. For instance, let d_j be a document for which $\vec{d}_j = (0, 1, 0)$. Document d_j includes the index term k_b but is considered nonrelevant to the query $[q = k_a \wedge (k_b \vee \neg k_c)]$.

The main advantages of the Boolean model are the clean formalism behind the model and its simplicity. The main disadvantages are that exact matching may lead to retrieval of too few or too many documents (see Chapter 10). Today, it is well known that index term weighting can lead to a substantial improvement in retrieval performance. Index term weighting brings us to the vector model.

3.3.3 Vector model

The vector model [88, 89] recognizes that the use of binary weights is too limiting and proposes a framework in which partial matching is possible. This is accomplished by assigning *nonbinary* weights to index terms in queries and in documents. These term weights are ultimately used to compute the *degree of similarity* between each document stored in the system and the user query. By sorting the retrieved documents in decreasing order of this degree of similarity, the vector model takes into consideration documents which match the query terms only partially. The main resultant effect is that the ranked document answer set is a lot more precise (in the sense that it better matches the user information need) than the document answer set retrieved by the Boolean model.

Definition 3.3. For the vector model, the weight $w_{i,j}$ associated with a pair (k_i, d_j) is positive and nonbinary. Further, the index terms in the query are also weighted. Let $w_{i,q}$ be the weight associated with the pair $[k_i, q]$, where $w_{i,q} \geq 0$. Then, the query vector \vec{q} is defined as $\vec{q} = (w_{1,q}, w_{2,q}, \dots, w_{t,q})$ where t is the total number of index terms in the system. As before, the vector for a document d_j is represented by $\vec{d}_j = (w_{1,j}, w_{2,j}, \dots, w_{t,j})$.

Therefore, a document d_j and a user query q are represented as t -dimensional vectors as shown in **Figure 3.2**. The vector model proposes to evaluate the degree of similarity of the document d_j with regard to the query q as the correlation between the vectors \vec{d}_j and \vec{q} . This correlation can be quantified, for instance, by the *cosine of the angle* between these two vectors. That is,

$$\begin{aligned} \text{sim}(d_j, q) &= \frac{\vec{d}_j \cdot \vec{q}}{|\vec{d}_j| \times |\vec{q}|} \\ &= \frac{\sum_{i=1}^t w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^t w_{i,j}^2} \times \sqrt{\sum_{i=1}^t w_{i,q}^2}}, \end{aligned} \quad (3.3)$$

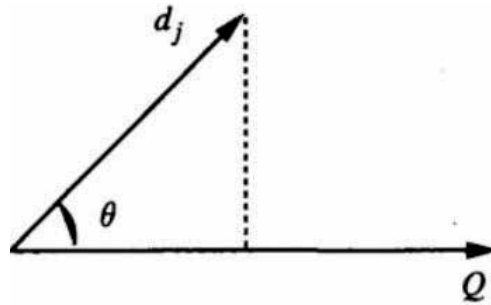


Figure 3.2.
The cosine of θ is adopted as $\text{sim}(d_j, q)$.

where $|\vec{d}_j|$ and $|\vec{q}|$ are the norms of the document and query vectors. The factor $|\vec{q}|$ does not affect the ranking (i.e., the ordering of the documents) because it is the same for all documents. The factor $|\vec{d}_j|$ provides a normalization in the space of the documents.

Since $w_{i,j} \geq 0$ and $w_{i,j} \geq 0$, $\text{sim}(q, d_j)$ varies from 0 to +1. Thus, instead of attempting to predict whether a document is relevant or not, the vector model ranks the documents according to their degree of similarity to the query. A document might be retrieved even if it matches the query only partially. For instance, one can establish a threshold on $\text{sim}(d_j, q)$ and retrieve the documents with a degree of similarity above that threshold. But to compute rankings, we need first to specify how index term weights are obtained.

Index term weights can be calculated in many different ways. The work by Salton and McGill [88] reviews various term-weighting techniques. Here, we do not discuss them in detail. Instead, we concentrate on elucidating the main idea behind the most effective term-weighting techniques. This idea is related to the basic principles which support clustering techniques, as follows.

Given a collection C of objects and a vague description of a set A , the goal of a simple clustering algorithm might be to separate the collection C of objects into two sets: the first one which is composed of objects related to the set A and the second one which is composed of objects not related to the set A . Vague description here means that we do not have complete information for deciding precisely which objects are and which are not in the set A . For instance, one might be looking for a set A of cars which have a price comparable to that of a Lexus 400. Since it is not clear what the term comparable means exactly, there is not a precise (and unique) description of the set A . More sophisticated clustering algorithms might attempt to separate the objects of a collection into various clusters (or classes) according to their properties. For instance, patients of a doctor specializing in cancer could be classified into five classes: terminal, advanced, metastasis, diagnosed, and healthy. Again, the possible class descriptions might be imprecise (and not unique), and the problem is one of deciding to which of these classes a new patient should be assigned. In what follows, however, we only discuss the simpler version of the clustering problem (i.e., the one which considers only two classes) because all that is required is a decision on which documents are predicted to be relevant and which ones are predicted to be not relevant (with regard to a given user query).

To view the IR problem as one of clustering, we refer to the early work of Salton. We think of the documents as a collection C of objects and think of the user query as a (vague) specification of a set A of objects. In this scenario, the IR problem can be reduced to the problem of determining which documents are in the set A and

which ones are not (i.e., the IR problem can be viewed as a clustering problem). In a clustering problem, two main issues have to be resolved. First, one needs to determine what the features are which better describe the objects in the set A . Second, one needs to determine what the features are which better distinguish the objects in the set A from the remaining objects in the collection C . The first set of features provides for quantification of *intra-cluster similarity*, while the second set of features provides for quantification of *inter-cluster dissimilarity*. The most successful clustering algorithms try to balance these two effects.

In the vector model, intra-clustering similarity is quantified by measuring the raw frequency of a term k_i inside a document d_j . Such term frequency is usually referred to as the *tf factor* and provides one measure of how well that term describes the document contents (i.e., intra-document characterization). Furthermore, inter-cluster dissimilarity is quantified by measuring the inverse of the frequency of a term k_i among the documents in the collection. This factor is usually referred to as the inverse document frequency or the *idf factor*. The motivation for the usage of an *idf factor* is that terms which appear in many documents are not very useful for distinguishing a relevant document from a nonrelevant one. As with good clustering algorithms, the most effective term-weighting schemes for IR try to balance these two effects.

Definition 3.4. Let N be the total number of documents in the system and n_i be the number of documents in which the index term k_i appears. Let $freq_{i,j}$ be the row frequency of term k_i in the document d_j (i.e., the number of times the term k_i is mentioned in the text of the document d_j). Then, the normalized frequency $f_{i,j}$ of term k_i in document d_j is given by

$$f_{i,j} = \frac{freq_{i,j}}{\max_l freq_{l,j}}, \quad (3.4)$$

where the maximum is computed over all terms which are mentioned in the text of the document d_j . If the term k_i does not appear in the document d_j , then $f_{i,j} = 0$. Further, let *idf_i*, inverse document frequency for k_i , be given by

$$idf_i = \log \frac{N}{n_i}. \quad (3.5)$$

The best known term-weighting schemes use weights which are given by

$$w_{i,j} = f_{i,j} \times \log \frac{N}{n_i} \quad (3.6)$$

or by a variation of this formula. Such term-weighting strategies are called *tf-idf schemes*.

Several variations of the above expression for the weight $w_{i,j}$ are described in an interesting paper by Salton and Buckley which appeared in 1988 [90]. However, in general, the above expression should provide a good weighting scheme for many collections.

For the query term weights, Salton and Buckley suggest

$$w_{i,q} = 0.5 + \frac{0.5 freq_{i,q}}{\max_l freq_{l,q}} \left(\times \log \frac{N}{n_i} \right), \quad (3.7)$$

where $freq_{i,q}$ is the raw frequency of the term k_i in the text of the information request q .

The main advantages of the vector model are:

1. Its term-weighting scheme improves retrieval performance.
2. Its partial-matching strategy allows retrieval of documents that approximate the query conditions.
3. Its cosine ranking formula sorts the documents according to their degree of similarity to the query.

Theoretically, the vector model has the disadvantage that index terms are assumed to be mutually independent [Eq. (2.3) does not account for index term dependencies]. However, in practice, consideration of term dependencies might be a disadvantage. Due to the locality of many term dependencies, their indiscriminate application to all the documents in the collection might in fact hurt the overall performance.

Despite its simplicity, the vector model is a resilient ranking strategy with general collections. It yields ranked answer sets which are difficult to improve upon without query expansion or relevance feedback within the framework of the vector model. A large variety of alternative ranking methods have been compared to the vector model, but the consensus seems to be that, in general, the vector model is either superior or almost as good as the known alternatives. Furthermore, it is simple and fast. For these reasons, the vector model is a popular retrieval model nowadays.

3.3.4 Probabilistic model

In this section, we describe the classic probabilistic model introduced in 1976 by Robertson and Sparck Jones [91] which later became known as the *binary independence retrieval* (BIR) model. Our discussion is intentionally brief and focuses mainly on highlighting the key features of the model. With this purpose in mind, we do not detain ourselves in subtleties regarding the binary independence assumption for the model. The section on bibliographic discussion points to references which cover these details.

The probabilistic model attempts to capture the IR problem within a probabilistic framework. The fundamental idea is as follows. Given a user query, there is a set of documents which contains exactly the relevant documents and no other. Let us refer to this set of documents as the ideal answer set. Given the description of this ideal answer set, we would have no problems in retrieving its documents. Thus, we can think of the querying process as a process of specifying the properties of an ideal answer set (which is analogous to interpreting the IR problem as a problem of clustering). The problem is that we do not know exactly what these properties are. All we know is that there are index terms whose semantics should be used to characterize these properties. Since these properties are not known at query time, an effort has to be made at initially guessing what they could be. This initial guess allows us to generate a preliminary probabilistic description of the ideal answer set which is used to retrieve the first set of documents. An interaction with the user is then initiated with the purpose of improving the probabilistic description of the ideal answer set. Such interaction could proceed as follows.

The user takes a look at the retrieved documents and decides which ones are relevant and which ones are not (in truth, only the first top documents need to be examined). The system then uses this information to refine the description of the ideal answer set. By repeating this process many times, it is expected that such a

description will evolve and become closer to the real description of the ideal answer set. Thus, one should always have in mind the need to guess at the beginning the description of the ideal answer set. Furthermore, a conscious effort is made to model this description in probabilistic terms.

The probabilistic model is based on the following fundamental assumption.

Definition 3.5. Assumption (probabilistic principle). Given a user query q and a document d_j in the collection, the probabilistic model tries to estimate the probability that the user will find the document d_j interesting (i.e., relevant). The model assumes that this probability of relevance depends on the query and the document representations only. Further, the model assumes that there is a subset of all documents which the user prefers as the answer set for the query q . Such an ideal answer set is labeled R and should maximize the overall probability of relevance to the user. Documents in the set R are predicted to be relevant to the query. Documents not in this set are predicted to be nonrelevant.

This assumption is quite troublesome because it does not state explicitly how to compute the probabilities of relevance. In fact, not even the sample space which is to be used for defining such probabilities is given.

Given a query q , the probabilistic model is assigned to each document d_j , as a measure of its similarity to the query, the ratio $P(d_j \text{ relevant to } q) / P(d_j \text{ nonrelevant to } q)$ which computes the odds of the document d_j being relevant to the query q . Taking the odds of relevance as the rank minimizes the probability of an erroneous judgment [92, 93].

Definition 3.6. For the probabilistic model, the index term weight variables are all binary, i.e., $w_{i,j} \in \{0, 1\}$ and $w_{i,q} \in \{0, 1\}$. A query q is a subset of index terms. Let R be the set of documents known (or initially guessed) to be relevant. Let \bar{R} be the complement of R [i.e., the set of nonrelevant documents]. Let $P(R|\vec{d}_j)$ be the probability that the document d_j is relevant to the query q and $P(\bar{R}|\vec{d}_j)$ be the probability that d_j is nonrelevant to q . The similarity $\text{sim}(d_j, q)$ of the document d_j to the query q is defined as the ratio:

$$\text{sim}(d_j, q) = \frac{P(R|\vec{d}_j)}{P(\bar{R}|\vec{d}_j)}. \quad (3.8)$$

Using Bayes' rule,

$$\text{sim}(d_j, q) = \frac{P(\vec{d}_j|R) \times P(R)}{P(\vec{d}_j|\bar{R}) \times P(\bar{R})}. \quad (3.9)$$

$P(\vec{d}_j|R)$ stands for the probability of randomly selecting the document d_j from the set R of relevant documents. Further, $P(R)$ stands for the probability that a document randomly selected from the entire collection is relevant. The meanings attached to $P(\vec{d}_j|\bar{R})$ and $P(\bar{R})$ are analogous and complementary.

Since $P(R)$ and $P(\bar{R})$ are the same for all the documents in the collection, we write

$$\text{sim}(d_j, q) \sim \frac{P(\vec{d}_j|R)}{P(\vec{d}_j|\bar{R})}. \quad (3.10)$$

Assuming independence of index terms,

$$\text{sim}(d_j, q) \sim \frac{\left(\prod_{g_i(\vec{d}_j=1)} P(k_i|R) \right) \times \left(\prod_{g_i(\vec{d}_j=0)} P(\bar{k}_i|R) \right)}{\left(\prod_{g_i(\vec{d}_j=1)} P(k_i|\bar{R}) \right) \times \left(\prod_{g_i(\vec{d}_j=0)} P(\bar{k}_i|\bar{R}) \right)}. \quad (3.11)$$

$P(k_i|R)$ stands for the probability that the index term k is present in a document randomly selected from the set R . $P(\bar{k}_i|R)$ stands for the probability that the index term k_i is not present in a document randomly selected from the set R . The probabilities associated with the set \bar{R} have meanings which are analogous to the ones just described.

Taking logarithms, recalling that $P(k_i|R) + P(\bar{k}_i|R) = 1$, and ignoring factors which are constant for all documents in the context of the same query, we can finally write

$$\text{sim}(d_j, q) \sim \sum_{i=1}^t w_{i,q} \times w_{i,j} \times \left(\log \frac{P(k_i|R)}{1 - P(k_i|R)} + \log \frac{1 - P(k_i|\bar{R})}{P(k_i|\bar{R})} \right), \quad (3.12)$$

which is a key expression for ranking computation in the probabilistic model.

Since we do not know the set R at the beginning, it is necessary to devise a method for initially computing the probabilities $P(k_i|R)$ and $P(k_i|\bar{R})$. There are many alternatives for such computation. We discuss a couple of them below.

In the very beginning (i.e., immediately after the query specification), there are no retrieved documents. Thus, one has to make simplifying assumptions such as (a) assume that $P(k_i|R)$ is constant for all index terms k_i (typically, equal to 0.5) and (b) assume that the distribution of index terms among the nonrelevant documents can be approximated by the distribution of index terms among all the documents in the collection. These two assumptions yield

$$\begin{aligned} P(k_i|R) &= 0.5 \\ P(k_i|\bar{R}) &= \frac{n_i}{N}, \end{aligned} \quad (3.13)$$

where, as already defined, n_i is the number of documents which contain the index term k_i and N is the total number of documents in the collection. Given this initial guess, we can then retrieve documents which contain query terms and provide an initial probabilistic ranking for them. After that, this initial ranking is improved as follows.

Let V be a subset of the documents initially retrieved and ranked by the probabilistic model. Such a subset can be defined, for instance, as the top r ranked documents where r is a previously defined threshold. Further, let V_i be the subset of V composed of the documents in V which contain the index term k_i . For simplicity, we also use V and V_i to refer to the number of elements in these sets (it should always be clear when the used variable refers to the set or to the number of elements in it). For improving the probabilistic ranking, we need to improve our guesses for $P(k_i|R)$ and $P(k_i|\bar{R})$. This can be accomplished with the following assumptions: (a) we can approximate $P(k_i|R)$ by the distribution of the index term k_i among the documents retrieved so far and (b) we can approximate $P(k_i|\bar{R})$ by considering that all the non-retrieved documents are not relevant. With these assumptions, we can write

$$\begin{aligned} P(k_i|R) &= \frac{v_i}{V} \\ P(k_i|\bar{R}) &= \frac{n_i - V_i}{N - V}. \end{aligned} \quad (3.14)$$

This process can then be repeated recursively. By doing so, we are able to improve on our guesses for the probabilities $P(k_i|R)$ and $P(k_i|\bar{R})$ without any assistance from a human subject (contrary to the original idea). However, we can also use assistance from the user for definition of the subset V as originally conceived.

The last formulas for $P(k_i|R)$ and $P(k_i|\bar{R})$ pose problems for small values of V and V_i which arise in practice (such as $V = 1$ and $V_i = 0$). To circumvent these problems, an adjustment factor is often added in which yields

$$\begin{aligned} P(k_i|R) &= \frac{v_i + 0.5}{V + 1} \\ P(k_i|\bar{R}) &= \frac{n_i - V_i + 0.5}{N - V + 1}. \end{aligned} \quad (3.15)$$

An adjustment factor which is constant and equal to 0.5 is not always satisfactory. An alternative is to take the fraction n_i/N as the adjustment factor which yields

$$\begin{aligned} P(k_i|R) &= \frac{v_i + \frac{n_i}{N}}{V + 1} \\ P(k_i|\bar{R}) &= \frac{n_i - V_i + \frac{n_i}{N}}{N - V + 1}. \end{aligned} \quad (3.16)$$

This completes our discussion of the probabilistic model.

The main advantage of the probabilistic model, in theory, is that documents are ranked in decreasing order of their probability of being relevant. The disadvantages include (1) the need to guess the initial separation of documents into relevant and nonrelevant sets, (2) the fact that the method does not take into account the frequency with which an index term occurs inside a document (i.e., all weights are binary), and (3) the adoption of the independence assumption for index terms. However, as discussed for the vector model, it is not clear that independence of index terms is a bad assumption in practical situations.

3.3.5 Brief comparison of classic models

In general, the Boolean model is considered to be the weakest classic method. Its main problem is the inability to recognize partial matches which frequently leads to poor performance. There is some controversy as to whether the probabilistic model outperforms the vector model. Croft performed some experiments and suggested that the probabilistic model provides a better retrieval performance. However, experiments done afterward by Salton and Buckley refute that claim. Through several different measures, Salton and Buckley showed that the vector model is expected to outperform the probabilistic model with general collections. This also seems to be the dominant thought among researchers, practitioners, and the Web community, where the popularity of the vector model runs high.

3.4. Text retrieval

3.4.1 Formats

There is no single format for a text document, and an IR system should be able to retrieve information from many of them. In the past, IR systems would convert a

document to an internal format. However, that has many disadvantages, because the original application related to the document is not useful anymore. On top of that, we cannot change the contents of a document. Current IR systems have filters that can handle most popular documents, in particular those of word processors with some binary syntax such as Word, WordPerfect, or FrameMaker. Even then, good filters might not be possible if the format is proprietary and its details are not public. This is not the case for full ASCII syntax, as in TeX documents. Although documents can be in a binary format (e.g., parts of a Word document), documents that are represented in human-readable ASCII form imply more portability and are easier to modify (e.g., they can be edited with different applications).

Other text formats were developed for document interchange. Among these we should mention the rich text format (RTF), which is used by word processors and has ASCII syntax. Other important formats were developed for displaying or printing documents. The most popular ones are the portable document format (PDF) and PostScript (which is a powerful programming language for drawing). Other interchange formats are used to encode electronic mail, for example, multipurpose internet mail exchange (MIME). MIME supports multiple character sets, multiple languages, and multiple media.

On top of these formats, nowadays many files are compressed. Text compression is treated in detail, but here we comment on the most popular compression software and associated formats. These include compress (Unix), ARJ (PCs), and ZIP (e.g., gzip in Unix and WinZip in Windows). Other tools allow us to convert binary files, in particular compressed text, to ASCII text such that it can be transmitted through a communication line using only seven bits. Examples of these tools are uuencode/ uudecode and BinHex.

3.4.2 Modeling natural language

Text is composed of symbols from a finite alphabet. We can divide the symbols in two disjoint subsets: symbols that separate words and symbols that belong to words. It is well known that symbols are not uniformly distributed. If we consider just letters (a to z), we observe that vowels are usually more frequent than most consonants. For example, in English, letter “e” has the highest frequency. A simple model to generate text is the binomial model. In it, each symbol is generated with a certain probability. However, natural language has a dependency on previous symbols. For example, in English, letter “f” cannot appear after letter “c,” and vowels or certain consonants have a higher probability of occurring. Therefore, the probability of a symbol depends on previous symbols. We can use a finite context or Markovian model to reflect this dependency. The model can consider one, two, or more letters to generate the next symbol. If we use k letters, we say that it is a k -order model (so, the binomial model is considered an 0-order model). We can use these models taking words as symbols. For example, text generated by a five-order model using the distribution of words in the Bible might make sense (i.e., it can be grammatically correct) but will be different from the original. More complex models include finite-state models (which define regular languages) and grammar models (which define context-free and other languages). However, finding the right grammar for natural language is still a difficult open problem.

The next issue is how the different words are distributed inside each document. An approximate model is *Zipf's law*, which attempts to capture the distribution of the frequencies (i.e., number of occurrences) of the words in the text. The rule states that the frequency of the i th most frequent word is $1/i^\theta$ times that of the most frequent word. This implies that in a text of n words with a vocabulary of V words,

the i th most frequent word appears $n/(i^\theta H_V(\theta))$ times, where $H_V(\theta)$ is the harmonic number of order θ of V , defined as

$$H_V(\theta) = \sum_{j=1}^V \frac{1}{j^\theta}, \quad (3.17)$$

so that the sum of all frequencies is n . The left side of **Figure 3.3** illustrates the distribution of frequencies considering that the words are arranged in decreasing order of their frequencies. The value of θ depends on the text. In the most simple formulation, $\theta = 1$, and therefore $H_V(\theta) = O(\log n)$. However, this simplified version is very inexact, and the case $\theta > 1$ (more precisely, between 1.5 and 2.0) fits better the real data. This case is very different, since the distribution is much more skewed, and $H_V(\theta) = O(1)$. Experimental data suggests that a better model is $k/(c+i)^\theta$ where c is an additional parameter and k is such that all frequencies add to n . This is called a Mandelbrot distribution.

Since the distribution of words is very skewed (i.e., there are a few hundred words which take up 50% of the text), words that are too frequent, such as stopwords, can be disregarded. A stopword is a word which does not carry meaning in natural language and therefore can be ignored (i.e., made not searchable), such as “a,” “the,” “by,” etc. Fortunately, the most frequent words are stopwords, and, therefore, half of the words appearing in a text do not need to be considered. This allows us, for instance, to significantly reduce the space overhead of indices for natural language texts. For example, the most frequent words in the TREC-2 collection are “the,” “of,” “and,” “a,” “to,” and “in.”

Another issue is the distribution of words in the documents of a collection. A simple model is to consider that each word appears the same number of times in every document. However, this is not true in practice. A better model is to consider a negative binomial distribution, which says that the fraction of documents containing a word k times is

$$F(k) = \binom{\alpha + k - 1}{k} p^k (1 + p)^{-\alpha - k}, \quad (3.18)$$

where p and α are parameters that depend on the word and the document collection. For example, for the Brown Corpus [94] and the word “said,” we have $p = 9.24$ and $\alpha = 0.42$ [95]. The latter reference gives other models derived from a Poisson distribution.

The next issue is the number of distinct words in a document. This set of words is referred to as the document vocabulary. To predict the growth of the vocabulary size in natural language text, we use the so-called Heaps’ law [96]. This is a very precise law which states that the vocabulary of a text of size n words is of size $V = Kn^\beta = O(n^\beta)$, where K and β depend on the particular text. The right side of **Figure 3.3** illustrates how the vocabulary size varies with the text size. K is normally between 10 and 100, and β is a positive value less than one. Some experiments on the TREC-2 collection show that the most common values for β are between 0.4 and 0.6. Hence, the vocabulary of a text grows sublinearly with the text size, in a proportion close to its square root.

Notice that the set of different words of a language is fixed by a constant (e.g., the number of different English words is finite). However, the limit is so high that it is much more accurate to assume that the size of the vocabulary is $O(n^\beta)$ instead of $O(1)$, although the number should stabilize for huge enough texts. On the other hand, many authors argue that the number keeps growing anyway because of typing or spelling errors.

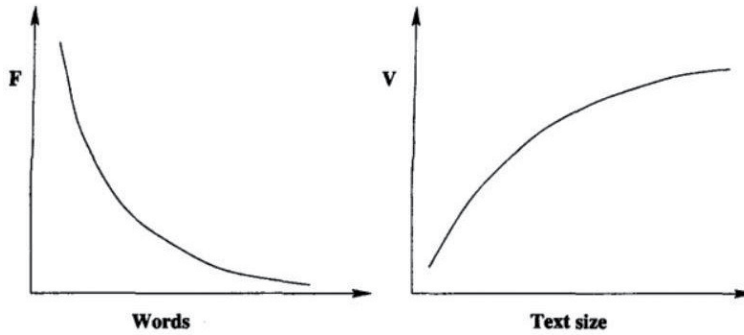


Figure 3.3. Distribution of sorted word frequencies (left) and size of the vocabulary (right).

Heaps' law also applies to collections of documents because, as the total text size grows, the predictions of the model become more accurate. Furthermore, this model is also valid for the World Wide Web.

The last issue is the average length of words. This relates the text size in words with the text size in bytes (without accounting for punctuation and other extra symbols). For example, in the different subcollections of the TREC-2 collection, the average word length is very close to five letters, and the range of variation of this average in each subcollection is small (from 4.8 to 5.3 letters). If the stopwords were move, the average length of a word increases to a number between 6 and 7 (letters). If we take only the words of the vocabulary, the average length is higher (about 8 or 9). This defines the total space needed for the vocabulary.

Heaps' law implies that the length of the words in the vocabulary increases logarithmically with the text size and, thus, that longer and longer words should appear as the text grows. However, in practice, the average length of the words in the overall text is constant because shorter words are common enough (e.g., stopwords). This balance between short and long words, such that the average word length remains constant, has been noticed many times in different contexts and can also be explained by a finite-state model in which (a) the space character has a probability close to 0.2, (b) the space character cannot appear twice subsequently, and (c) there are 26 letters. This simple model is consistent with Zipf's and Heaps' laws.

The models presented in this section are used in Chapters 8 and 13, in particular Zipf's and Heaps' laws.

3.4.3 Similarity models

In this section, we define notions of syntactic similarity between strings and documents. Similarity is measured by a distance function. For example, if we have strings of the same length, we can define the distance between them as the number of positions that have different characters. Then, the distance is 0 if they are equal. This is called the Hamming distance. A distance function should also be symmetric (i.e., the order of the arguments does not matter) and should satisfy the triangle inequality (i.e., $distance(a, c) \leq distance(a, b) + distance(b, c)$).

An important distance over strings is the edit or Levenshtein distance mentioned earlier. The edit distance is defined as the minimum number of characters, insertions, deletions, and substitutions that we need to perform in any of the strings to make them equal. For instance, the edit distance between "color" and "colour" is

one, while the edit distance between “survey” and “surgery” is two. The edit distance is considered to be superior for modeling syntactic errors than other more complex methods such as the Soundex system, which is based on phonetics [97]. Extensions to the concept of edit distance include different weights for each operation, adding transpositions, etc.

There are other measures. For example, assume that we are comparing two given strings and the only operation allowed is deletion of characters. Then, after all non-common characters have been deleted, the remaining sequence of characters (not necessarily contiguous in the original string, but in the same order) is the longest common subsequence (LCS) of both strings. For example, the LCS of “survey” and “surgery” is “surey.”

Similarity can be extended to documents. For example, we can consider lines as single symbols and compute the longest common sequence of lines between two files. This is the measure used by the diff command in Unix-like operating systems. The main problem with this approach is that it is very time-consuming and does not consider lines that are similar. The latter drawback can be fixed by taking a weighted edit distance between lines or by computing the LCS over all the characters. Other solutions include extracting fingerprints (any piece of text that in some sense characterizes it) for the documents and comparing them or finding large repeated pieces. There are also visual tools to see document similarity. For example, Dotplot draws a rectangular map where both coordinates are file lines and the entry for each coordinate is a gray pixel that depends on the edit distance between the associated lines.

3.5. Document preprocessing

Document preprocessing is a procedure which can be divided mainly into five text operations (or transformations):

1. Lexical analysis of the text with the objective of treating digits, hyphens, punctuation marks, and the case of letters.
2. Elimination of stopwords with the objective of filtering out words with very low discrimination values for retrieval purposes.
3. Stemming of the remaining words with the objective of removing affixes (i.e., prefixes and suffixes) and allowing the retrieval of documents containing syntactic variations of query terms (e.g., connect, connecting, connected, etc.).
4. Selection of index terms to determine which words/stems (or groups of words) will be used as indexing elements. Usually, the decision on whether a particular word will be used as an index term is related to the syntactic nature of the word. In fact, noun words frequently carry more semantics than adjectives, adverbs, and verbs.
5. Construction of term categorization structures such as a thesaurus, or extraction of structure directly represented in the text, for allowing the expansion of the original query with related terms (a usually useful procedure).

In the following, each of these phases is discussed in detail. But, before proceeding, let us take a look at the logical view of the documents which results after each of the above phases is completed. **Figure 3.4** is repeated here for convenience as

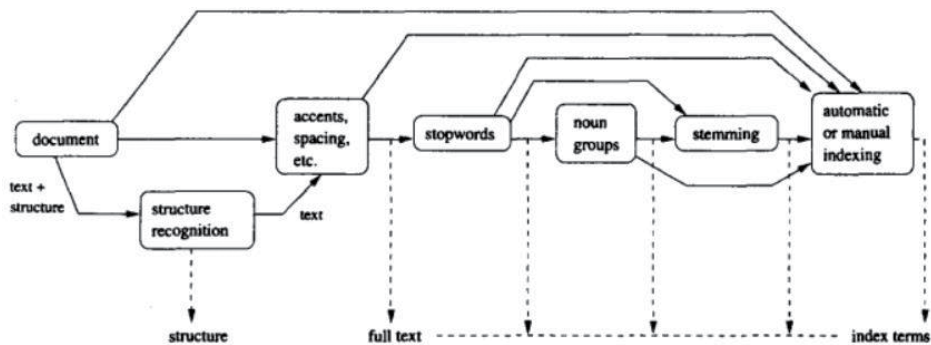


Figure 3.4.
Logical view of a document throughout the various phases of text preprocessing.

Figure 3.4. As already discussed, by aggregating the preprocessing phases, we are able to move the logical view of the documents (adopted by the system) from that of a full text to that of a set of high-level indexing terms.

3.5.1 Lexical analysis of the text

Lexical analysis is the process of converting a stream of characters (the text of the documents) into a stream of words (the candidate words to be adopted as index terms). Thus, one of the major objectives of the lexical analysis phase is the identification of the words in the text. At the first glance, all that seems to be involved is the recognition of spaces as word separators (in which case, multiple spaces are reduced to one space). However, there is more to it than this. For instance, the following four particular cases have to be considered with care [98]: digits, hyphens, punctuation marks, and the case of the letters (lower and upper case).

Numbers are usually not good index terms because, without a surrounding context, they are inherently vague. For instance, consider that a user is interested in documents about the number of deaths due to car accidents between the years 1910 and 1989. Such a request could be specified as the set of index terms: deaths, car, accidents, years, 1910 and 1989. However, the presence of the numbers 1910 and 1989 in the query could lead to the retrieval, for instance, of a variety of documents which refer to either of these 2 years. The problem is that numbers by themselves are just too vague. Thus, in general it is wise to disregard numbers as index terms. However, we have also to consider that digits might appear mixed within a word. For instance, “51OB.C.” is a clearly important index term. In this case, it is not clear what rule should be applied. Furthermore, a sequence of 16 digits identifying a credit card number might be highly relevant in a given context and, in this case, should be considered as an index term. A preliminary approach for treating digits in the text might be to remove all words containing sequences of digits unless specified otherwise (through regular expressions). Further, an advanced lexical analysis procedure might perform some date and number normalization to unify formats.

Hyphens pose another difficult decision to the lexical analyzer. Breaking up hyphenated words might be useful due to inconsistency of usage. For instance, this allows treating “state-of-the-art” and “state of the art” identically. However, there are words which include hyphens as an integral part, for instance, giltedge, B-49, etc. Again, the most suitable procedure seems to adopt a general rule and specify the exceptions on a case-by-case basis.

Normally, punctuation marks are removed entirely in the process of lexical analysis. While some punctuation marks are an integral part of the word (for instance, “51OB.C.”), removing them does not seem to have an impact in retrieval

performance because the risk of misinterpretation in this case is minimal. In fact, if the user specifies “5LOB.C” in his query, removal of the dot both in the query term and in the documents will not affect retrieval. However, very particular scenarios might again require the preparation of a list of exceptions. For instance, if a portion of a program code appears in the text, it might be wise to distinguish between the variables “x.id” and “xid.” In this case, the dot mark should not be removed.

The case of letters is usually not important for the identification of index terms. As a result, the lexical analyzer normally converts all the text to either lower or upper case. However, once more, very particular scenarios might require the distinction to be made. For instance, when looking for documents which describe details about the command language of a Unix-like operating system, the user might explicitly desire the non-conversion of upper cases because this is the convention in the operating system. Further, part of the semantics might be lost due to case conversion. For instance, the words Bank and bank have different meanings—a fact common to many other pairs of words.

As pointed out by Fox, all these text operations can be implemented without difficulty. However, careful thought should be given to each one of them because they might have a profound impact at document retrieval time. This is particularly worrisome in those situations in which the user finds it difficult to understand what the indexing strategy is doing. Unfortunately, there is no clear solution to this problem. As already mentioned, some Web search engines are opting for avoiding text operations altogether because this simplifies the interpretation the user has of the retrieval task. Whether this strategy will be the one of choice in the long term remains to be seen.

3.5.2 Elimination of stopwords

As discussed in Chapter 2, words which are too frequent among the documents in the collection are not good discriminators. In fact, a word which occurs in 80% of the documents in the collection is useless for purposes of retrieval. Such words are frequently referred to as stopwords and are normally filtered out as potential index terms. Articles, prepositions, and conjunctions are natural candidates for a list of stopwords.

Elimination of stopwords has an additional important benefit. It reduces the size of the indexing structure considerably. In fact, it is typical to obtain a compression in the size of the indexing structure (for instance, in the size of an inverted list, see Chapter 8) of 40% or more solely with the elimination of stopwords.

Since stopword elimination also provides for compression of the indexing structure, the list of stopwords might be extended to include words other than articles, prepositions, and conjunctions. For instance, some verbs, adverbs, and adjectives could be treated as stopwords. In [99], a list of 425 stopwords is illustrated. Programs in C for lexical analysis are also provided.

Despite these benefits, elimination of stopwords might reduce recall. For instance, consider a user who is looking for documents containing the phrase “to be or not to be.” Elimination of stopwords might leave only the term *bemaking* it almost impossible to properly recognize the documents which contain the phrase specified. This is one additional reason for the adoption of a full-text index (i.e., insert all words in the collection into the inverted file) by some Web search engines.

3.5.3 Stemming

Frequently, the user specifies a word in a query, but only a variant of this word is present in a relevant document. Plurals, gerund forms, and past tense suffixes are

examples of syntactical variations which prevent a perfect match between a query word and a respective document word. This problem can be partially overcome with the substitution of the words by their respective stems.

A stem is the portion of a word which is left after the removal of its affixes (i.e., prefixes and suffixes). A typical example of a stem is the word *connect* which is the stem for the variants *connected*, *connecting*, *connection*, and *connections*. Stems are thought to be useful for improving retrieval performance because they reduce variants of the same root word to a common concept. Furthermore, stemming has the secondary effect of reducing the size of the indexing structure because the number of distinct index terms is reduced.

While the argument supporting stemming seems sensible, there is controversy in the literature about the benefits of stemming for retrieval performance. In fact, different studies lead to rather conflicting conclusions. Frakes [99] compares eight distinct studies on the potential benefits of stemming. While he favors the usage of stemming, the results of the eight experimental studies he investigated do not allow us to reach a satisfactory conclusion. As a result of these doubts, many Web search engines do not adopt any stemming algorithm whatsoever.

Frakes distinguishes four types of stemming strategies: affix removal, table lookup, successor variety, and n-grams. Table lookup consists simply of looking for the stem of a word in a table. It is a simple procedure but one which is dependent on data on stems for the whole language. Since such data is not readily available and might require considerable storage space, this type of stemming algorithm might not be practical. Successor variety stemming is based on the determination of morpheme boundaries, uses knowledge from structural linguistics, and is more complex than affix removal stemming algorithms. N-grams stemming is based on the identification of digrams and trigrams and is more a term clustering procedure than a stemming one. Affix removal stemming is intuitive and simple and can be implemented efficiently. Thus, in the remainder of this section, we concentrate our discussion on algorithms for affix removal stemming only.

In affix removal, the most important part is suffix removal because most variants of a word are generated by the introduction of suffixes (instead of prefixes). While there are three or four well-known suffix removal algorithms, the most popular one is that by Porter because of its simplicity and elegance. Despite being simpler, the Porter algorithm yields results comparable to those of the more sophisticated algorithms.

The Porter algorithm uses a suffix list for suffix stripping. The idea is to apply a series of rules to the suffixes of the words in the text. For instance, the rule

$$s \rightarrow \emptyset \quad (3.19)$$

is used to convert plural forms into their respective singular forms by substituting the letter *s* by nil. Notice that to identify the suffix we must examine the last letters in the word. Furthermore, we look for the longest sequence of letters which matches the left-hand side in a set of rules. Thus, application of the two following rules

$$\begin{aligned} sses &\rightarrow ss \\ s &\rightarrow \emptyset \end{aligned} \quad (3.20)$$

to the word *stresses* yields the stem *stress* instead of the stem *stresse*. By separating such rules into five distinct phases, the Porter algorithm is able to provide effective stemming while running fast.

3.5.4 Index terms selection

If a full-text representation of the text is adopted, then all words in the text are used as index terms. The alternative is to adopt a more abstract view in which not all words are used as index terms. This implies that the set of terms used as indices must be selected. In the area of bibliographic sciences, such a selection of index terms is usually done by a specialist. An alternative approach is to select candidates for index terms automatically.

Distinct automatic approaches for selecting index terms can be used. A good approach is the identification of noun groups (as done in the INQUERY system [73]) which we now discuss.

A sentence in natural language text is usually composed of nouns, pronouns, articles, verbs, adjectives, adverbs, and connectives. While the words in each grammatical class are used with a particular purpose, it can be argued that most of the semantics is carried by the noun words. Thus, an intuitively promising strategy for selecting index terms automatically is to use the nouns in the text. This can be done through the systematic elimination of verbs, adjectives, adverbs, connectives, articles, and pronouns.

Since it is common to combine two or three nouns in a single component (e.g., computer science), it makes sense to cluster nouns which appear nearby in the text into a single indexing component (or concept). Thus, instead of simply using nouns as index terms, we adopt noun groups. A noun group is a set of nouns whose syntactic distance in the text (measured in terms of number of words between two nouns) does not exceed a predefined threshold (for instance, three).

When noun groups are adopted as indexing terms, we obtain a conceptual logical view of the documents in terms of sets of nonelementary index terms.

3.5.5 Text compression

Text compression is about finding ways to represent the text in fewer bits or bytes. The amount of space required to store text on computers can be reduced significantly using compression techniques. Compression methods create a reduced representation by identifying and using structures that exist in the text. From the compressed version, the original text can be reconstructed exactly.

Text compression is becoming an important issue in an information retrieval environment. The widespread use of digital libraries, office automation systems, document databases, and the Web has led to an explosion of textual information available online. In this scenario, text compression appears as an attractive option for reducing costs associated with space requirements, input/output (I/O) overhead, and communication delays. The gain obtained from compressing text is that it requires less storage space, it takes less time to be transmitted over a communication link, and it takes less time to search directly the compressed text. The price paid is the time necessary to code and decode the text.

A major obstacle for storing text in compressed form is the need for IR systems to access text randomly. To access a given word in a compressed text, it is usually necessary to decode the entire text from the beginning until the desired word is reached. It could be argued that a large text could be divided into blocks that are compressed independently, thus allowing fast random access to each block. However, efficient compression methods need to process some text before making compression effective (usually more than 10 kilobytes). The smaller the blocks, the less effective compression is expected to be.

Our discussion here focuses on text compression methods which are suitable for use in an IR environment. For instance, a successful idea aimed at merging the

requirements of compression algorithms and the needs of IR systems is to consider that the symbols to be compressed are words and not characters (character-based compression is the more conventional approach). Words are the atoms on which most IR systems are built. Moreover, it is now known that much better compression is achieved by taking words as symbols (instead of characters). Further, new word-based compression methods allow random access to words within the compressed text which is a critical issue for an IR system.

Besides the economy of space obtained by a compression method, there are other important characteristics to be considered such as compression and decompression speed. In some situations, decompression speed is more important than compression speed. For instance, this is the case with textual databases in which it is common to compress the text once and to read it many times from disk.

Another important characteristic of a compression method is the possibility of performing compressed pattern matching, defined as the task of performing pattern matching in a compressed text without decompressing it. In this case, sequential searching can be speeded up by compressing the search key rather than decoding the compressed text being searched. As a consequence, it is possible to search faster on compressed text because much less text has to be scanned. Chapter 8 presents efficient methods to deal with searching the compressed text directly.

When the text collection is large, efficient text retrieval requires specialized index techniques. A simple and popular indexing structure for text collections are the inverted files. Inverted files (see Chapter 8 for details) are especially adequate when the pattern to be searched for is formed by simple words. Since this is a common type of query (for instance, when searching the Web), inverted files are widely used for indexing large text collections.

An inverted file is typically composed of (a) a vector containing all the distinct words in the text collection (which is called the vocabulary) and (b) for each word in the vocabulary a list of all documents (identified by document numbers) in which that word occurs. Because each list of document numbers (within the inverted file) is organized in ascending order, specific compression methods have been proposed for them, leading to very efficient index compression schemes. This is important because query processing time is highly related to index access time. Thus, in this section, we also discuss some of the most important index compression techniques.

We first introduce basic concepts related to text compression. We then present some of the most important statistical compression methods, followed by a brief review of compression methods based on a dictionary. At the end, we discuss the application of compression to inverted files.

There are two general approaches to text compression: statistical and dictionary based. Statistical methods rely on generating good probability estimates (of appearance in the text) for each symbol. The more accurate the estimates are, the better the compression obtained. A symbol here is usually a character, a text word, or a fixed number of characters. The set of all possible symbols in the text is called the alphabet. The task of estimating the probability on each next symbol is called modeling. A model is essentially a collection of probability distributions, one for each context in which a symbol can be coded. Once these probabilities are available, the symbols are converted into binary digits, a process called coding. In practice, both the encoder and decoder use the same model. The decoder interprets the output of the encoder (with reference to the same model) to find out the original symbol.

There are two well-known statistical coding strategies: Huffman coding and arithmetic coding. The idea of Huffman coding is to assign a fixed-length bit encoding to each different symbol of the text. Compression is achieved by assigning

a smaller number of bits to symbols with higher probabilities of appearance. Huffman coding was first proposed in the early 1950s and was the most important compression method until the late 1970s, when arithmetic coding made higher compression rates possible.

Arithmetic coding computes the code incrementally, one symbol at a time, as opposed to the Huffman coding scheme in which each different symbol is pre-encoded using a fixed-length number of bits. The incremental nature does not allow decoding a string which starts in the middle of a compressed file. To decode a symbol in the middle of a file compressed with arithmetic coding, it is necessary to decode the whole text from the very beginning until the desired word is reached. This characteristic makes arithmetic coding inadequate for use in an IR environment.

Dictionary methods substitute a sequence of symbols by a pointer to a previous occurrence of that sequence. The pointer representations are references to entries in a dictionary composed of a list of symbols (often called phrases) that are expected to occur frequently. Pointers to the dictionary entries are chosen so that they need less space than the phrase they replace, thus obtaining compression. The distinction between modeling and coding does not exist in dictionary methods, and there are no explicit probabilities associated to phrases. The most well-known dictionary methods are represented by a family of methods, known as the Ziv-Lempel family.

Character-based Huffman methods are typically able to compress English texts to approximately five bits per character (usually, each uncompressed character takes 7–8 bits to be represented). More recently, a word-based Huffman method has been proposed as a better alternative for natural language texts. This method is able to reduce English texts to just over two bits per character. As we will see later on, word-based Huffman coding achieves compression rates close to the entropy and allows random access to intermediate points in the compressed text. Ziv-Lempel methods are able to reduce English texts to fewer than four bits per character. Methods based on arithmetic coding can also compress English texts to just over two bits per character. However, the price paid is slower compression and decompression, and the impossibility of randomly accessing intermediate points in the compressed text.

Before proceeding, let us present an important definition which will be useful from now on.

Definition 3.7. Compression ratio is the size of the compressed file as a fraction of the uncompressed file.

3.6. Index construction and compression

3.6.1 Inverted files

An inverted file (or inverted index) is a word-oriented mechanism for indexing a text collection in order to speed up the searching task. The inverted file structure is composed of two elements: the vocabulary and the occurrences. The vocabulary is the set of all different words in the text. For each such word, a list of all the text positions where the word appears is stored. The set of all those lists is called the “occurrences” (**Figure 3.5** shows an example). These positions can refer to words or characters. Word positions (i.e., position i refers to the i th word) simplify phrase and proximity queries, while character positions (i.e., the position i is the i th character) facilitate direct access to the matching text positions.

Some authors make the distinction between inverted files and inverted lists. In an inverted file, each element of a list points to a document or file name, while

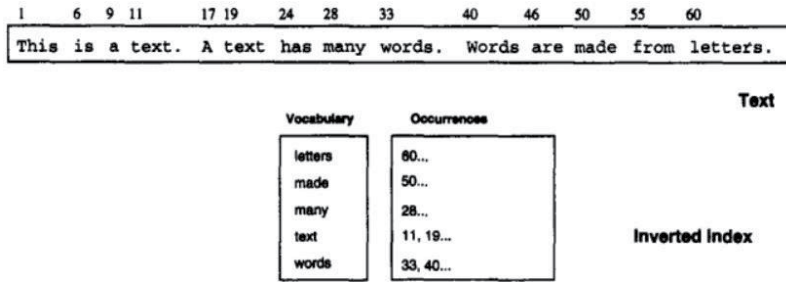


Figure 3.5. A sample text and an inverted index built on it. The words are converted to lower case, and some are not indexed. The occurrences point to character positions in the text.

inverted lists match our definition. We prefer not to make such a distinction because, as we will see later, this is a matter of the addressing granularity, which can range from text positions to logical blocks.

The space required for the vocabulary is rather small. According to Heaps’ law, the vocabulary grows as $O(n^\beta)$, where β is a constant between 0 and 1 dependent on the text, being between 0.4 and 0.6 in practice. For instance, for 1 Gb of the TREC-2 collection, the vocabulary has a size of only 5 Mb. This may be further reduced by stemming and other normalization techniques as described.

The occurrences demand much more space. Since each word appearing in the text is referenced once in that structure, the extra space is $O(n)$. Even omitting stopwords (which is the default practice when words are indexed), in practice the space overhead of the occurrences is between 30 and 40% of the text size.

To reduce space requirements, a technique called block addressing is used. The text is divided in blocks, and the occurrences point to the blocks where the word appears (instead of the exact positions). The classical indices which point to the exact occurrences are called “full inverted indices.” Using block addressing not only can the pointers be smaller because there are fewer blocks than positions, but also all the occurrences of a word inside a single block are collapsed to one reference (see **Figure 3.6**). Indices of only 5% overhead over the text size are obtained with this technique. The price to pay is that, if the exact occurrence positions are required (for instance, for a proximity query), then an online search over the qualifying blocks has to be performed. For instance, block addressing indices with 256 blocks stop working well with texts of 200 Mb.

Table 3.1 presents the projected space taken by inverted indices for texts of different sizes, with and without the use of stopwords. The full inversion stands for inverting all the words and storing their exact positions, using four bytes per

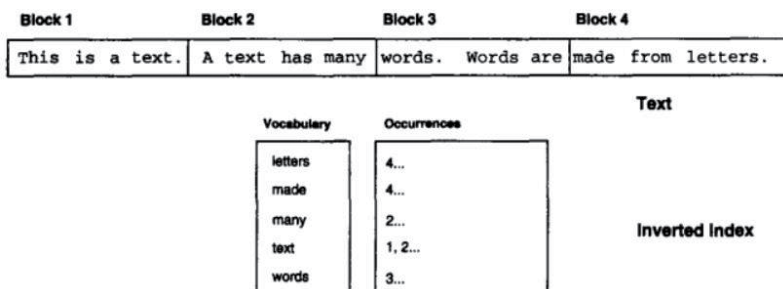


Figure 3.6. The sample text splits into four blocks, and an inverted index uses block addressing built on it. The occurrences denote block numbers. Notice that both occurrences of “words” collapsed into one.

Index	Small collection (1 Mb)		Medium collection (2 Mb)		Large collection (2 Gb)	
Addressing words	45%	73%	36%	64%	35%	63%
Addressing documents	19%	26%	18%	32%	26%	47%
Addressing 64K block	27%	41%	18%	32%	5%	9%
Addressing 256 blocks	18%	25%	1.7%	2.4%	0.5%	0.7%

Table 3.1.

Sizes of an inverted file as approximate percentages of the size of the whole text collection.

pointer. The document addressing index assumes that we point to documents which are of size 10 kb (and the necessary number of bytes per pointer, that is, one, two, and three bytes, depending on text size). The block addressing index assumes that we use 256 or 64K blocks (one or two bytes per pointer) independently of the text size. The space taken by the pointers can be significantly reduced by using compression. We assume that 45% of all the words are stopwords and that there is one non-stopword each 11.5 characters. Our estimation for the vocabulary is based on Heaps' law with parameters $V = 30n^{0.5}$. All these decisions were taken according to our experience and experimentally validated.

The blocks can be of fixed size (imposing a logical block structure over the text database), or they can be defined using the natural division of the text collection into files, documents, Web pages, or others. The division into blocks of fixed size improves efficiency at retrieval time, that is, the more variance in the block sizes, the more amount of text sequentially traversed on average. This is because larger blocks match queries more frequently and are more expensive to traverse.

Alternatively, the division using natural cuts may eliminate the need for online traversal. For example, if one block per retrieval unit is used and the exact match positions are not required, there is no need to traverse the text for single-word queries, since it is enough to know which retrieval units to report. But if, on the other hand, many retrieval units are packed into a single block, the block has to be traversed to determine which units to retrieve.

It is important to notice that in order to use block addressing, the text must be readily available at search time. This is not the case for remote text (as in Web search engines) or if the text is in a CD-ROM that has to be mounted, for instance. Some restricted queries not needing exact positions can still be solved if the blocks are retrieval units.

Four granulates and three collections are considered. For each collection, the right column considers that stopwords are not indexed, while the left column considers that all words are indexed.

3.6.2 Blocked sort-based indexing

The basic steps in constructing a non-positional index. We first make a pass through the collection assembling all termdocID pairs. We then sort the pairs with the term as the dominant key and docID as the secondary key. Finally, we organize the docIDs for each term into a postings list and compute statistics like term and document frequency. For small collections, all this can be done in memory. In this chapter, we describe methods for large collections that require the use of secondary storage.

To make index construction more efficient, we represent terms as termIDs, where each termID is a unique serial number. We can build the mapping from

terms to termIDs on the fly while we are processing the collection, or, in a two-pass approach, we compile the vocabulary in the first pass and construct the inverted index in the second pass. The index construction algorithms described in this chapter all do a single pass through the data. Section 4.7 gives references to multipass algorithms that are preferable in certain applications, for example, when disk space is scarce.

We work with the Reuters Corpus Volume I (RCV1) collection as our model collection in this chapter, a collection with roughly 1 GB of text. It consists of about 800,000 documents that were sent over the Reuters newswire during a 1-year period between August 20, 1996, and August 19, 1997. A typical document is shown in **Figure 3.7**, but note that we ignore multimedia information like images in this book and are only concerned with text. Reuters Corpus Volume I (RCV1) covers a wide range of international topics, including politics, business, sports, and (as in this example) science. Some key statistics of the collection are shown in **Table 3.2**.

Values are rounded for the computations in this book. The unrounded values are 806,791 documents, 222 tokens per document, 391,523 (distinct) terms, 6.04 bytes per token with spaces and punctuation, 4.5 bytes per token without spaces and punctuation, 7.5 bytes per term, and 96,969,056 tokens

Reuters Corpus Volume I (RCV1) has 100 million tokens. Collecting all termID-docID pairs of the collection using 4 bytes each for termID and docID therefore requires 0.8 GB of storage. Typical collections today are often one or two orders of magnitude larger than Reuters Corpus Volume I (RCV1). You can easily see how



Figure 3.7.
Document from the Reuters newswire.

Symbol	Statistic	Value
N	Documents	800,000
L	Avg. number of tokens per document	200
M	Terms	400,000
	Avg. number of bytes per token (incl. spaces/punct.)	6
	Avg. number of bytes per token (without spaces/punct.)	4.5
	Avg. number of bytes per term	7.5
T	Tokens	100,000,000

Table 3.2.
Collection statistics for Reuters Corpus Volume I (RCV1).

such collections overwhelm even large computers if we try to sort their termID-docID pairs in memory. If the size of the intermediate files during index construction is within a small factor of available memory, then the compression techniques introduced in Chapter 5 can help; however, the postings file of many large collections cannot fit into memory even after compression (**Figure 3.7**).

With the insufficient main memory, we need to use an external sorting algorithm, that is, one that uses disk. To achieve acceptable sorting efficiency, the central requirement of such an algorithm is that it minimizes the number of random disk seeks during sorting-sequential disk reads are far faster than seeks as we explained in Section 4.1. One solution is the blocked sort-based indexing algorithm or BSBI in **Figure 3.8**. BSBI (i) segments the collection into parts of equal size, (ii) sorts the termID-docID pairs of each part in memory, (iii) stores intermediate sorted results on disk, and (iv) merges all intermediate results into the final index.

The algorithm parses documents into termID-docID pairs and accumulates the pairs in memory until a block of a fixed size is full (PARSENEXTBLOCK in **Figure 3.8**). We choose the block size to fit comfortably into memory to permit a fast in-memory sort. The block is then inverted and written to disk. Inversion involves two steps. First, we sort the termID-docID pairs. Next, we collect all termID-docID pairs with the same termID into a postings list, where a posting is simply a docID. The result, an inverted index for the block we have just read, is then written to disk. Applying this to Reuters Corpus Volume I (RCV1) and assuming we can fit 10 million termID-docID pairs into memory, we end up with ten blocks, each an inverted index of one part of the collection.

In the final step, the algorithm simultaneously merges the 10 blocks into 1 large merged index. An example with two blocks is shown in **Figure 3.9**, where we use d_i to denote the i_{th} document of the collection. To do the merging, we open all block files simultaneously and maintain small read buffers for the ten blocks we are reading and a write buffer for the final merged index we are writing. In each iteration, we select the lowest termID that has not been processed yet using a priority queue or a similar data structure. All postings lists for this termID are read and merged, and the merged list is written back to disk. Each read buffer is refilled from its file when necessary.

How expensive is BSBI? Its time complexity is $\Theta(T \log T)$ because the step with the highest time complexity is sorting and T is an upper bound for the number of items we must sort (i.e., the number of termID-docID pairs). But the actual indexing time is usually dominated by the time it takes to parse the documents (PARSENEXTBLOCK) and to do the final merge (MERGEBLOCKS). Exercise 4.6 asks you to compute the total index construction time for RCV1 that includes these steps as well as invert the blocks and write them to disk (**Figure 3.9**).

```

BSBIINDEXCONSTRUCTION()
1   $n \leftarrow 0$ 
2  while (all documents have not been processed)
3  do  $n \leftarrow n + 1$ 
4      $block \leftarrow$  PARSENEXTBLOCK()
5     BSBI-INVERT( $block$ )
6     WRITEBLOCKTODISK( $block, f_n$ )
7  MERGEBLOCKS( $f_1, \dots, f_n; f_{merged}$ )

```

Figure 3.8.

Blocked sort-based indexing. The algorithm stores inverted blocks in files f_1, \dots, f_n and the merged index in f_{merged} .

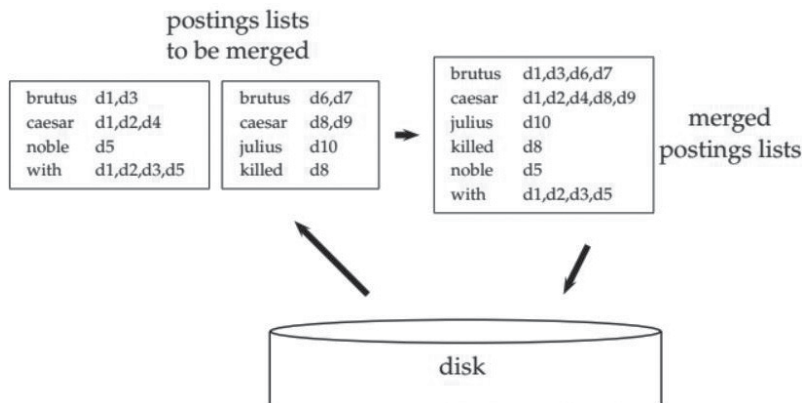


Figure 3.9.

Merging in blocked sort-based indexing. Two blocks (postings lists to be merged) are loaded from disk into memory, merged in memory (merged postings lists), and written back to disk. We show terms instead of termIDs for better readability.

Notice that Reuters Corpus Volume I (RCV1) is not particularly large in an age when one or more GB of memory is standard on personal computers. With appropriate compression, we could have created an inverted index for RCV1 in memory on a not overly beefy server. The techniques we have described are needed, however, for collections that have several orders of larger magnitude.

3.6.3 Single-pass in-memory indexing

Blocked sort-based indexing has excellent scaling properties, but it needs a data structure for mapping terms to termIDs. For very large collections, this data structure does not fit into memory. A more scalable alternative is single-pass in-memory indexing or SPIMI. SPIMI uses terms instead of termIDs, writes each block's dictionary to disk, and then starts a new dictionary for the next block. SPIMI can index collections of any size as long as there is enough disk space available.

The SPIMI algorithm is shown in **Figure 3.10**. The part of the algorithm that parses documents and turns them into a stream of termdocID pairs, which we call tokens here, has been omitted. SPIMI-INVERT is called repeatedly on the token stream until the entire collection has been processed. Tokens are processed one by

```

SPIMI-INVERT(token_stream)
1  output_file = NEWFILE()
2  dictionary = NEWHASH()
3  while (free memory available)
4  do token ← next(token_stream)
5     if term(token) ∉ dictionary
6         then postings_list = ADDTODICTIONARY(dictionary, term(token))
7         else postings_list = GETPOSTINGSLIST(dictionary, term(token))
8     if full(postings_list)
9         then postings_list = DOUBLEPOSTINGSLIST(dictionary, term(token))
10    ADDTODICTIONARY(postings_list, docID(token))
11 sorted_terms ← SORTTERMS(dictionary)
12 WRITEBLOCKTODISK(sorted_terms, dictionary, output_file)
13 return output_file

```

Figure 3.10.

Inversion of a block in single-pass in-memory indexing.

one (line 4) during each successive call of SPIMI-INVERT. When a term occurs for the first time, it is added to the dictionary (best implemented as a hash), and a new postings list is created (line 6). The call in line 7 returns this postings list for subsequent occurrences of the term.

A difference between BSBI and SPIMI is that SPIMI adds a posting directly to its postings list (line 10). Instead of first collecting all termID-docID pairs and then sorting them (as we did in BSBI), each postings list is dynamic (i.e., its size is adjusted as it grows), and it is immediately available to collect postings. This has two advantages: it is faster because there is no sorting required, and it saves memory because we keep track of the term a postings list belongs to, so the termIDs of postings need not be stored. As a result, the blocks that individual calls of SPIMI-INVERT can process are much larger, and the index construction process as a whole is more efficient.

Because we do not know how large the postings list of a term will be when we first encounter it, we allocate space for a short postings list initially and double the space each time it is full (lines 8–9). This means that some memory is wasted, which counteracts the memory savings from the omission of termIDs in intermediate data structures. However, the overall memory requirements for the dynamically constructed index of a block in SPIMI are still lower than in BSBI.

When memory has been exhausted, we write the index of the block (which consists of the dictionary and the postings lists) to disk (line 12). We have to sort the terms (line 11) before doing this because we want to write postings lists in lexicographic order to facilitate the final merging step. If each block's postings lists were written in unsorted order, merging blocks could not be accomplished by a simple linear scan through each block.

Each call of SPIMI-INVERT writes a block to disk, just as in BSBI. The last step of SPIMI (corresponding to line 7 in **Figure 3.8**; not shown in **Figure 3.10**) is then to merge the blocks into the final inverted index.

In addition to constructing a new dictionary structure for each block and eliminating the expensive sorting step, SPIMI has a third important component: compression. Both the postings and the dictionary terms can be stored compactly on disk if we employ compression. Compression increases the efficiency of the algorithm further because we can process even larger blocks and because the individual blocks require less space on disk. We refer readers to the literature for this aspect of the algorithm.

The time complexity of SPIMI is $\Theta(T)$ because no sorting of tokens is required and all operations are at most linear in the size of the collection.

3.6.4 Dictionary compression

This section presents a series of dictionary data structures that achieve increasingly higher compression ratios. The dictionary is small compared with the postings file as suggested by **Table 5.1**. So, why compress it if it is responsible for only a small percentage of the overall space requirements of the IR system?

One of the primary factors in determining the response time of an IR system is the number of disk seeks necessary to process a query. If parts of the dictionary are on disk, then many more disk seeks are necessary in query evaluation. Thus, the main goal of compressing the dictionary is to fit it in the main memory, or at least a large portion of it, to support high query throughput. Although dictionaries of very large collections fit into the memory of a standard desktop machine, this is not true of many other application scenarios. For example, an enterprise search server for a large corporation may have to index a multiterabyte collection with a comparatively large vocabulary because of the presence of documents in many different languages.

term	document frequency	pointer to postings list
a	656,265	→
aachen	65	→
...
zulu	221	→

space needed: 20 bytes 4 bytes 4 bytes

Figure 3.11.
Storing the dictionary as an array of fixed-width entries.

We also want to be able to design search systems for limited hardware such as mobile phones and onboard computers. Other reasons for wanting to conserve memory are fast start-up time and having to share resources with other applications. The search system on your PC must get along with the memory-hogging word processing suite you are using at the same time (**Figure 3.11**).

3.7. XML retrieval

Information retrieval systems are often contrasted with relational databases. Traditionally, IR systems have retrieved information from *unstructured text* by which we mean raw text without markup. Databases are designed for querying *relational data*: sets of records that have values for predefined attributes such as employee number, title, and salary. There are fundamental differences between information retrieval and database systems in terms of retrieval model, data structures, and query languages shown in **Table 3.3**.

Some highly structured text search problems are most efficiently handled by a relational database (RDB), for example, if the employee table contains an attribute for short textual job descriptions and you want to find all employees who are involved with invoicing. In this case, the SQL query, *select lastname from employees where job_desc like invoic%*, may be sufficient to satisfy your information need with high precision and recall.

However, many structured data sources containing text are best modeled as structured documents rather than relational data. We call the search over such structured documents and structured retrieval. Queries in structured retrieval can be either structured or unstructured, but we will assume in this chapter that the collection consists only of structured documents. Applications of structured retrieval include digital libraries, patent databases, blogs, text in which entities like persons and locations have been tagged (in a process called named entity tagging), and output from office suites like OpenOffice that save documents as marked up text. In all of these applications, we want to be able to run queries that combine textual criteria with structural criteria. Examples of such queries give me a

	RDB search	Unstructured retrieval	Structured retrieval
Objects	Records	Unstructured documents	Trees with text at leaves
Model	Relational model	Vector space & others	?
Main data structure	Table	Inverted index	?
Queries	SQL	Free text queries	?

Table 3.3.
RDB (relational database) search, unstructured information retrieval, and structured information retrieval.

full-length article on fast Fourier transforms (digital libraries), give me patents whose claims mention RSA public-key encryption and that cite US 4,405,829 patents, or give me articles about sightseeing tours of the Vatican and the Coliseum (entity-tagged text). These three queries are structured queries that cannot be answered well by an unranked retrieval system. As we argued in Example 1.1 (p. 15), unranked retrieval models like the Boolean model suffer from low recall. For instance, an unranked system would return a potentially large number of articles that mention the Vatican, the Coliseum, and the sightseeing tours without ranking the ones that are most relevant for the query first. Most users are also notoriously bad at precisely stating structural constraints. For instance, users may not know for which structured elements the search system supports search. In our example, the user may be unsure whether to issue the query as sightseeing AND (COUNTRY:Vatican OR LANDMARK:Coliseum), as sightseeing AND (STATE:Vatican OR BUILDING:Coliseum), or in some other form. Users may also be completely unfamiliar with structured search and advanced search interfaces or unwilling to use them. In this chapter, we look at how ranked retrieval methods can be adapted to structured documents to address these problems.

There is no consensus yet as to which methods work best for structured retrieval although many researchers believe that XQuery will become the standard for structured queries.

We will only look at one standard for encoding structured documents, Extensible Markup Language or XML, which is currently the most widely used such standard. We will not cover the specifics that distinguish XML from other types of markup such as HTML and SGML. But most of what we say in this chapter is applicable to markup languages in general.

In the context of information retrieval, we are only interested in XML as a language for encoding text and documents. A perhaps more widespread use of XML is to encode non-text data. For example, we may want to export data in XML format from an enterprise resource planning system and then read them into an analytic program to produce graphs for a presentation. This type of application of XML is called data-centric because numerical and non-text attribute-value data dominate and text is usually a small fraction of the overall data. Most data-centric XML is stored in databases—in contrast to the inverted index-based methods for text-centric XML that we present in this chapter.

We call XML retrieval a structured retrieval in this chapter. Some researchers prefer the term semistructured retrieval to distinguish XML retrieval from database querying. We have adopted the terminology that is widespread in the XML retrieval community. For instance, the standard way of referring to XML queries is structured queries, not semistructured queries. The term structured retrieval is rarely used for database querying, and it always refers to XML retrieval in this book.

There is a second type of information retrieval problem that is intermediate between unstructured retrieval and querying a relational database, parametric and zone search, which we discussed in Section 6.1 (p. 110). In the data model of parametric and zone search, there are parametric fields (relational attributes like date or file size) and zones “C text attributes that each takes a chunk of unstructured text as value. The data model is flat, that is, there is no nesting of attributes. The number of attributes is small. In contrast, XML documents have the more complex tree structure that we see in **Figure 3.13** in which attributes are nested. The number of attributes and nodes is greater than in parametric and zone search.

After presenting the basic concepts of XML in Section 3.71, this chapter first discusses the challenges we face in XML retrieval (Section 3.72). Next, we describe a vector space model for XML retrieval (Section 3.73). Presents INEX, a shared task

evaluation that has been held for a number of years and currently is the most important venue for XML retrieval research.

3.7.1 Basic XML concepts

An XML document is an ordered, labeled tree. Each node of the tree is an XML element and is written with an opening and closing tag. An element can have one or more XML attributes. In the XML document in **Figure 3.12**, the scene element is enclosed by the two tags `<scene...>` and `</scene>`. It has an attribute number with value `vii` and two child elements, title and verse.

Figure 3.13 shows **Figure 3.12** as a tree. The leaf nodes of the tree consist of text, for example, Shakespeare, Macbeth, and Macbeth's castle. The tree's internal nodes encode either the structure of the document (title, act, and scene) or metadata functions (author).

The standard for accessing and processing XML documents is the XML Document Object Model or DOM. The DOM represents elements, attributes, and text within elements as nodes in a tree. **Figure 3.13** is a simplified DOM representation of the XML document in **Figure 3.12**. With a DOM API, we can process an XML document by starting at the root element and then descending down the tree from parents to children.

XPath is a standard for enumerating paths in an XML document collection. We will also refer to paths as XML contexts or simply contexts in this chapter. Only a small subset of XPath is needed for our purposes. The XPath expression `node` selects all nodes of that name. Successive elements of a path are separated by slashes, so `act/scene` selects all scene elements whose parent is an act element. Double slashes indicate that an arbitrary number of elements can intervene on a path: `play//scene` selects all scene elements occurring in a play element. In **Figure 3.13**, this set consists of a single scene element, which is accessible via the path `play, act, and scene` from the top. An initial slash starts the path at the root element. `/play/title` selects the plays title in **Figure 3.12**, `/play//title` selects a set with two members (the plays title and the scenes title), and `/scene/title` selects no elements. For notational convenience, we allow the final element of a path to be a vocabulary term and separate it from the element path by the symbol `#`, even though this does not conform to the XPath standard. For example, `title#“Macbeth”` selects all titles containing the term Macbeth.

We also need the concept of schema in this chapter. A schema puts constraints on the structure of allowable XML documents for a particular application. A schema for Shakespeare's plays may stipulate that scenes can only occur as children of acts and that only acts and scenes have the number attribute. Two standards for schemas for XML documents are XML DTD (document-type definition) and XML schema.

```
<play>
  <author>Shakespeare</author>
  <title>Macbeth</title>
  <act number="I">
    <scene number="vii">
      <title>Macbeth's castle</title>
      <verse>Will I with wine and wassail ...</verse>
    </scene>
  </act>
</play>
```

Figure 3.12.
An XML document.

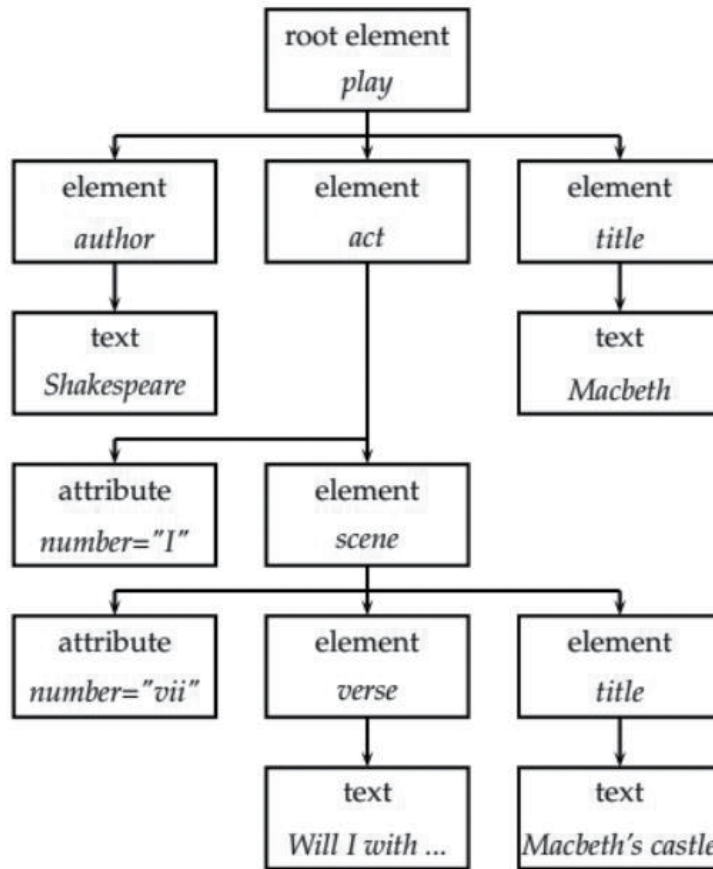


Figure 3.13.
 The XML document in Figure 10.1 as a simplified DOM object.

Users can only write structured queries for an XML retrieval system if they have some minimal knowledge about the schema of the collection

A common format for XML queries is NEXI (Narrowed Extended XPath I). We give an example in **Figure 3.14**. We display the query on four lines for typographical convenience, but it is intended to be read as one unit without line breaks. In particular, `//section` is embedded under `//article`.

The query in **Figure 3.14** specifies a search for sections about the summer holidays that are part of articles from 2001 to 2002. As in XPath double slashes indicate that an arbitrary number of elements can intervene on a path. The dot in a clause in square brackets refers to the element the clause modifies. The clause

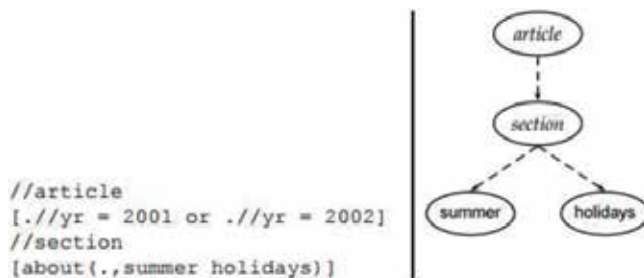


Figure 3.14.
 An XML query in NEXI format and its partial representation as a tree.

[./yr = 2001 or./yr = 2002] modifies //article. Thus, the dot refers to //article in this case. Similarly, the dot in [about(., summer holidays)] refers to the section that the clause modifies.

The 2-year conditions are relational attribute constraints. Only articles whose year attribute is 2001 or 2002 (or that contain an element whose year attribute is 2001 or 2002) are to be considered. The about clause is a ranking constraint: sections that occur in the right type of article are to be ranked according to how relevant they are to the topic summer holidays.

We usually handle relational attribute constraints by prefiltering or postfiltering: we simply exclude all elements from the result set that do not meet the relational attribute constraints. In this chapter, we will not address how to do this efficiently and instead focus on the core information retrieval problem in XML retrieval, namely, how to rank documents according to the relevance criteria expressed in the about conditions of the NEXI query.

If we discard relational attributes, we can represent documents as trees with only one type of node: element nodes. In other words, we remove all attribute nodes from the XML document, such as the number attribute in **Figure 3.12**. **Figure 3.15** shows a subtree of the document in **Figure 3.12** as an element-node tree (labeled d_1).

We can represent queries as trees in the same way. This is a query-by-example approach to query language design because users pose queries by creating objects that satisfy the same formal description as documents. In **Figure 3.15**, q_1 is a search for books whose titles score highly for the keywords Julius Caesar. q_2 is a search for books whose author elements score highly for Julius Caesar and whose title elements score highly for Gallic war. To represent the semantics of NEXI queries fully we would also need to designate one node in the tree as a “target node”, for example, the section in the tree in **Figure 3.14**. Without the designation of a target node, the tree in **Figure 3.14** is not a search for sections embedded in articles (as specified by NEXI), but a search for articles that contain sections.

3.7.2 Challenges in XML retrieval

In this section, we discuss a number of challenges that make structured retrieval more difficult than unstructured retrieval. Recall from page 195 the basic setting we assume in structured retrieval: the collection consists of structured documents, and queries are either structured (as in **Figure 3.14**) or unstructured (e.g., summer holidays).

The first challenge in structured retrieval is that users want us to return parts of documents (i.e., XML elements), not the entire documents as IR systems usually do in unstructured retrieval. If we query Shakespeare’s plays for Macbeth’s castle, should we return the scene, the act, or the entire play in **Figure 3.13**? In this case,

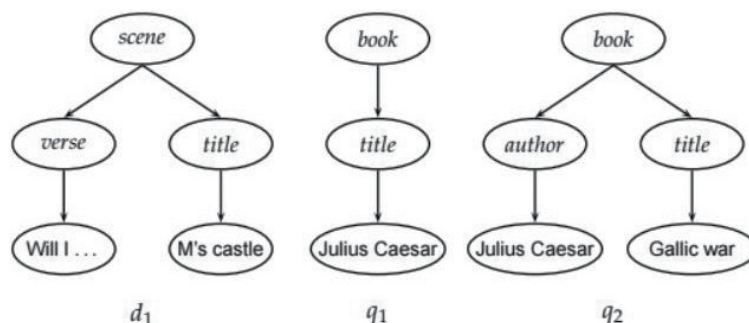


Figure 3.15.
Tree representation of XML documents and queries.

the user is probably looking for the scene. On the other hand, another wise unspecified search for Macbeth should return the play of this name, not a subunit.

One criterion for selecting the most appropriate part of a document is the *structured document retrieval principle*:

Structured document retrieval principle. A system should always retrieve the most specific part of a document answering the query.

This principle motivates a retrieval strategy that returns the smallest unit that contains the information sought, but does not go below this level. However, it can be hard to implement this principle algorithmically. Consider the query title#“Macbeth” applied to **Figure 3.13**. The title of the tragedy, Macbeth, and the title of act I, scene vii, Macbeth’s castle, are both good hits because they contain the matching term Macbeth. But in this case, the title of the tragedy, the higher node, is preferred. Deciding which level of the tree is right for answering a query is difficult.

Parallel to the issue of which parts of a document to return to the user is the issue of which parts of a document to index. In Section 2.1.2 (p. 20), we discussed the need for a document unit or indexing unit in indexing and retrieval. In unstructured retrieval, it is usually clear what the right document unit is: files on your desktop, email messages, web pages on the web, etc. In structured retrieval, there are a number of different approaches to defining the indexing unit.

One approach is to group nodes into non-overlapping pseudo-documents as shown in **Figure 3.16**. In the example, books, chapters, and sections have been designated to be indexing units but without overlap. For example, the leftmost dashed indexing unit contains only those parts of the tree dominated by book that are not already part of other indexing units. The disadvantage of this approach is that pseudo-documents may not make sense to the user because they are not coherent units. For instance, the leftmost indexing unit in **Figure 3.16** merges three disparate elements, the class, author, and title elements.

We can also use one of the largest elements as the indexing unit, for example, the book element in a collection of books or the play element for Shakespeare’s works. We can then postprocess search results to find for each book or play the subelement that is the best hit. For example, the query Macbeth’s castle may return the play Macbeth, which we can then postprocess to identify act I, scene vii, as the best-matching subelement. Unfortunately, this two-stage retrieval process fails to return the best subelement for many queries because the relevance of a whole book is often not a good predictor of the relevance of small subelements within it.

Instead of retrieving large units and identifying subelements (top-down), we can also search all leaves, select the most relevant ones, and then extend them to larger units in postprocessing (bottom-up). For the query Macbeth’s castle in **Figure 3.12**, we would retrieve the title Macbeth’s castle in the first pass and then decide in a postprocessing step whether to return the title, the scene, the actor, or the play. This approach has a similar problem as the last one: the relevance of a leaf element is often not a good predictor of the relevance of elements it is contained in.

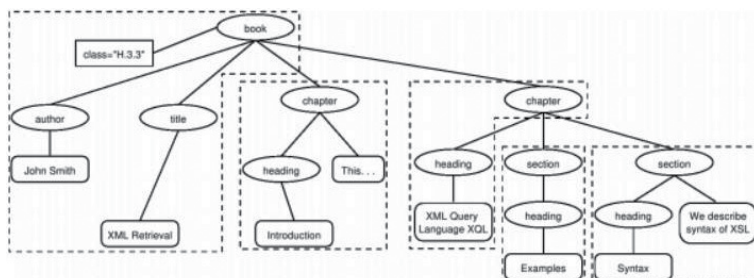


Figure 3.16.
Partitioning an XML document into non-overlapping indexing units.

The least restrictive approach is to index all elements. This is also problematic. Many XML elements are not meaningful search results, for example, typographical elements like *definitely* or an ISBN number which cannot be interpreted without context. Also, indexing all elements means that search results will be highly redundant. For the query Macbeth's castle and the document in **Figure 3.12**, we would return all of the play, act, scene, and title elements on the path between the root node and Macbeth's castle. The leaf node would then occur four times in the result set, once directly, and three times as part of other elements. We call elements that are contained within each other nested. Returning redundant nested elements in a list of returned hits is not very user-friendly.

Because of the redundancy caused by nested elements, it is common to restrict the set of elements that are eligible to be returned. Restriction strategies include:

- Discard all small elements.
- Discard all element types that users do not look at (this requires a working XML retrieval system that logs this information).
- Discard all element types that assessors generally do not judge to be relevant (if relevance assessments are available).
- Only keep element types that a system designer or librarian has deemed to be useful search results.

In most of these approaches, result sets will still contain nested elements. Thus, we may want to remove some elements in a postprocessing step to reduce redundancy. Alternatively, we can collapse several nested elements in the results list and use highlighting of query terms to draw the user's attention to the relevant passages. If query terms are highlighted, then scanning a medium-sized element (e.g., a section) takes little more time than scanning a small subelement (e.g., a paragraph). Thus, if the section and the paragraph both occur in the results list, it is sufficient to show the section. An additional advantage of this approach is that the paragraph is presented together with its context (i.e., the embedding section). This context may be helpful in interpreting the paragraph (e.g., the source of the information reported) even if the paragraph on its own satisfies the query.

If the user knows the schema of the collection and is able to specify the desired type of element, then the problem of redundancy is alleviated as few nested elements have the same type. But as we discussed in the introduction, users often don't know what the name of an element in the collection is (Is the Vatican a country or a city?), or they may not know how to compose structured queries at all.

A challenge in XML retrieval related to nesting is that we may need to distinguish different contexts of a term when we compute term statistics for ranking, in particular inverse document frequency (idf) statistics as defined in Section 6.2.1 (p. 117). For example, the term Gates under the node author is unrelated to an occurrence under a content node like section if used to refer to the plural of gate. It makes little sense to compute a single document frequency for Gates in this example.

One solution is to compute idf for XML-context/term pairs, for example, to compute different idf weights for author#"Gates" and section#"Gates." Unfortunately, this scheme will run into sparse data problems; that is, many XML-context pairs occur too rarely to reliably estimate df (see Section 13.2, p. 260, for a discussion of sparseness). A compromise is only to consider the parent node x of the term and not the rest of the path from the root to x to distinguish contexts. There are still conflation of contexts that are harmful in this scheme. For instance, we do not

distinguish names of authors and names of corporations if both have the parent node name. But most important distinctions, like the example contrast author#“Gates” vs. section#“Gates,” will be respected.

In many cases, several different XML schemas occur in a collection since the XML documents in an IR application often come from more than one source. This phenomenon is called schema heterogeneity or schema diversity and presents yet another challenge. As illustrated in **Figure 3.17**, comparable elements may have different names: creator in d_2 vs. author in d_3 . In other cases, the structural organization of the schemas may be different: author names are direct descendants of the node author in q_3 , but there are the intervening nodes *firstname* and *lastname* in d_3 . If we employ strict matching of trees, then q_3 will retrieve neither d_2 nor d_3 although both documents are relevant. Some form of approximate matching of element names in combination with semiautomatic matching of different document structures can help here. Human editing of correspondences of elements in different schemas will usually do better than automatic methods.

Schema heterogeneity is one reason for query-document mismatches like q_3/d_2 and q_3/d_3 . Another reason is that users often are not familiar with the element names and the structure of the schemas of collections they search as mentioned. This poses a challenge for interface design in XML retrieval. Ideally, the user interface should expose the tree structure of the collection and allow users to specify the elements they are querying. If we take this approach, then designing the query interface in structured retrieval is more complex than a search box for keyword queries in unstructured retrieval.

We can also support the user by interpreting all parent-child relationships in queries as descendant relationships with any number of intervening nodes allowed. We call such queries extended queries. The tree in **Figure 3.14** and q_4 in **Figure 3.17** are examples of extended queries. We show edges that are interpreted as descendant relationships as dashed arrows. In q_4 , a dashed arrow connects *book* and *Gates*. As a pseudo-XPath notation for q_4 , we adopt *book//#“Gates”*: a book that somewhere in its structure contains the word *Gates* where the path from the *book* node to *Gates* can be arbitrarily long. The pseudo-XPath notation for the extended query that in addition specifies that *Gates* occurs in a section of the book is *book//section//#“Gates.”* It is convenient for users to be able to issue such extended queries without having to specify the exact structural configuration in which a query term should occur—either because they do not care about the exact configuration or because they do not know enough about the schema of the collection to be able to specify it.

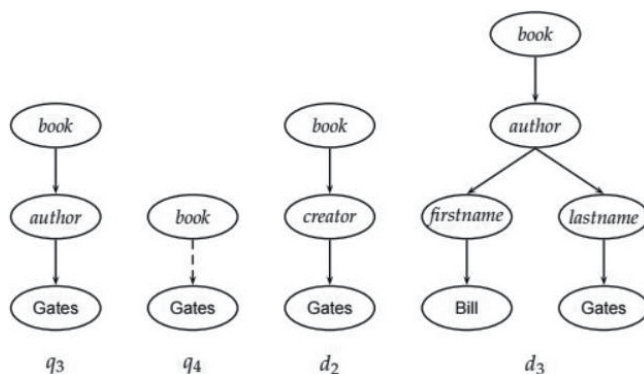


Figure 3.17.
Schema heterogeneity: intervening nodes and mismatched names.

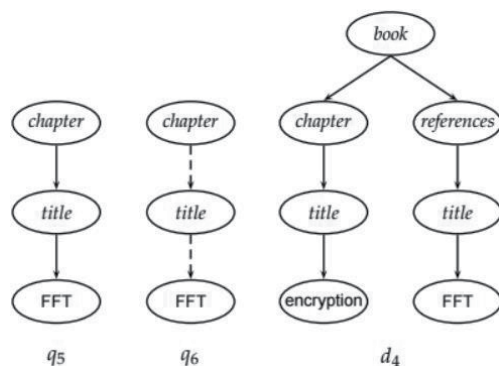


Figure 3.18.
A structural mismatch between two queries and a document.

In **Figure 3.18**, the user is looking for a chapter entitled FFT (q_5). Suppose that there is no such chapter in the collection but that there are references to books on FFT (d_4). A reference to a book on FFT is not exactly what the user is looking for, but it is better than returning nothing. Extended queries do not help here. The extended query q_6 also returns nothing. This is a case where we may want to interpret the structural constraints specified in the query as hints as opposed to as strict conditions. As we will discuss in Section 10.4, users prefer a relaxed interpretation of structural constraints: elements that do not meet structural constraints perfectly should be ranked lower, but they should not be omitted from search results.

3.7.3 A vector space model for XML retrieval

In this section, we present a simple vector space model for XML retrieval. It is not intended to be a complete description of a state-of-the-art system. Instead, we want to give the reader a flavor of how documents can be represented and retrieved in XML retrieval.

To take into account the structure in retrieval in **Figure 3.15**, we want a book entitled Julius Caesar to be a match for q_1 and no match (or a lower-weighted match) for q_2 . In unstructured retrieval, there would be a single dimension of the vector space for Caesar. In XML retrieval, we must separate the title word Caesar from the author name Caesar. One way of doing this is to have each dimension of the vector space encode a word together with its position within the XML tree.

Figure 3.19 illustrates this representation. We first take each text node (which in our setup is always a leaf) and break it into multiple nodes, one for each word. So, the leaf node Bill Gates is split into two leaves Bill and Gates. Next, we define the dimensions of the vector space to be lexicalized subtrees of documents—subtrees that contain at least one vocabulary term. A subset of these possible lexicalized subtrees is shown in the figure, but there are others, for example, the subtree corresponding to the whole document with the leaf node Gates removed. We can now represent queries and documents as vectors in this space of lexicalized subtrees and compute matches between them. This means that we can use the vector space formalism from Chapter 6 for XML retrieval. The main difference is that the dimensions of vector space in unstructured retrieval are vocabulary terms, whereas they are lexicalized subtrees in XML retrieval.

There is a trade-off between the dimensionality of the space and accuracy of query results. If we trivially restrict dimensions to vocabulary terms, then we have a standard vector space retrieval system that will retrieve many documents that do

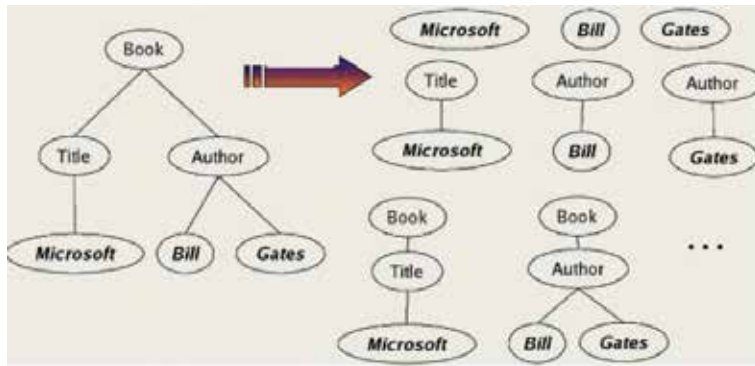


Figure 3.19.
 A mapping of an XML document (left) to a set of lexicalized subtrees (right).

not match the structure of the query (e.g., Gates in the title as opposed to the author element). If we create a separate dimension for each lexicalized subtree occurring in the collection, the dimensionality of the space becomes too large. A compromise is to index all paths that end in a single vocabulary term, in other words, all XML-context/term pairs. We call such an XML-context/term pair a structural term and denote it by $\langle c, t \rangle$: a pair of XML-context c and vocabulary term t . The document in **Figure 3.19** has nine structural terms. Seven are shown (e.g., “Bill” and Author#“Bill”) and two are not shown: /Book/Author#“Bill” and /Book/Author#“Gates.” The tree with the leaves Bill and Gates is a lexicalized subtree that is not a structural term. We use the previously introduced pseudo-XPath notation for structural terms.

As we discussed in the last section, users are bad at remembering details about the schema and at constructing queries that comply with the schema. We will therefore interpret all queries as extended queries; that is, there can be an arbitrary number of intervening nodes in the document for any parent-child node pair in the query. For example, we interpret q_5 in **Figure 3.18** as q_6 .

But we still prefer documents that match the query structure closely by inserting fewer additional nodes. We ensure that retrieval results respect this preference by computing a weight for each match. A simple measure of the similarity of a path c_q in a query and a path c_d in a document is the following context resemblance function C_R :

$$C_R(c_q, c_d) = \begin{cases} \frac{1+|c_q|}{1+|c_d|} & \text{if } c_q \text{ matches } c_d \\ 0 & \text{if } c_q \text{ does not match } c_d \end{cases}, \quad (3.21)$$

where $|c_q|$ and $|c_d|$ are the number of nodes in the query path and document path, respectively, and c_q matches c_d if we can transform c_q into c_d by inserting additional nodes. Two examples from **Figure 3.17** are $C_R(c_{q4}, c_{d2}) = 3/4 = 0.75$ and $C_R(c_{q4}, c_{d3}) = 3/5 = 0.6$ where c_{q4} , c_{d2} and c_{d3} are the relevant paths from top to leaf node in q_4 , d_2 and d_3 , respectively. The value of $C_R(c_q, c_d)$ is 1.0 if q and d are identical.

The final score for a document is computed as a variant of the cosine measure [Eq. (6.10), p. 121], which we call SIMNOMERGE for reasons that will become clear shortly. SIMNOMERGE is defined as follows:

$$SIMNOMERGE(q, d) = \frac{\sum_{c_k \in B_{c_l} \in B} C_R(c_k, c_l) \sum_{t \in V} weight(q, t, c_k) \frac{weight(d, t, c_l)}{\sqrt{\sum_{c \in B, t \in V} weight^2(d, t, c)}}}{\sqrt{\sum_{c \in B, t \in V} weight^2(d, t, c)}}. \quad (3.22)$$

where V is the vocabulary of nonstructural terms, B is the set of all XML contexts, and $weight(q, t, c)$ and $weight(d, t, c)$ are the weights of term t in XML context c in query q and document d , respectively. We compute the weights using one of the weightings from Chapter 6, such as $idf_t \cdot wf_{t,d}$. The inverse document frequency idf_t depends on which elements we use to compute df_t as discussed in Section 10.2. The similarity measure $SIMNOMERGE(q, d)$ is not a true cosine measure since its value can be larger than 1.0 (Exercise 10.11). We divide by $\sqrt{\sum_{c \in B, t \in V} weight^2(d, t, c)}$ to normalize for document length (Section 6.3.1, page 121). We have omitted query length normalization to simplify the formula. It has no effect on ranking since, for a given query, the normalizer $\sqrt{\sum_{c \in B, t \in V} weight^2(d, t, c)}$ is the same for all documents.

The algorithm for computing $SIMNOMERGE$ for all documents in the collection is shown in **Figure 3.22**. The array normalizer in **Figure 3.22** contains

$\sqrt{\sum_{c \in B, t \in V} weight^2(d, t, c)}$ from Eq. (3.22) for each document.

We give an example of how $SIMNOMERGE$ computes query-document similarities in **Figure 3.21**. $\langle c_1, t \rangle$ is one of the structural terms in the query. We successively retrieve all postings lists for structural terms $\langle c', t \rangle$ with the same vocabulary term t . Three example postings lists are shown. For the first one, we have $C_R(c_1, c_1) = 1.0$ since the two contexts are identical. The next context has no context resemblance with c_1 : $C_R(c_1, c_2) = 0$, and the corresponding postings list is ignored. The context match of c_1 with c_3 is $0.63 \geq 0$ and it will be processed. In this example, the highest ranking document is d_9 with a similarity of $1.0 \times 0.2 + 0.63 \times 0.6 = 0.578$. To simplify the figure, the query weight of $\langle c_1, t \rangle$ is assumed to be 1.0 (**Figures 3.20** and **3.21**).

The query-document similarity function in **Figure 3.22** is called $SIMNOMERGE$ because different XML contexts are kept separate for the purpose of weighting. An alternative similarity function is $SIMMERGE$ which relaxes the matching conditions of query and document further in the following three ways:

- We collect the statistics used for computing $weight(q, t, c)$ and $weight(d, t, c)$ from all contexts that have a non-zero resemblance to c (as opposed to just from c as in $SIMNOMERGE$). For instance, for computing the document frequency of the structural term `atl#“recognition,”` we also count occurrences of recognition in XML contexts `fm/atl, article//atl, etc.`
- We modify Eq. (3.22) by merging all structural terms in the document that have a non-zero context resemblance to a given query structural term. For

```

SCOREDOCUMENTSWITHSIMNOMERGE( $q, B, V, N, normalizer$ )
1  for  $n \leftarrow 1$  to  $N$ 
2  do  $score[n] \leftarrow 0$ 
3  for each  $\langle c_q, t \rangle \in q$ 
4  do  $w_q \leftarrow WEIGHT(q, t, c_q)$ 
5     for each  $c \in B$ 
6     do if  $CR(c_q, c) > 0$ 
7         then  $postings \leftarrow GETPOSTINGS(\langle c, t \rangle)$ 
8             for each  $posting \in postings$ 
9                 do  $x \leftarrow CR(c_q, c) * w_q * weight(posting)$ 
10                     $score[docID(posting)] += x$ 
11 for  $n \leftarrow 1$  to  $N$ 
12 do  $score[n] \leftarrow score[n] / normalizer[n]$ 
13 return  $score$ 

```

Figure 3.20.
The algorithm for scoring documents with $SIMNOMERGE$.

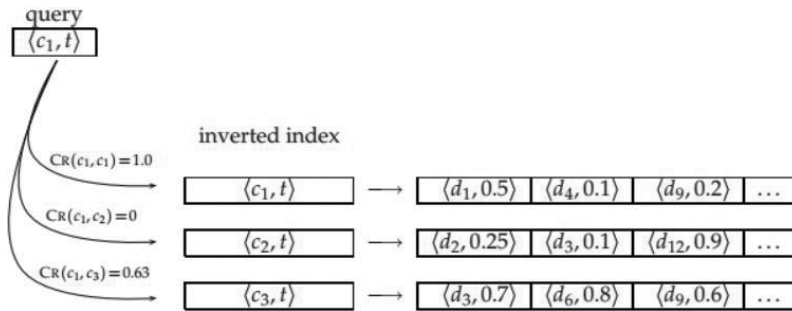


Figure 3.21.
 Scoring of a query with one structural term in SIMNOMERGE.

example, the contexts `/play/act/scene/title` and `/play/title` in the document will be merged when matching against the query term `/play/title#“Macbeth.”`

- The context resemblance function is further relaxed: contexts have a non-zero resemblance in many cases where the definition of C_R in Eq. (3.21) returns 0.

These three changes alleviate the problem of sparse term statistics discussed in Section 3.2 and increase the robustness of the matching function against poorly posed structural queries. The evaluation of SIMNOMERGE and SIMMERGE in the next section shows that the relaxed matching conditions of SIMMERGE increase the effectiveness of XML retrieval.

3.7.4 Evaluation of XML retrieval

The premier venue for research on XML retrieval is the INEX (INitiative for the Evaluation of XML retrieval) program, a collaborative effort that has produced reference collections, sets of queries, and relevance judgments. A yearly INEX meeting is held to present and discuss research results. The INEX 2002 collection consisted of about 12,000 articles from IEEE journals. We give collection statistics in **Table 3.4** and show part of the schema of the collection in **Figure 3.22**. The IEEE journal collection was expanded in 2005. Since 2006 INEX uses the much larger English Wikipedia as a test collection.

Two types of information needs or topics in INEX are content-only or CO topics and content-and-structure (CAS) topics. CO topics are regular keyword queries as in unstructured information retrieval. CAS topics have structural constraints in addition to keywords. We already encountered an example of a CAS topic in

12,107	Number of documents
494 Mb	Size
1995–2002	Time of publication of articles
1532	Average number of XML nodes per document
6.9	Average depth of a node
30	Number of CAS topics
30	Number of CO topics

Table 3.4.
 INEX 2002 collection statistics.

```

SCOREDOCUMENTSWITHSIMNOMERGE( $q, B, V, N, normalizer$ )
1 for  $n \leftarrow 1$  to  $N$ 
2 do  $score[n] \leftarrow 0$ 
3 for each  $\langle c_q, t \rangle \in q$ 
4 do  $w_q \leftarrow WEIGHT(q, t, c_q)$ 
5   for each  $c \in B$ 
6     do if  $CR(c_q, c) > 0$ 
7       then  $postings \leftarrow GETPOSTINGS(\langle c, t \rangle)$ 
8         for each  $posting \in postings$ 
9           do  $x \leftarrow CR(c_q, c) * w_q * weight(posting)$ 
10             $score[docID(posting)] += x$ 
11 for  $n \leftarrow 1$  to  $N$ 
12 do  $score[n] \leftarrow score[n] / normalizer[n]$ 
13 return  $score$ 

```

Figure 3.22.

Simplified schema of the documents in the INEX collection.

Figure 3.14. The keywords in this case are summer and holidays, and the structural constraints specify that the keywords occur in a section that in turn is part of an article and that this article has an embedded year attribute with value 2001 or 2002.

Since CAS queries have both structural and content criteria, relevance assessments are more complicated than in unstructured retrieval. INEX 2002 defined component coverage and topical relevance as orthogonal dimensions of relevance. The component coverage dimension evaluates whether the element retrieved is structurally correct, that is, neither too low nor too high in the tree. We distinguish four cases:

- Exact coverage (E). The information sought is the main topic of the component, and the component is a meaningful unit of information.
- Too small (S). The information sought is the main topic of the component, but the component is not a meaningful (self-contained) unit of information.
- Too large (L). The information sought is present in the component, but is not the main topic.
- No coverage (N). The information sought is not a topic of the component.

The topical relevance dimension also has four levels: highly relevant (3), fairly relevant (2), marginally relevant (1), and nonrelevant (0). Components are judged on both dimensions, and the judgments are then combined into a digit-letter code. 2S is a fairly relevant component that is too small, and 3E is a highly relevant component that has exact coverage. In theory, there are 16 combinations of coverage and relevance, but many cannot occur. For example, a nonrelevant component cannot have exact coverage, so the combination 3N is not possible.

The relevance-coverage combinations are quantized as follows:

$$Q(rel, cov) = \begin{cases} 1.00 & \text{if } (rel, cov) = 3E \\ 0.75 & \text{if } (rel, cov) \in \{2E, 3L\} \\ 0.50 & \text{if } (rel, cov) \in \{1E, 2L, 2S\} \\ 0.25 & \text{if } (rel, cov) \in \{1S, 1L\} \\ 0.00 & \text{if } (rel, cov) 0N \end{cases} \quad (3.23)$$

This evaluation scheme takes into account the fact that binary relevance judgments, which are standard in unstructured information retrieval, are not

appropriate for XML retrieval. A 2S component provides incomplete information and may be difficult to interpret without more context, but it does answer the query partially. The quantization function Q does not impose a binary choice relevant/nonrelevant and instead allows us to grade the component as partially relevant.

The number of relevant components in a retrieved set A of components can then be computed as

$$\#(\text{relevant items retrieved}) = \sum_{c \in A} Q(\text{rel}(c), \text{cov}(c)). \quad (3.24)$$

As an approximation, the standard definitions of precision, recall, and F from can be applied to this modified definition of relevant items retrieved, with some subtleties because we sum graded as opposed to binary relevance assessments.

One flaw of measuring relevance this way is that overlap is not accounted for. We discussed the concept of marginal relevance in the context of unstructured retrieval. This problem is worse in XML retrieval because of the problem of multiple nested elements occurring in a search result as we discussed on p. 80. Much of the recent focus at INEX has been on developing algorithms and evaluation measures that return non-redundant results lists and evaluate them properly.

Table 3.5 shows two INEX 2002 runs of the vector space system we described in Section 3.7.3. The better run is the SIMMERGE run, which incorporates few structural constraints and mostly relies on keyword matching. SIMMERGE median average precision (where the median is with respect to average precision numbers over topics) is only 0.147. Effectiveness in XML retrieval is often lower than in unstructured retrieval since XML retrieval is harder. Instead of just finding a document, we have to find the subpart of a document that is most relevant to the query. Also, XML retrieval effectiveness—when evaluated as described here—can be lower than unstructured retrieval effectiveness on a standard evaluation because graded judgments lower measured performance. Consider a system that returns a document with graded relevance 0.6 and binary relevance 1 at the top of the retrieved list. Then, interpolated precision at 0.00 recall is 1.0 on a binary evaluation but can be as low as 0.6 on a graded evaluation.

Table 3.5 gives us a sense of the typical performance of XML retrieval, but it does not compare structured with unstructured retrieval. **Table 3.6** directly shows the effect of using structure in retrieval. The results are for a language model-based system that is evaluated on a subset of CAS topics from INEX 2003 and 2004. The evaluation metric is precision at k . The discretization function used for the evaluation maps highly relevant elements (roughly corresponding to the 3E elements defined for Q) to 1 and all other elements to 0. The content only system treats queries and documents as unstructured bags of words. The full-structure model ranks elements that satisfy structural constraints higher than elements that do not. For instance, for the query in **Figure 3.14**, an element that contains the phrase summer holidays in a section will be rated higher than one that contains it in an abstract.

Algorithm	Average precision
SIMNOMERGE	0.242
SIMMERGE	0.271

Table 3.5.

INEX 2002 results of the vector space model in Section 3.7.3 for content and structure (CAS) queries and the quantization function Q .

		Content only	full structure	Improvement
precision at 5	0.2000	0.3265		63.3%
precision at 10	0.1820	0.2531		39.1%
precision at 20	0.1700	0.1796		5.6%
precision at 30	0.1527	0.1531		0.3%

Table 3.6.

The **Table 3.6** shows that structure helps increase precision at the top of the results list. There is a large increase of precision at $k = 5$ and at $k = 10$. There is almost no improvement at $k = 30$. These results demonstrate the benefits of structured retrieval. Structured retrieval imposes additional constraints on what to return and documents that pass the structural filter are more likely to be relevant. Recall may suffer because some relevant documents will be filtered out, but for precision-oriented tasks, structured retrieval is superior.

Classical Machine Learning

This chapter focuses on two major problems in machine learning: classification and clustering. Classification is based on some given sample of known class labels, training a learning machine (i.e., to obtain a certain objective function), so that it can classify unknown samples, and it belongs to supervised learning. However, clustering is the algorithm that does not know any kind of sample in advance, and it is desirable to classify a set of unknown categories of samples into several categories by some algorithm. When we use the clustering, we do not care about what kind it is. The goal that we need to achieve is just to bring together similar things, which in the machine learning is called unsupervised learning.

4.1. Classification

4.1.1 Performance evaluation for classification

When we use a classifier to predict, we will encounter a very important question: how to evaluate the predicted effect of this classifier. The evaluation of classifier performance is the basis for choosing excellent classifiers. Traditional classifier performance evaluation, such as accuracy, recall, sensitivity, and specificity, cannot fully consider a classifier. This book, based on the traditional classifier performance evaluation criteria, also referred to the confusion matrix, area under curve (*AUC*) and other methods of performance evaluation.

4.1.1.1 Performance evaluation of general classification

An instance can be divided into positive or negative. This will result in four classification results:

- True positive (*TP*): positive data that is correctly marked as positive data.
 - True negative (*TN*): negative data that is correctly marked as negative data.
 - False positive (*FP*): negative data that is incorrectly marked as positive data.
 - False negative (*FN*): positive data that is incorrectly marked as negative data.
- The formula for precision is

$$P = \frac{TP}{TP + FP} \quad (4.1)$$

The formula for recall is

$$P = \frac{TP}{TP + FN} \quad (4.2)$$

Sensitivity, also known as true positive rate (*TPR*), represents the proportion of positive instances identified by the classifier to all positive instances. The formula for sensitivity is

$$TPR = \frac{TP}{TP + FN} \quad (4.3)$$

Specificity, also known as negative rate (*FPR*), represents the proportion of negative instances identified into positive classes by the classifier. The formula for specificity is

$$FPR = \frac{FP}{FP + TN} \quad (4.4)$$

4.1.1.2 ROC curve and AUC

ROC curve is also known as the sensitivity curve. The ROC curve is based on a series of different binary classification (threshold or threshold), with *TP* as the ordinate and *FN* as the abscissa drawing points.

Area under curve (*AUC*) is defined as the area under the ROC curve, or it can be considered as the ratio of the area under the ROC curve to the unit area. It is clear that the area value is not greater than one. Also, since the ROC curve is generally above the straight line of $y = x$, the *AUC* value ranges from 0.5 to 1. The larger the *AUC* is, the higher the accuracy of the classification becomes.

4.1.1.3 Confusion matrix

Confusion matrix is used to sum up the supervision of the classification of the results. The items on the main diagonal indicate the total number of the correct categories, and the items of the other non-main diagonals indicate the number of errors in the classification. As shown in **Table 4.1**, there are two different types of errors: “false acceptance” and “false rejection” [100]. If the confusion matrix of the dichotomous problem is normalized, it is a joint distribution probability for discrete variables of 0 and 1 binary values. For the two classified issues, confusion matrix can be expressed in **Table 4.1**.

4.1.1.4 F-Score

F-Score, also known as F-Measure, is precision and recall weighted harmonic average and commonly used to evaluate if the classification model is good or bad. In the F-Score function, when the parameter $\alpha = 1$, F-Score combines the results of precision and recall; when F-Score is high, it can explain that the test method is more effective [101]. As the classification accuracy sometimes does not well highlight the characteristics of the sample set and determines the performance of a classifier, for the two classification problems, you can use the following two parameters to evaluate the performance of the classifier:

Category	ω_2	ϖ_2
ω_1	Accept correctly	Wrongly rejected
ϖ_1	Accept correctly	Wrongly rejected

ω_1 , the actual category one; ϖ_1 , the actual category two; ω_1 , classifier determines the category one; and ϖ_2 , classifier determines the category two.

Table 4.1.
The confusion matrix.

$$\begin{aligned} TNR &= \frac{TN}{TN + FP} = 1 - FPR \\ FNR &= \frac{FN}{TP + FN} = 1 - TPR \end{aligned} \quad (4.5)$$

It is generally believed that the higher the F-Score is, the better the classification effect of the classifier for the positive sample shows. It should be noted that *TNR* and *FNR* will affect each other. Therefore, the use of a separate parameter to evaluate the performance of the classifier cannot fully evaluate a classifier.

4.1.2 Decision tree

Decision tree is an important method in data mining classification algorithm. In a variety of classification algorithms, the decision tree is the most intuitional one. Decision tree is a method of decision analysis that evaluates the project risk and determines the feasibility. And it is a graphic method for probabilistic analysis of intuitive application. Because the decision branch is painted like a tree branch of a tree, we called it decision tree. In machine learning, the decision tree is a predictive model, which represents a mapping between attributes and values [102].

4.1.2.1 Model introduction

The classification decision tree model is a tree structure that describes the classification of instances. The decision tree consists of nodes and directed edges. There are two types of nodes: internal node and leaf node. An internal node represents a feature or attribute, and a leaf node represents a class [103]. A decision tree is used to classify a particular feature of the instance from the root node, and an instance is assigned to its child node according to the test result. At this time, each child node corresponds to a value of the feature. The instance is then recursively tested and assigned until the leaf node is finally assigned to the instance of the leaf node.

Figure 4.1 shows the decision tree, the circle, and the box, respectively, that represent the internal nodes and leaf nodes.

4.1.2.2 Feature selection

Feature selection is to select the characteristics of classification ability for training data, which can improve the efficiency of decision tree learning.

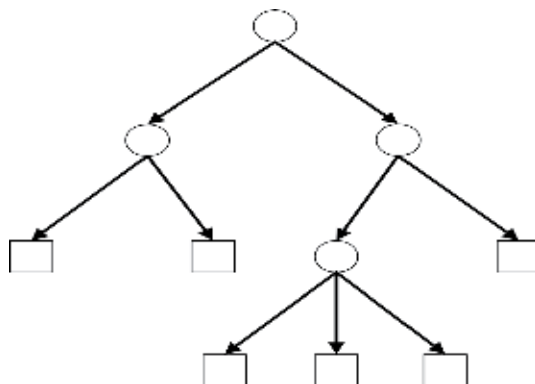


Figure 4.1.
Decision tree model.

The information gain can be used as one of the criteria for the selection of features [104]. First, we need to introduce the concept of entropy. In probability statistics, entropy is a measure of the uncertainty to random variables. Supposing X is a discrete random variable with a finite number of values, and its probability distribution is

$$P(X = x_i) \quad (4.6)$$

The entropy of the random variable X is defined as

$$H(X) = -\sum_{i=1}^n p_i \log p_i \quad (4.7)$$

With random variables (X, Y) , the joint probability distribution is

$$P(X = x_i, y_j) = p_{ij}, i = 1, 2, \dots, n; j = 1, 2, \dots, m \quad (4.8)$$

The conditional entropy H represents the uncertainty of the random variable Y under the condition of the known random variable X . The conditional entropy $H(Y|X)$ of the random variable Y under the condition given by the random variable X is defined as the entropy of the conditional probability distribution of Y under the given condition of X given the mathematical expectation for X :

$$H(Y|X) = \sum_{i=1}^n p_i H(Y|X = x_i) \quad (4.9)$$

where $p_i = P(X = x_i), i = 1, 2, \dots, n$.

Information gain: The information gain $g(D, A)$ of the feature A for the training data set D is defined as the difference between the empirical entropy $H(D)$ of the set D and the empirical condition entropy $H(D|A)$ of D under the given condition of the characteristic A , [105] which is

$$g(D, A) = H(D) - H(D|A) \quad (4.10)$$

According to the information gain criterion, the feature selection method is to calculate the information gain of each feature and compare the size of the training data set (or subset) D to select the characteristic of the information gain.

4.1.2.3 Decision tree generation

ID3 algorithm: The core is to apply the selected feature of information gain criterion to each node of the decision tree and construct the decision tree recursively [106]. The concrete method is to start from the root node, calculate the information gain of all the possible features to the node, select the feature with the largest information gain as the characteristic of the node, and establish the subnode from the different values of the feature. And then the subnodes recursively call the above method to build the decision tree, until all the information gain is very small or no features can be selected so far, and finally get a decision tree [107].

4.1.2.4 Decision tree pruning

In the decision tree learning, the process of simplifying the generated tree is called pruning.

Pruning algorithm:

Input: Generates the entire tree generated by the algorithm T , the parameter a ;

Output: pruned subtree T_a .

1. Calculating the empirical entropy of each node ϵ
2. Recursively retracting from the leaf node of the tree. Supposing a set of leaf nodes is retracted to their parent nodes before and after the whole tree is T_B and T_A respectively, the corresponding loss function values are $C_{\alpha}(T_B)$ and $C_{\alpha}(T_A)$, if $C_{\alpha}(T_B) > C_{\alpha}(T_A)$, then cuts the section, that is, the parent node becomes a new leaf node.
3. Return (2), before it can end. The subtree with the smallest loss function is obtained [108].

4.1.3 Bayes-based classification

Bayesian classification algorithm using probability and statistics to classify is a statistical classification method. Naive Bayes (NB) classification algorithm can be compared with decision tree and Neural Network Classification algorithm. The algorithm can be applied to large databases. The method is simple but has high accuracy and quick classification.

Bayesian theorem is

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)} \quad (4.11)$$

4.1.3.1 Naive Bayesian algorithm model

In machine learning, naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features. Naive Bayesian classification is a very simple classification algorithm. And the thought of naive Bayesian is that we solve the probability of each category under the conditions. Then we divide the items into the categories with the highest probability of occurrence. Naive Bayesian classification model is depicted in **Figure 4.2**.

Naive Bayes has been studied extensively since the 1950s. It was introduced under a different name into the text retrieval community in the early 1960s [109] and remains a popular (baseline) method for text categorization, the problem of judging documents as belonging to one category or the other (such as spam or legitimate, sports or politics, etc.) with word frequencies as the features. With appropriate preprocessing, it is competitive in this domain with more advanced methods including support vector machines [110]. It also finds application in automatic medical diagnosis [111].

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. It is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable [112]. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the

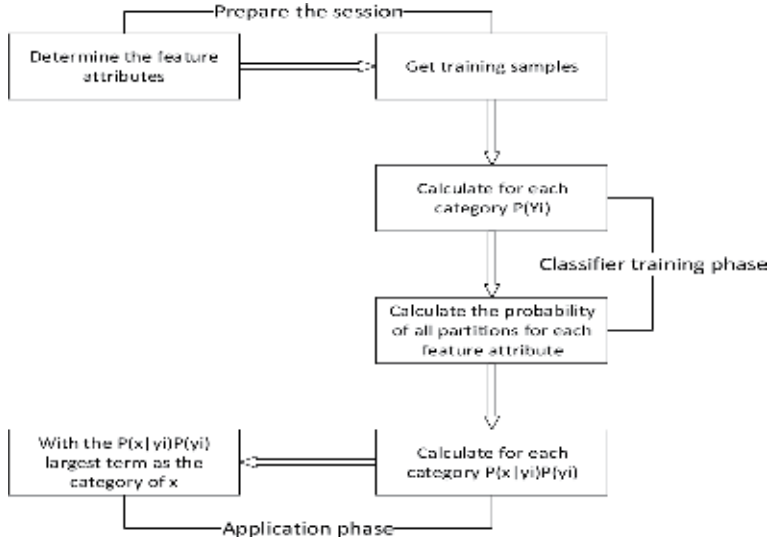


Figure 4.2.
Naive Bayesian algorithm model.

probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features.

Probabilistic model: Abstractly, naive Bayes is a conditional probability model—given a problem instance to be classified, represented by a vector $\mathbf{x} = (x_1, \dots, x_n)$ representing some n features (independent variables), it assigns to this instance probability

$$P(C_k | x_1, \dots, x_n) \quad (4.12)$$

for each of k possible outcomes or classes C_k .

The problem with the above formulation is that if the number of features n is large or if a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable. Using Bayes' theorem, the conditional probability can be decomposed as

$$p(C_k | \mathbf{x}) = \frac{p(C_k)p(\mathbf{x}|C_k)}{p(\mathbf{x})} \quad (4.13)$$

In plain English, using Bayesian probability terminology, the above equation can be written as

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}} \quad (4.14)$$

In practice, there is interest only in the numerator of that fraction, because the denominator does not depend on C and the values of the features F_i are given, so that the denominator is effectively constant. The numerator is equivalent to the joint probability model:

$$p(C_k, x_1, \dots, x_n) \quad (4.15)$$

which can be rewritten as follows, using the chain rule for repeated applications of the definition of conditional probability:

$$\begin{aligned}
 p(C_k, x_1, \dots, x_n) &= p(x_1, \dots, x_n, C_k) \\
 &= p(x_1|x_2, \dots, x_n, C_k)p(x_2, \dots, x_n, C_k) \\
 &= p(x_1|x_2, \dots, x_n, C_k)p(x_2|x_3, \dots, x_n, C_k)p(x_3, \dots, x_n, C_k) \\
 &= \dots \\
 &= p(x_1|x_2, \dots, x_n, C_k)p(x_2|x_3, \dots, x_n, C_k)\dots p(x_{n-1}|x_n, C_k)p(x_n|C_k)p(C_k)
 \end{aligned}
 \tag{4.16}$$

Now the “naive” conditional independence assumptions come into play: assume that each feature F_i is conditionally independent of every other feature F_j for $j \neq i$, given the category C . This means that

$$p(x_i|x_{i+1}, \dots, x_n, C_k) = p(x_i|C_k) \tag{4.17}$$

Thus, the joint model can be expressed as

$$\begin{aligned}
 p(C_k|x_1, \dots, x_n) &\propto p(C_k, x_1, \dots, x_n) \\
 &\propto p(C_k)p(x_1|C_k)p(x_2|C_k)p(x_3|C_k)\dots \\
 &\propto p(C_k)\prod_{i=1}^n p(x_i|C_k)
 \end{aligned}
 \tag{4.18}$$

This means that under the above independence assumptions, the conditional distribution over the class variable C is

$$p(C_k|x_1, \dots, x_n) = \frac{1}{Z} p(C_k) \prod_{i=1}^n p(x_i|C_k) \tag{4.19}$$

where the evidence $Z = p(\mathbf{x})$ is a scaling factor dependent only on x_1, \dots, x_n , that is, a constant if the values of the feature variables are known.

4.1.3.2 Parameter estimation of naive Bayesian algorithm

In the naive Bayesian algorithm, learning means that $P(Y = c)$ and $P(x = x = 1)$; the maximum likelihood estimation can be used to estimate the prior probability. The formal of maximum likelihood estimation is

$$P(Y = c_k) = \frac{\sum_{i=1}^N I(y_i = c_k)}{N}, k = 1, 2, \dots, K \tag{4.20}$$

Supposing the set of the j th feature x^j possible values is $a_{j1}, a_{j2}, \dots, a_{jS_j}$, the maximum likelihood estimation of the conditional probability $P(X^{(j)} = a_{ji}|Y = c_k)$ is

$$\begin{aligned}
 P(X^{(j)} = a_{ji}|Y = c_k) &= \frac{\sum_{i=1}^N I(x_i^{(j)} = a_{ji}, y_i = c_k)}{\sum_{i=1}^N I(y_i = c_k)} \\
 j &= 1, 2, \dots, n; I = 1, 2, \dots, S_j; k = 1, 2, \dots, K
 \end{aligned}
 \tag{4.21}$$

where x is the j th feature of the i th sample, a is the i th value that the j th feature may take, and i is the instruction function.

4.1.4 Support vector machine

Support vector machine is a supervised learning model and relates learning algorithm for analyzing data in classification and regression analysis [113]. Given a set of training instances, each training instance is marked as belonging to one or the other of the two categories. The SVM training algorithm creates a model that assigns the new instance to one of the two categories, making it a nonprobabilistic two-element linear classifier. The SVM model is to represent an instance as a point in space, so that the mapping allows instances of different categories to be separated by as wide a clear interval as possible. Then, the new instances are mapped to the same space, and the categories are predicted based on which side of the interval they fall [114].

4.1.4.1 Linear SVM

We consider the following form of point test set:

$$(\vec{x}_1, y_1), \dots, (\vec{x}_n, y_n) \quad (4.22)$$

where y_i is 1 or -1 , indicating that each one is a p -dimensional real vector. Find a “maximum interval hyperplane” that separates the point set of $y_i = 1$ and the set of points $y_i = -1$, and maximize the distance between the hyperplane and the nearest point. Any hyperplane can be written to the following set of points \tilde{x} :

$$\vec{w} \cdot \vec{x} - b = 0 \quad (4.23)$$

where w (not necessarily normalized) is the normal vector.

Hard interval: If these training data are linearly separable, two parallel hyperplanes of two types of data can be chosen so that the distance between them is as large as possible. The regions in these two hyperplanes are called “spaces,” and the maximum interval hyperplane is the hyperplane located in the middle of them [115]. These hyperplanes can be represented by equation

$$\vec{w} \cdot \vec{x} - b = 1 \quad \vec{w} \cdot \vec{x} - b = -1 \quad (4.24)$$

or

$$\vec{w} \cdot \vec{x} - b = -1 \quad (4.25)$$

It is not difficult to get the distance between the two hyperplanes by geometry, so we want to maximize the distance between the two planes; we need to minimize. At the same time, in order to make the sample data points in the hyperplane interval, we need to ensure that we meet one that represents all of the conditions:

$$\vec{w} \cdot \vec{x} - b \geq 1, y_i = 1 \quad (4.26)$$

or

$$\vec{w} \cdot \vec{x} - b \leq -1, y_i = -1 \quad (4.27)$$

These formalisms indicate that each data point must be on the correct side of the interval. These two formulas can be written as

$$y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1, 1 \leq i \leq n \quad (4.28)$$

You can use the formulas to get optimization problems: “Under the condition of minimizing, for $i = 1, \dots, n$.” The solution w and b of this problem determines our classifier.

Soft interval: In order to extend the SVM to the inseparable linearity of the data, we introduce the hinge loss function:

$$\max(0, 1 - y_i(\vec{w} \cdot \vec{x}_i - b)) \quad (4.29)$$

When the formal 4.28 that satisfies this function is zero, the value of the function is proportional to the distance from the interval for the data on the wrong side of the interval. Then we want to minimize it:

$$\left[\frac{1}{n} \sum_{i=1}^n \max(0, 1 - y_i(\vec{w} \cdot \vec{x}_i - b)) \right] + \lambda \|\vec{w}\|^2 \quad (4.30)$$

where the parameter is used to balance the relationship between increasing the size of the interval and ensuring that the correct side is on the interval. Thus, for sufficiently small values, the soft-interval SVM and the hard-spaced SVM will behave the same if the input data is linearly classified. But a feasible classification rule can be learned even if it is not linearly classified [115].

4.1.4.2 Kernels

Back in our discussion of linear regression, we had a problem in which the input x was the living area of a house, and we considered performing regression using the features x , x^2 , and x^3 (say) to obtain a cubic function. To distinguish between these two sets of variables, we’ll call the “original input value” the input attributes of a problem (in this case, x , the living area). When that is mapped to some new set of quantities that are then passed to the learning algorithm, we’ll call those new quantities the input features. (Unfortunately, different authors use different terms to describe these two things, but we’ll try to use this terminology consistently in these notes.) We will also let ϕ denote the feature mapping, which maps from the attributes to the features. For instance, in our example, we had

$$\Phi(X) = [x, x^2, x^3]^T \quad (4.31)$$

Rather than applying SVMs using the original input attributes x , we may instead want to learn using some features $\phi(x)$. To do so, we simply need to go over our previous algorithm and replace x everywhere in it with $\phi(x)$.

Since the algorithm can be written entirely in terms of the inner products $\langle x, z \rangle$, this means that we would replace all those inner products with $\langle \phi(x), \phi(z) \rangle$. Specially, given a feature mapping, we define the corresponding Kernel to be

$$K(x, z) = \phi(x)^T \phi(z) \quad (4.32)$$

Then, everything we previously had $\langle x, z \rangle$ in our algorithm, which we could simply replace with $K(x, z)$, and our algorithm would now be learning using the features.

Now, given ϕ , we could easily compute $K(x, z)$ by finding $\phi(x)$ and $\phi(z)$ and taking their inner product. But what's more interesting is that often, $K(x, z)$ may be very inexpensive to calculate, even though $\phi(x)$ itself may be very expensive to calculate (perhaps because it is an extremely high-dimensional vector) [116]. In such settings, by using in our algorithm an efficient way to calculate $K(x, z)$, we can get SVMs to learn in the high-dimensional feature space given by ϕ but without ever having to explicitly find or represent vectors $\phi(x)$ [117].

Let's see an example. Suppose $x, z \in \mathbb{R}^n$, and consider

$$K(x, z) = (x^T z)^2 \quad (4.33)$$

We can also write this as

$$\begin{aligned} K(x, z) &= \left(\sum_{i=1}^n x_i z_i \right) \left(\sum_{j=1}^n x_j z_j \right) \\ &= \sum_{i=1}^n \sum_{j=1}^n x_i x_j z_i z_j \\ &= \sum_{i,j=1}^n (x_i x_j) (z_i z_j) \end{aligned} \quad (4.34)$$

Thus, we see that $K(x, z) = \phi(x)^T \phi(z)$, where the feature mapping ϕ is given (shown here for the case of $n = 3$) by

$$\phi(x) = [x_1 x_1, x_1 x_2, x_1 x_3, x_2 x_1, x_2 x_2, x_2 x_3, x_3 x_1, x_3 x_2, x_3 x_3]^T \quad (4.35)$$

Note that whereas calculating the high-dimensional $\tilde{\phi}(x)$ requires $O(n^2)$ time, finding $K(x, z)$ takes only $O(n)$ time linear in the dimension of the input attributes. For a related kernel, also consider

$$\begin{aligned} K(x, z) &= (x^T z + c)^2 \\ &= \sum_{i,j=1}^n (x_i x_j) (z_i z_j) + \sum_{i=1}^n (\sqrt{2c} x_i) (\sqrt{2c} z_i) + c^2 \end{aligned} \quad (4.36)$$

(Check this yourself.) This corresponds to the feature mapping (again shown for $n = 3$):

$$\phi(x) = [x_2 x_1, x_2 x_2, x_2 x_3, x_3 x_1, x_3 x_2, x_3 x_3, \sqrt{2c} x_1, \sqrt{2c} x_2, \sqrt{2c} x_3, c]^T \quad (4.37)$$

and the parameter c controls the relative weighting between the x_i (first order) and the $x_i x_j$ (second order) terms.

More broadly the kernel $K(x, z) = (x^T z + c)^d$ corresponds to a feature mapping to a feature space, corresponding of all monomials of the form $x_{i_1} x_{i_2} \dots x_{i_k}$ that are up to order d . However, despite working in this $O(nd)$ -dimensional space, computing $K(x, z)$ still takes only $O(n)$ time, and hence we never need to explicitly represent feature vectors in this very high-dimensional feature space [118].

Now, let's talk about a slightly different view of kernels. Intuitively (and there are things wrong with this intuition, but never mind), if $\phi(x)$ and $\phi(z)$ are close together, then we might expect $K(x, z) = \phi(x)^T \phi(z)$ to be large. Conversely, if $\phi(x)$ and $\phi(z)$ are far apart say nearly orthogonal to each other, then $K(x, z) = \phi(x)^T \phi(z)$

will be small [119]. So, we can think of $K(x, z)$ as some measurement of how similar are $\phi(x)$ and $\phi(z)$ or of how similar are x and z .

Given this intuition, suppose that for some learning problem that you're working on, you've come up with some function $K(x, z)$ that you think might be a reasonable measure of how similar x and z are. For instance, perhaps you chose

$$K(x, z) = \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right) \quad (4.38)$$

This is a reasonable measure of x and z 's similarity and is close to 1 when x and z are close and near 0 when x and z are far apart. Can we use this definition of K as the kernel in an SVM? In this particular example, the answer is yes. (This kernel is called the Gaussian kernel and corresponds to an infinite dimensional feature mapping ϕ .) But more broadly, given some function K , how can we tell if it's a valid kernel, i.e., can we tell if there is some feature mapping so that $K(x, z) = \phi(x)^T \phi(z)$ for all x, z ?

Suppose for now that K is indeed a valid kernel corresponding to some feature mapping ϕ . Now, consider some finite set of m points (not necessarily the training set) $\{x^{(1)}, \dots, x^{(m)}\}$, and let a square m -by- m matrix K be defined so that its (i, j) -entry is given by $K_{ij} = K(x^{(i)}, x^{(j)})$. This matrix is called the Kernel matrix [120]. Note that we've overloaded the notation and used K to denote both the kernel function $K(x, z)$ and the kernel matrix K , due to their obvious close relationship.

4.1.4.3 Regularization and the nonseparable case

The derivation of the SVM as presented so far assumed that the data are linearly separable. While mapping data to a high-dimensional feature space via ϕ does generally increase the likelihood that the data is separable, we can't guarantee that it always will be so. Also, in some cases it is not clear that finding a separating hyperplane is exactly what we'd want to do, since that might be susceptible to outliers. For instance, the left figure below shows an optimal margin classifier, and when a single outlier is added in the upper-left region (right figure) (Figure 4.3), it causes the decision boundary to make a dramatic swing, and the resulting classifier has a much smaller margin [121].

To make the algorithm work for nonlinearly separable data sets as well as be less sensitive to outliers, we reformulate our optimization (using l_1 regularization) as follows:

$$\begin{aligned} \min_{\gamma, \omega, b} \quad & \frac{1}{2} \|\omega\|^2 + C \sum_{i=1}^m \varepsilon_i \\ \text{s.t.} \quad & y^{(i)}(\omega^T x^{(i)} + b) \geq 1 - \varepsilon_i, \quad i = 1, \dots, m \\ & \varepsilon_i \geq 0, \quad i = 1, \dots, m. \end{aligned} \quad (4.39)$$

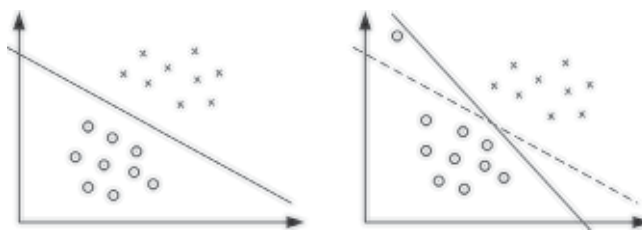


Figure 4.3.
 Linearly separable.

Thus, examples are now permitted to have (functional) margin less than 1, and if an example has functional margin $1 - \varepsilon_i$ (with $\varepsilon > 0$), we would pay a cost of the objective function being increased by $C\varepsilon_i$. The parameter C controls the relative weighting between the twin goals of making the $\|\omega\|^2$ small (which we saw earlier makes the margin large) and of ensuring that most examples have functional margin at least 1.

As before, we can form the Lagrangian:

$$Y(\omega, b, \varepsilon, \alpha, \gamma) = \frac{1}{2}\omega^T\omega + C\sum_{i=1}^m\varepsilon_i - \sum_{i=1}^m\alpha_i[y^{(i)}(x^T\omega + b) - 1 + \varepsilon_i] - \sum_{i=1}^m\gamma_i\varepsilon_i. \quad (4.40)$$

Here, the α_i 's and γ_i 's are our Lagrange multipliers (constrained to be ≥ 0). We won't go through the derivation of the dual again in detail, but after setting the derivatives with respect to ω and b to zero as before, substituting them back in, and simplifying, we obtain the following dual form of the problem:

$$\begin{aligned} \max_{\alpha} \quad & W(\alpha) = \sum_{i=1}^m\alpha_i - \frac{1}{2}\sum_{i,j=1}^m y^{(i)}y^{(j)}\alpha_i\alpha_j\langle x^{(i)}, x^{(j)} \rangle \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C, \quad i = 1, \dots, m \\ & \sum_{i=1}^m\alpha_i y^{(i)} = 0, \quad i = 1, \dots, m. \end{aligned} \quad (4.41)$$

As before, we also have that ω can be expressed in terms of the α_i 's, so that after solving the dual problem, we can continue making our predictions. Note that, somewhat surprisingly, in adding l_1 regularization, the only change to the dual problem is that what was originally a constraint that $0 \leq \alpha_i$ has now become $0 \leq \alpha_i \leq C$. Also, the KKT dual-complementarity conditions (which in the next section will be useful for testing for the convergence of the SMO algorithm) are

$$\begin{aligned} \alpha_i = 0 & \Rightarrow y^{(i)}(\omega^T x^{(i)} + b) \geq 1 \\ \alpha_i = C & \Rightarrow y^{(i)}(\omega^T x^{(i)} + b) \leq 1 \\ 0 < \alpha_i < C & \Rightarrow y^{(i)}(\omega^T x^{(i)} + b) = 1 \end{aligned} \quad (4.42)$$

Now, all that remains is to give an algorithm for actually solving the dual problem, which we will do in the next section.

4.1.5 Other classification approaches

4.1.5.1 Random forest

As a newly developed, highly flexible machine learning algorithm, random forest (RF) has a wide range of application prospects, from marketing to health-care insurance; both can be used to do marketing modeling, statistics of customer sources, retention, and loss and can also be used to predict the risk of disease and the susceptibility of the patient [122]. As the name implies, random forest is a random way to build a forest, the forest which has lots of decision tree compositions. There is no connection between each decision tree of a random forest. After setting the forest, when there is a new input sample into the forest, each tree in the forest makes a separate judgment, to see which class (for the classification algorithm) the sample should belong to and then see which class is selected most to predict this sample for that class [123].

Advantage:

1. In all current algorithms, there is excellent accuracy.
2. Can run effectively on large data sets, can handle input samples with high-dimensional characteristics, and do not need dimensionality reduction.
3. Can assess the importance of each feature in classification issues.
4. In the generation process, an unbiased estimate of the internal generation error can be obtained.
5. For the default value of the problem can also get very good results [124].

4.1.5.2 *K-nearest neighbors*

K-nearest neighbor (*KNN*) classification algorithm is a more mature method and one of the simplest machine learning algorithms. The idea of this method is as follows: in the *KNN* algorithm, the selected neighbors are objects that have been correctly classified [95]. The method determines the class of the sample to be sorted only on the classification decision based on the nearest one or several samples. Although the *KNN* method relies on the limit theorem in principle, it is only related to a very small number of adjacent samples in the category decision [125]. The *KNN* method mainly depends on the surrounding neighboring samples, rather than by discriminating the domain method to determine the category; therefore, *KNN* method is more suitable for other methods than for crossover or overlapping more sample sets [126].

KNN algorithm can be used not only for classification but also for regression. By finding the *k*-nearest neighbors of a sample and assigning the average of the attributes of those neighbors to the sample, the attributes of the sample can be obtained. A more useful way is to give different weights to the effects of the neighbors on the samples, such as the weight and the distance being proportional [127].

The main drawback of the algorithm in the classification is that when the sample is not balanced, if a class of sample size is large, and other class sample capacities are very small, it may lead to input when a new sample, the sample *K* neighbors in the large-capacity class of samples, is accounted for the majority. So it can be improved by using the method of weighting (and the neighbor weights with small distance from the sample) [128]. Another disadvantage of this approach is that the computational complexity is large because the distance to its known sample is calculated for each text to be classified in order to obtain its nearest neighbor. The current common solution is to preknow the sample point of the clip, in advance to remove the classification of the role of small samples [129]. The algorithm is more suitable for the automatic classification of the class with relatively large sample size, and the classification of the class with smaller sample size is more likely to produce errors [130]. Its advantages are high precision, not sensitive to the outliers, and no data input assumptions. And it applies to numeric and nominal data.

4.1.5.3 *Logistic regression and boosting*

In addition to the algorithms described above, there are two more sophisticated classification algorithms. Logistic regression can also be used for classification. First, we need to find a suitable predictive function to predict the result of the input data and then construct a loss function to represent the deviation between the predicted

output and the training data and use the gradient descent method to complete the logistic regression to find the minimum value of the loss function. Features are simple to achieve; the calculation is very small and fast and has low storage resources. Another boosting algorithm and the above algorithm ideas are not the same; the classifier is weak. It's better only in some of the situations or characteristics under the performance. We need to find some in their respective circumstances performance of good classifiers; they can be combined to become a strong classifier [131].

4.2. Clustering

Evaluating the performance of a clustering algorithm is not as trivial as counting the number of errors or the precision and recall of a supervised classification algorithm. In particular any evaluation metric should not take the absolute values of the cluster labels into account but rather if this clustering defines separations of the data similar to some ground-truth set of classes or satisfies some assumption such that members belong to the same class are more similar than members of different classes according to some similarity metric [132].

4.2.1 Performance evaluation for clustering

4.2.1.1 Adjusted Rand index

$$RI = \frac{a + b}{C_2^{n_{samples}}} \quad (4.43)$$

where C is the actual category information, K is the result of the clustering, a is the logarithm of the same class in both C and K , and b is the number of elements in C and K , where $C_2^{n_{samples}}$ represents the logarithm that can be formed in the data set and the RI is in the range of $[0,1]$. The larger the value means that the clustering result is more consistent with the real situation. The higher the RI is, the higher the accuracy of the clustering effect, and the higher the purity in each class it will get [133].

In order to achieve the “random results in the case of clustering results, the indicators should be close to zero.” If the Rand coefficient (adjusted rand index) is raised, it has a higher degree of distinction:

$$ARI = \frac{RI - E[RI]}{\max RI - E[RI]} \quad (4.44)$$

The ARI is in the range of $[-1, 1]$. The larger value means that the clustering results are consistent with the real situation. From a broad perspective, the coincident degree of the two data distributions is measured by ARI [134].

Advantages:

1. Random (uniform) label assignments have an ARI score close to 0.0.
2. Bounded range in $[-1, 1]$.
3. No assumption is made on the cluster structure.

Drawbacks: ARI requires knowledge of the ground-truth classes.

4.2.1.2 Mutual information-based scores

The mutual information is a function that measures the agreement of the two assignments, ignoring permutations. Two different normalized versions of this measure are available, normalized mutual information (NMI) and adjusted mutual information (AMI) [135]:

$$I(X, Y) = \sum_{x,y} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} \quad (4.45)$$

Entropy as the denominator to adjust the MI value between 0 and 1 is used by standardized mutual information, a more common realization is as follows:

$$U(X, Y) = 2R = 2 \frac{I(X; Y)}{H(X) + H(Y)} \quad (4.46)$$

$$H(X) = \sum_{i=1}^n p(x_i) I(x_i) = \sum_{i=1}^n p(x_i) \log_b \frac{1}{p(x_i)} = - \sum_{i=1}^n p(x_i) \log_b p(x_i) \quad (4.47)$$

Advantages:

1. Random (uniform) label assignments have a AMI score close to 0.0
2. Bounded range in [0, 1].
3. No assumption is made on the cluster structure.

4. Drawbacks:

MI-based measures require the knowledge of the ground-truth classes.

4.2.1.3 Homogeneity, completeness, and V-measure

Homogeneity: Each cluster contains only members of a single class:

$$h = 1 - \frac{H(C|K)}{H(C)} \quad (4.48)$$

where $H(C|K)$ is the conditional entropy of a class given a cluster assignment, which is obtained by the following formula:

$$H(C|K) = - \sum_{c=1}^{|C|} \sum_{k=1}^{|K|} \frac{n_{c,k}}{n} \log \left(\frac{n_{c,k}}{n_k} \right) \quad (4.49)$$

$H(C)$ is the class entropy, the formula is

$$H(C) = - \sum_{c=1}^{|C|} \frac{n_c}{n} \log \left(\frac{n_c}{n} \right) \quad (4.50)$$

where n is the total number of samples; n_c and n_k belong to the number of samples of class c and class k , respectively; and $n_{c,k}$ is the number of samples divided from class c to class k .

Completeness: All members of a given class are assigned to the same cluster:

$$h = 1 - \frac{H(K|C)}{H(K)} \quad (4.51)$$

where the conditional entropy $H(K|C)$ and the class entropy $H(K)$ are obtained symmetrically according to the above formula.

V-measure is homogeneity and completeness of the harmonic mean; the formula is

$$v = 2 \cdot \frac{h \cdot c}{h + c} \quad (4.52)$$

Advantages:

1. Bounded scores.
2. Intuitive interpretation.
3. No assumption is made on the cluster structure.

Drawbacks:

1. Not normalized with regard to random labeling.
2. Require the knowledge of the ground-truth classes.

There are many other performance evaluation approaches for clustering: Fowlkes-Mallows scores, Silhouette Coefficient, and Calinski-Harabasz index [136].

4.2.2 Partition clustering

4.2.2.1 K-means

K-means and *KNN* are all starting with k , but they are two kinds of algorithms—*KNN* is the classification algorithm in supervised learning, while K-means is the clustering algorithm in unsupervised learning. They both use the neighbor information to mark the category.

K-means is the simplest and most efficient in the clustering algorithm. Its core idea is that the initial centroid is specified by the user to repeat the iteration until the algorithm converges as a cluster [137].

The *K*-means algorithm divides a set of N samples X into K disjoint cluster C , each described by the mean μ_j of the samples in the cluster. The means are commonly called the cluster “centroid”; note that they are not, in general, points from X , although they live in the same space. The *K*-means algorithm aims to choose centroid that minimizes the inertia or within-cluster sum of squared criterion:

$$\sum_{i=0}^n \min_{\mu_j \in C} (\|x_j\| - \|\mu_i\|)^2 \quad (4.53)$$

This algorithm is actually very simple, as shown in **Figure 4.4**.

From the above figure, we can see that A , B , C , D , and E are five points in the figure. And the gray point is our seed point, that is, we used to find a point group. There are two seed points, so $K = 2$.

K-means algorithm:

1. Randomly take K (where $K = 2$) seed points in the graph;
2. Then calculate the distance of the K seed points at all points in the graph. If P_i is closest to the seed point S_i , then P_i belongs to the S_i point group. In the picture above, we can see that A and B belong to the above seed points, C, D, E belong to the middle of the seed points (below);
3. Next, we want to move the seed point to the center of his “point group”. (See (3) on the **Figure 4.4**);
4. Then repeat step. 2 and 3 until the seed point does not move (we can see the fourth step in the figure above the seed points to aggregate A, B, C , the following seed points are aggregated with D, E) [138].

In general, the algorithm for finding the point group center point can be very simple to use the average of the X/Y coordinates of each point. However, there are three other ways to find the center point:

1. Minkowski distance formula— λ can be arbitrary, can be negative, and can be positive or infinity:

$$d_{ij} = \sqrt[\lambda]{\sum_{k=1}^n |x_{ik} - x_{jk}|^\lambda} \quad (4.54)$$

2. Euclidean distance formula—that is, the case of the first formula when $\lambda = 2$

$$d_{ij} = \sqrt{\sum_{k=1}^n |x_{ik} - x_{jk}|^2} \quad (4.55)$$

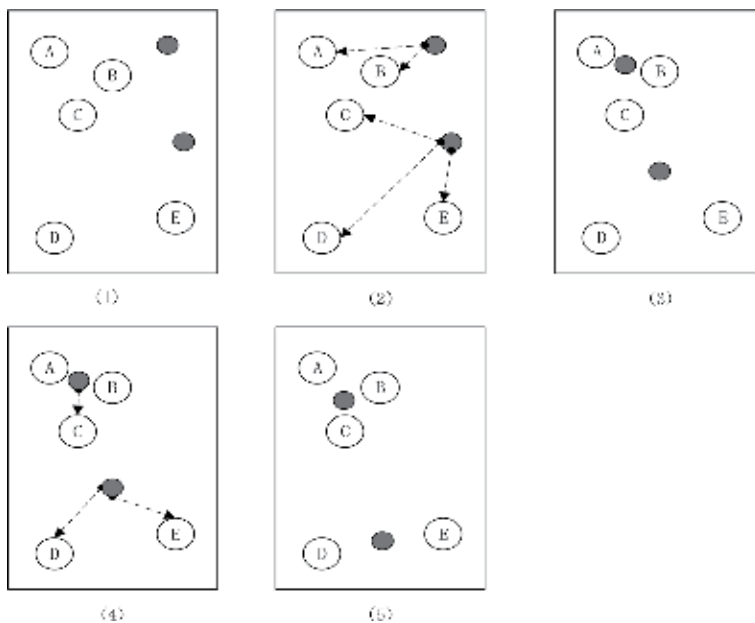


Figure 4.4.
 The procedure of K-means.

3. CityBlock distance formula—that is, the case of the first formula when $\lambda = 1$

$$d_{ij} = \sum_{k=1}^n |x_{ik} - x_{jk}| \quad (4.56)$$

Advantage:

1. The principle is simple.
2. It is easy to implement.
3. Clustering effect mainly depends on the choice of K .

Drawbacks:

1. The number of K cannot be determined.
2. It is sensitive to outliers and easily leads to the center point offset.
3. Algorithm complexity is not easy to control $O(NKm)$; the number of iterations may be more.
4. The algorithm is calculated as a local optimal solution rather than a global optimal.
5. The results are not stable and cannot be calculated incrementally.

4.2.2.2 *K-medoids*

It looks like K-means, but *K-medoids* and K-means are different. The difference is the center of the selection. In K-means, we will be the center of the current cluster as the average of all data points. In *K-medoids* algorithm, we will choose from the current cluster such a point—to all other (the current cluster) points of the sum of the minimum—as the center point [139].

K-medoids algorithm description:

1. First, randomly select a set of clustering samples as the center point set.
2. Each center point corresponds to a cluster.
3. Calculate the distance from each sample point to each center point (e.g., Euclidean distance); place the sample point in the cluster with the shortest center point.
4. Calculate the point at which the absolute error of the distance from each sample point in the cluster is the smallest, as the new center point.
5. If the new center point set is the same as the original center point set, the algorithm terminates; if the new center point set is not exactly the same as the original center point set, return b [140].

In order to determine whether a non-object representative $O(\text{random})$ is a better alternative to the current representative object O_j , consider the following four cases for each nonrepresentative object P (**Figure 4.5**):

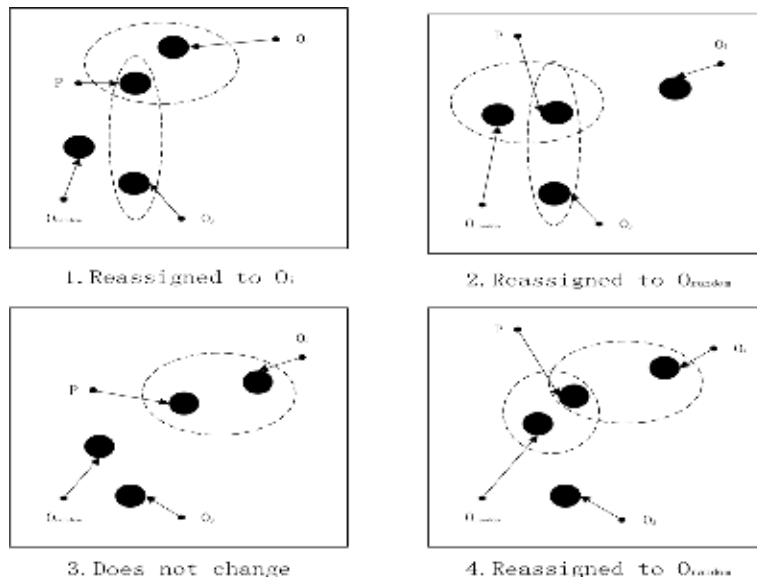


Figure 4.5.
 Examples of the four cases.

1. p is currently attached to the representative object O_j . If O_j is replaced by $O(\text{random})$, and p is closest to O_i , $i \neq j$, then p is reassigned to O_i .
2. p is currently attached to the representative object O_j . If O_j is replaced by $O(\text{random})$, and p is closest to $O(\text{random})$, then p is reassigned to $O(\text{random})$.
3. p is currently attached to the representative object O_i , $i \neq j$. If O_j is replaced by $O(\text{random})$, and p is still closest to O_i , then the membership of the object does not change.
4. p is currently attached to the representative object O_i , $i \neq j$. If O_j is replaced by $O(\text{random})$ and p is nearest, then p is reassigned to $O(\text{random})$.

K-medoids algorithm advantages and drawbacks:

1. The K-medoids algorithm has the advantage of being able to handle large datasets, the result clusters are quite compact, and the clusters are clearly distinct from the cluster. This is the same as the K-means algorithm.
2. The algorithm's shortcomings is the same as K-means, for example, must determine the number of clusters and the central point, the number of clusters and the choice of the central point of the results of a great impact; generally get a local optimal solution after the stop; is not suitable for data other than the value type; only for clustering results for the convex data set and so on.
3. Compared with K-means, the K-medoids algorithm is less sensitive to noise. As a result, the outliers do not cause the division of the results, even if the deviation is too large and a small number of data will not cause significant impact.
4. K-medoids are considered to be an improvement over K-means for the above reasons, but the time complexity of the algorithm is increased by $O(n)$

compared to K-means due to the calculation in the way of the center point selection.

4.2.3 Hierarchical clustering

Hierarchical clustering uses a method to decompose or aggregate the data set layer by layer, until all the underlying data of the last layer is separated to meet the requirements. Therefore, according to the different principles of decomposition between convergence, hierarchical clustering can be divided into agglomerative and divisive. Hierarchical clustering is used to split the data from top to bottom or bottom-up into a tree. Hierarchical clustering involves nested clustering [141]. Nested clustering means that R_1 contains another R_2 in a cluster. That is, R_2 is nested into R_1 or R_1 nested R_2 . What specifically does nest mean? Clustering $R_1 = \{\{x_1, x_2\}, \{x_3\}, \{x_4, x_5\}\}$ is nested in clustering $R_2 = \{\{x_1, x_2, x_3\}, \{x_4, x_5\}\}$ but is not nested in clusters $R_3 = \{\{x_1, x_4\}, \{x_3\}, \{x_2, x_5\}\}$, that is, each subclass of R_2 contains the corresponding subclasses of R_1 . The hierarchical clustering algorithm generates a nested clustering hierarchy. The algorithm contains N steps at most. In step t , the operation generates new clusters by merging or splitting on the basis of the clustering of the preceding $t - 1$.

4.2.3.1 AGNES

Agglomeration hierarchical clustering: This bottom-up strategy first takes each object as a separate cluster and then merges these clusters as growing clusters until all the objects are in a cluster or reach a certain termination condition. Agglomerative nesting (*AGNES*) is a hierarchical clustering algorithm. If the distance between an object in cluster C_1 and one of the objects in cluster C_2 is the smallest among the inter-object European distances of all clusters, C_1 and C_2 may be merge. This is a single join method, where each cluster can be represented by all the objects in the cluster. The similarity between the two clusters is determined by the similarity of the nearest pair of data points in the two clusters.

Algorithm description:

Input: the database containing n objects, the number of terminating cluster k ;

Output: k clusters.

1. Treat each object as an initial cluster;
2. Repeat the first step;
3. Find the nearest two clusters based on the most recent data points in the two clusters;
4. Merge two clusters to generate a new set of clusters;
5. Until reaches the number of defined clusters

Algorithm performance:

1. Simple, but faced with the merger point to choose the difficult situation;
2. Once a group of objects are merged, they will not be revoked;
3. The complexity of the algorithm is $O(n^2)$, not suitable for large data set calculation.

4.2.3.2 BIRCH

Balanced iterative reducing and clustering using hierarchies (*BIRCH*) is a balanced iterative protocol and clustering using hierarchical methods. First of all, *BIRCH* is a clustering algorithm; its biggest feature is the ability to use limited memory resources to complete the large data set of high-quality clustering, while single-scan data set can minimize the *I/O* cost [142].

The *BIRCH* algorithm introduces two concepts: clustering features and clustering feature trees. Clustering feature (CF) is the core of the *BIRCH* incremental clustering algorithm. The nodes in the *CF* tree are composed of *CF*, and one *CF* is a triple, which represents all the information of the cluster. Given N d -dimensional data points $\{x_1, x_2, \dots, x_n\}$, *CF* is defined as follows:

$$CF = \{N, LS, SS\} \quad (4.57)$$

where N is the number of nodes in the subclass, LS is the sum of the N nodes, and SS is the sum of squares of N nodes:

$$\begin{aligned} CF_1 = CF_2 &= (n_1 + n_2, LS_1 + LS_2, SS_1 + SS_2) \\ CF_1 = CF_2 &= (n_1 + n_2, LS_1 + LS_2, SS_2) \\ CF_1 &= SS_2. \end{aligned} \quad (4.58)$$

The *CF* tree is structured like a B-tree with two parameters: the internal node balance factor B , the leaf node balance factor L , and the cluster radius threshold T . Each node in the tree contains up to B child nodes, denoted as $(CF_i, CHILD_i)$, $1 \leq i \leq B$; CF_i is the i th clustering feature in this node; $CHILD_i$ points to the i th child node of the node, corresponding to the i th clustering feature of this node.

BIRCH algorithm features:

1. *BIRCH* attempts to use the available resources to generate the best clustering results. An important consideration is to minimize the *I/O* time with the limited memory.
2. *BIRCH* uses a multistage clustering technique: unilateral scanning of data sets produces a basic clustering, and one or more additional scans can further improve the quality of clustering.
3. *BIRCH* is an incremental clustering method because it makes decisions about the clustering of each data point based on the data points that have been processed, rather than the global data points.
4. If the cluster is not spherical, *BIRCH* does not work well because it uses the concept of radius or diameter to control the boundaries of the cluster [143].

4.2.3.3 CURE

CURE is a novel hierarchical clustering algorithm that selects intermediate strategies based on centroid and representative object-based methods. It is different from a single centroid or object to represent a class, rather a fixed number of representative points in the data space. A representative point of a class is generated by first selecting the scattered objects in the class and then “shrinking” or moving them according to a particular score or contraction factor. At each step of the algorithm, two classes with the closest distance pair (each point comes from

different class) are merged. Each class has more than one representative point so that the CURE can accommodate nonspherical geometries. The contraction or cohesion of a class can help to control the effects of isolated points. Therefore, CURE is more robust to the treatment of isolated points and can identify nonspherical and large changes in the size of the class. For large databases, CURE combines random sampling with division: a random sample is divided at first; then each partition is partially clustered [144].

At the beginning, each point is a cluster in CURE's algorithm, then we continue merging the nearest cluster until the number of clusters becomes K that we require. It is a split hierarchical clustering. The algorithm is divided into the following six steps:

1. Extract a random sample S from the source data object.
2. Divide the sample S into a set of partitions.
3. On the division of local clustering.
4. Propose isolated points by random sampling. If a cluster grows too slowly, remove it.
5. Cluster local clusters.
6. Mark the data with the corresponding cluster label.

4.2.4 Density-based clustering

4.2.4.1 DBSCAN

DBSCAN defines the cluster as the largest set of densely connected points, dividing the region with sufficient high density into clusters and finding clusters of arbitrary shapes in the spatial database of noise.

DBSCAN is based on a set of neighborhoods to describe the degree of compactness of the sample set, and the parameters are used to describe the degree of tightness of the sample distribution in the neighborhood. Where the neighborhood distance threshold of a sample is described, $MinPts$ describes the threshold of the number of samples in the neighborhood of the distance of a sample [145].

DBSCAN clustering algorithm:

Input: sample set $D = (x_1, x_2, \dots, x_m)$, neighborhood parameters $(\epsilon, MinPts)$, sample distance measurement;

Output: cluster division C .

1. Initialize the core object set $\Omega = \phi$, initialize the cluster number $k = 0$, initialize the unattached sample set $\Gamma = D$, cluster partition $C = \phi$.
2. For $j = 1, 2, \dots, m$, follow the steps below to find all the core objects: find the ϵ -neighborhood subsample set $N_\epsilon(x_j)$ of the sample x_j by the distance measurement method; add the sample x_j to the core object sample set if the number of sample samples satisfies $|N_\epsilon(x_j)| \geq MinPts$: $\Omega = \Omega \cup \{x_j\}$.
3. If the core object set $\Omega = \phi$, then the algorithm ends, otherwise go to step 4.
4. In the core object set Ω , randomly select a core object o , initialize the current cluster core object queue $\Omega_{cur} = o$, initialize the class number $k = k + 1$,

initialize the current cluster sample set $C_k = o$ Access to sample set $\Gamma = \Gamma - o$
 If the current cluster core object queue is $\Omega_{cur} = \phi$, then the current cluster C_k
 is completed, update the cluster partition $C = \{C_1, C_2, \dots, C_k\}$, update the core
 object set $\Omega = \Omega C_k$, Step 3.

5. In the current cluster core object queue Ω_{cur} out of a core object o' , through
 the neighborhood distance threshold Δ to find all the Δ -neighborhood
 sub-sample set $N_\varepsilon(o')$, so that $\Delta = N_\varepsilon(o') \cap \Gamma$, Update the current cluster sample
 set $C_k = C_k \cup \Delta$, update the nonaccess sample set $\Gamma = \Gamma - \Delta$, and proceed to
 step 5.

The output is: cluster partition $C = \{C_1, C_2, \dots, C_k\}$
 Advantages of *DBSCAN*:

1. It can be clustered for any form of dense data set; the relative, K-means
 clustering algorithm is generally applicable only to convex data set.
2. It can be found in the cluster at the same time abnormal points; the data set is
 not sensitive to the abnormal point.
3. The clustering results are not biased. In contrast, the initial values of clustering
 algorithms such as K-Means have a great influence on the clustering results.

Disadvantages of *DBSCAN*:

1. If the density of the sample set is not uniform, the clustering distance
 difference is very large, the clustering quality is poor, and then *DBSCAN*
 clustering is generally not suitable.
2. If the sample set is large, the clustering time is longer, and the *KD* tree or
 spheroid is established to search for the nearest neighbor.
3. Relative to the traditional *K-Means* clustering algorithm, tuning parameters is
 slightly more complex. The main parameters includes distance threshold, the
 number of neighborhood samples and the MinPts joint parameters. Different
 combinations of parameters have a great impact on the final clustering.

4.2.4.2 Ordering points to identify the clustering structure (*OPTICS*)

OPTICS algorithm can get different density clustering. It is directly said that
 through the *OPTICS* algorithm processing, you can get any density clustering.
 Because the *OPTICS* algorithm outputs an ordered queue of samples, a cluster of
 arbitrary densities can be obtained from this queue [146].

OPTICS is a density clustering algorithm developing from *DBScan*. The core idea
 of density clustering is to measure the density of the space in which the point is
 located through the number of neighbors in ε neighborhood. If the neighborhood of
 ε neighborhood exceeds a certain threshold MinPts, it is considered that the point is
 in a cluster called core object; otherwise it is considered to be the boundary of a
 cluster called the boundary point (boundary object).

In order to have more elaborate characterization capabilities, *OPTICS* defines
 the concept of core distance and reachable distance: *OPTICS* defines the core
 distance as that suppose the minimum radius of MinPts neighbors is MinPts dis-
 tance (p), then the core distance of p is defined as core distance ε .

In other words, the core distance is the smallest neighborhood radius at which a point becomes the core point. In this case, the kernel distance is the smallest neighborhood radius.

4.2.5 Grid-based clustering

4.2.5.1 STING

STING is a grid-based multi-resolution clustering technique that divides the spatial region into rectangular cells. For different levels of resolution, there are usually multiple levels of rectangular cells, which form a hierarchical structure: each unit of the upper layer is divided into multiple lower layers. Statistics (such as average, maximum, and minimum) for each grid cell attribute are precomputed and stored. These statistics are used to answer queries.

Statistical information grid (STING) divides the spatial region into rectangular units. The calculation of the STING algorithm grid is independent of the query; the grid structure facilitates parallel processing and incremental updating; the efficiency is high: the time complexity is $O(n)$, which is the number of objects. The granularity of the lowest level of the mesh structure determines the quality of the STING algorithm clustering. The algorithm handles faster, but the quality and accuracy of the cluster may be reduced [147].

Common parameters in the grid count are as follows: the number of objects in the grid mean, the average of all values in the grid stdev, the standard deviation of the attribute values in the grid min, the minimum value of the attribute value in the grid max, the maximum value of the attribute value in the grid distribution, and the distribution type of the attribute values in the grid, such as normal distribution, even distribution, exponential distribution, or none (distribution type unknown).

4.2.5.2 CLIQUE

The CLIQUE algorithm divides each dimension of the data space into equal-length intervals to form the same size and disjoint grid cells. Since the points in the same cell belong to the same class, points in the same grid can be aggregated and treated as an object. All clustering operations are performed on grid cells, so the clustering process is independent of the number of data points in the data set, only about the number of grids.

The central idea of the CLIQUE clustering algorithm is as follows:

1. Given a large set of multidimensional data points, the data points are usually not evenly distributed in the data space. The CLIQUE distinguishes between sparse and “crowded” areas (or cells) in the space to discover the global distribution pattern of the data set.
2. If the unit contains data points that exceed an input model parameter, the unit is dense. In the CLIQUE algorithm, the cluster is defined as the largest set of connected dense cells. According to the central idea of this algorithm, grid-based and density-based clustering algorithm can be defined as follows: the data space is divided into grid and then the number of points falling into a cell as the density of this unit. At this time you can specify a value, when a grid unit of the number of points greater than the value said that the cell is dense. Finally, clustering is defined as a collection of all “dense” cells that are connected.

4.2.6 Self-organizing maps (SOM)

Self-organizing mapping neural network, that is, self-organizing maps (SOM), can be unsupervised data clustering. Its idea is very simple; in essence, it is only input layer—hidden layer of the neural network. A node in the hidden layer represents a class that needs to be clustered. In the case of “competitive learning,” each input sample is found in the hidden layer and the node that matches it, called its active node, also called “winning neuron” [148]. The parameter of the active node is updated immediately by the random gradient descent method. At the same time, the points close to the active node also update the parameters appropriately according to their distance from the active node.

So, SOM is a feature of the hidden layer of the node, and also is a topological relationship. If you want a two-dimensional topological relationship, then the line is connected into a plane, as shown below (also known as Kohonen Network): if you want a one-dimensional model, then the hidden nodes are connected into a line (**Figure 4.6**).

Since the hidden layer has a topological relation, we can also say that the SOM can discretize the input of any dimension to discrete space of one or two dimensions (less common dimension). Computation layer inside the nodes and input layer nodes are fully connected.

After the topological relationship is determined, the calculation process begins and is roughly divided into several parts:

1. Initialization: Each node randomly initializes its own parameters. The number of parameters per node is the same as the input dimension.
2. For each input data, find the node that best matches it. Assuming that the input D is dimension, as a result, $X = \{x_i, i = 1, \dots, D\}$, then the discriminant function can be Euclidean distance:

$$d_j(x) = \sum (x_i - w_{ij})^2 \quad (4.59)$$

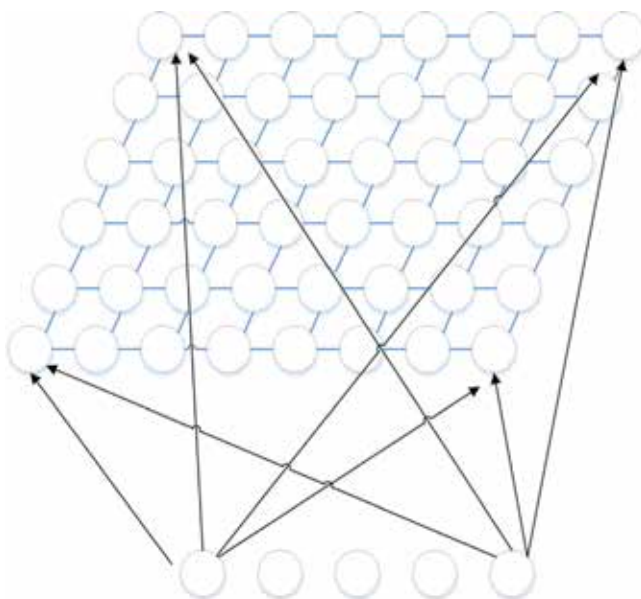


Figure 4.6.
SOM.

3. After finding the activation node $I(x)$, we also want to update its neighboring nodes. Let S_{ij} denote the distance between nodes i and j and assign an update weight to the nodes near $I(x)$:

$$T_{j,I(x)} = \exp(-s_{j,i(x)}^2/2\sigma^2) \quad (4.60)$$

4. Then update the node parameters. Update by gradient descent method:

$$\Delta w_{ji} = \eta(t) * T_{j,I(x)}(t) * (x_i - w_{ij}) \quad (4.61)$$

iterates until it converges.

Deep Learning

The concept of deep learning originates from the research of artificial neural networks. Multi-layer perceptions with multiple hidden layers are a deep learning structure. Depth learning represents the distributed characteristics of data by combining low-level features to form more abstract high-level representations of attribute categories or features [149].

The concept of deep learning was proposed in 2006 by Hinton et al. [150]. Based on the deep belief network (DBN) [15], the algorithm is proposed to solve the optimization problem of the deep structure, and then the deep structure of a multi-layer automatic encoder is proposed. In addition, the convolution neural network proposed by Lecun et al. [151] is the first real multi-layer structure learning algorithm, which uses the space relative relation to reduce the number of parameters to improve training performance. Deep learning is a method of representing learning in machine learning. Observations (such as an image) can be represented in a variety of ways, such as vectors for each pixel's strength value, or more abstracted into a range of edges, specific shapes, etc. It is easier to learn a task from an instance using some specific presentation methods (for example, face recognition or facial expression recognition). The advantage of deep learning is to replace the manual acquisition feature with non-supervised or semi-supervised feature learning and hierarchical feature extraction.

Deep learning is a new field in the study of machine learning. It is inspired by the learning process of human brain. It simulates human brain to illustrate the mechanism of data, such as images, sound and text. As with machine learning methods, deep machine learning methods also have supervised learning and unsupervised learning. The learning model established under different learning frameworks is very different. For example, convolutional neural network (CNN) is a supervised deep learning model, while deep belief network (DBN) is a model of unsupervised learning.

5.1. The biological mechanism of neurons

The basic unit of the neural network is the neuron, and the mathematical neuron model corresponds to the nerve cell in biology. Or, artificial neural network theory [152] uses the abstract mathematical model of neurons to describe the biological cells of the objective world. It is obvious that the nerve cell of biology is the foundation of the formation of neural network theory. Thus, the mathematical description of the neuron must be based on the objective behavior of the biological nerve cells. Therefore, it is important and necessary to understand the behavior characteristics of the biological nerve cells. The topology of a neural network is also based on the interconnection of nerve cells in biological anatomy. Disclosure of the interaction between nerve cells is also important. The neuron is the basic element of the neural network. Understanding the mechanism of neurons contributes to comprehend the nature of neural networks. In this section, biological solutions, information processing and transmission methods, working functions and mathematical models are given.

5.1.1 The biological anatomy of a neuron

In humans, the structure of neurons is not exactly the same; however, regardless of the structure, neurons are composed of some basic components. The neuron is composed of three parts: the cyton, the dendrite and the axon:

1. Cyton

The cell body is a complex composed of many molecules, including the nucleus, ribosome, and protoplast structure. It is a kind of energy supply of neuronal activity, where various biochemical processes such as metabolism are performed.

2. Dendrite

The branches of the extension of the cell body are called dendrites, which are the portals of information that receive messages from other neurons.

3. Axon

The longest protruding tubular fibers of the cell body are called axons. Axons can grow to 1 m or more. Axons are the exportation of excited neurons to other neurons.

At present, according to neurophysiological research, it has been found that neurons have four kinds of behavior. They can inhibit or excite, they can produce eruptions or have a calming effect, they can produce a recoil after transplantation and they are adaptive. In addition, there are four kinds of behavior: information synthesis, gradual change, electric contact and chemical contact, which can produce delayed excitation. Neurons for information processing and as a transmission mechanism have the following characteristics: they inhibit and excite, they pass threshold characteristics, they have information comprehensive features and they are exceptional at digital-to-analog conversion.

A synapse is a structure in which a neuron connects with another neuron and transmits information. As shown in **Figures 5.1** and **5.2**, synapses consist of pre-synaptic components, a synaptic cleft and postsynaptic components. The presynaptic component is one of the axons of a neuron. The gap between the synapse is the space between the presynaptic component and the posterior component, and the gap is usually 200–300 Å. The postsynaptic component can be a cell body, dendrite or axon. The presence of synapses suggests that the cytoplasm of two neurons is not

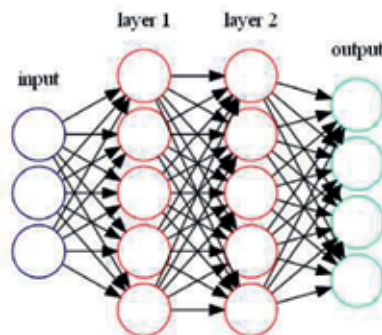


Figure 5.1.
Multi-layer feed-forward neural network.

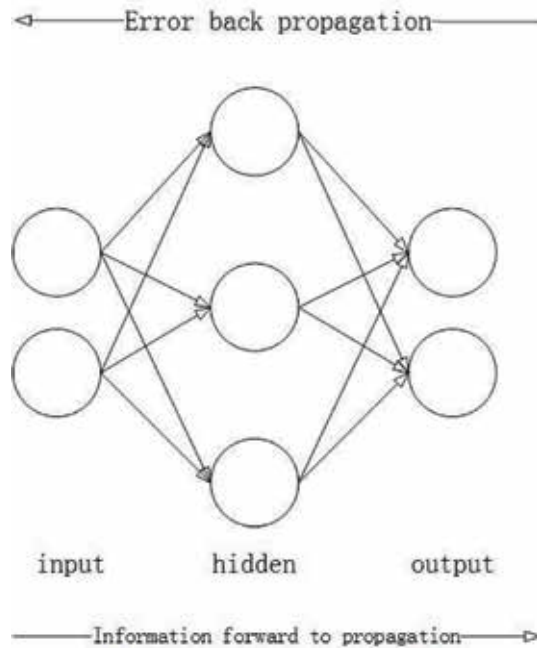


Figure 5.2.
 Back propagation.

connected directly, and that the two are connected through the structure of synapses. Sometimes, synapses can be connections between neurons.

5.2. Deep feed-forward neural network

5.2.1 Single-layer feed-forward neural network

A neural network is based on the mathematical model of neurons, and its model has a network topology, node characteristics and learning rules. It has four qualities: parallel distribution processing; a high degree of robustness and fault tolerance; distributed storage and learning ability; and it can fully approximate complex non-linear relationships.

According to the characteristics of neurons and biological function, neurons are a multi-input, single output, non-linear information processing unit.

The specific mathematical formula [152] is as follows:

$$\begin{cases} v = \sum_{i=1}^m x_i w_i + b \\ y = \phi(x) \end{cases} \quad (5.1)$$

The typical activation task has sigmoid, tanh, rectified linear unit (ReLU) and softplus functions, and the corresponding formula is:

$$\begin{cases} \text{sigmoid}(x) = \frac{1}{1+e^{-x}} \\ \text{tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \\ \text{ReLU}(x) = \max(0, x) \\ \text{softplus}(x) = \log(1 + e^x) \end{cases} \quad (5.2)$$

It is important to note that neural scientists found that neurons have unilateral inhibition, a wide excited boundary, sparse activation features compared with other activation functions and a ReLU function with biological interpretability; ReLU functions work better than others.

According to the topological structure of network connection, a neural network can be divided into a forward network (directed loop) and a feedback network (undirected complete graph, also called a cyclic network). For the feedback network, stability of the network model has a close relationship with associative memory. A network model of stability and associative memory has close relations. The Hopfield net and Boltzmann machine network belong to this type.

The forward network comes from the multiple coincidence of simple non-linear functions, and the network structure is simple and easy to implement. A brief introduction to a single layer hidden layer in feedforward neural networks is as follows, and the corresponding mathematical formula is:

$$\begin{cases} h^{(1)} = \phi^{(1)}(\sum_{i=1}^n x_i * w_i^{(1)} + b^{(1)}) \\ y = \phi^{(2)}(\sum_{j=1}^n h_j^{(1)} * w_j^{(2)} + b^{(2)}) \end{cases} \quad (5.3)$$

The input $x \in R^m$, the hidden layer output $h \in R^n$ and the outputs $y \in R^k$, $w^{(1)} \in R^{m*n}$ and $b^{(1)} \in R^n$ are the weights and biases of the input to the hidden layer; $w^{(2)} \in R^{n*k}$ and $b^{(2)} \in R^k$ are the weights and biases of the hidden layer to the output layer; and $\phi^{(1)}$ and $\phi^{(2)}$ are the activated functions for each layer.

In practice, suppose that the training data is set up as:

$$\begin{cases} \{x^{(2)}, y^{(2)}\}_{n=1}^N \\ x^n \in R^m \\ y^k \in R^k \end{cases} \quad (5.4)$$

The model between input and output is:

$$y = T(x, \theta) = \phi^{(2)} \left(\sum_{j=1}^n \phi^{(1)} \left(\sum_{i=1}^m x_i * w_i^{(1)} + b^{(1)} \right) * w_j^{(2)} + b^{(2)} \right) \quad (5.5)$$

Using parameter $\theta = (w^{(1)}, b^{(1)}; w^{(2)}, b^{(2)})$ to further optimize [153] the target function (loss, the regular item):

$$minL(\theta) = \frac{1}{N} \sum_{n=1}^N \|y^{(n)} - T(x^n; \theta)\|_F^2 + \lambda \sum_{i=1}^2 \|w^{(i)}\|_F^2 \quad (5.6)$$

Solve θ by gradient descent:

$$\begin{cases} \theta^k = \theta^{k-1} - \alpha * \nabla \theta|_{\theta=\theta_{k-1}} \\ \nabla \theta|_{\theta=\theta_{k-1}} = \frac{\partial(\theta)}{\partial \theta} \Big|_{\theta} = \theta^{k-1} \end{cases} \quad (5.7)$$

With the increase in the number of iterations k , the parameters will converge (the profile can be visualized by the target function $L(\theta_k)$):

$$\lim_{k \rightarrow \infty} \theta^{k-1} = \theta^* \quad (5.8)$$

The reason for convergence is that the above objective function is convex. For optimizing the target function (Eq. 5.6), but when the amount of data is large, storage and reading are very time-consuming, so the stochastic gradient descent is usually used to solve the problem. For the determination of the topology of the neural network, Hornik et al. show that: if the output layer adopts linear activation function, the hidden layer uses the sigmoid function, and the single hidden layer neural network can approximate any rational function with any degree of accuracy.

5.2.2 Multi-layer feed-forward neural network

Using the analysis of the single-hidden layer feed-forward neural network, when the number of hidden layers is more than two, it is called the multi-hidden layer feed-forward neural network [154], or the deep feed-forward neural network, whose network structure is shown in **Figure 5.1**.

It is necessary to point out that the topology of the deep feed-forward neural network is multi-layer, fully connected, and there is no ring. Based on **Figure 5.1**, the network input and output model is given by the following formulas:

input $x \in R^m$, output $y \in R^s$, the hidden layer's output is:

$$\begin{cases} h^{(l)} = \phi^{(l)}(\sum_{i=1}^{n_{l-1}} h_i^{(l-1)} w_i^{(l)} + b^{(l)}) \\ l = 1, 2, \dots, L \\ h^{(0)} = x \\ h^{(L)} = y \end{cases} \quad (5.9)$$

It is important to note that the input layer $h^{(0)}$, output layer $h^{(L)}$, the number of hidden layers of the $L - 1$ layer and the super parameter correspond (layer number, number of hidden units, activation function):

$$\begin{cases} L + 1 \rightarrow \text{numbers of layer(include input and output)} \\ [n_0, n_1, n_2, \dots, n_{L-1}, n_L] \rightarrow \text{dimension of every layer} \\ [\varphi^{(1)}, \varphi^{(2)}, \dots, \varphi^{(L-1)}, \varphi^{(L)}] \rightarrow \text{activate function} \end{cases} \quad (5.10)$$

Note that $n_0 = m$ and $n_L = s$. The parameter to be layer is:

$$\begin{cases} \theta = (\theta_1, \theta_2, \dots, \theta_L) \\ \theta_l = (w^{(l)} \in R^{n_{l-1} * n_l}, b^{(l)} \in R^{n_l}) \\ l = 1, 2, \dots, L \end{cases} \quad (5.11)$$

So, the relationship between input and output is:

$$\begin{aligned} y = h^{(L)} &= \varphi^{(L)} \left(\sum_{i_L=1}^{n_L} h_{i_L}^{(L-1)} * w_{i_L}^{(L)} + b^{(L)} \right) \rightarrow \text{remember as } \varphi^{(L)}(h^{(L-1)}, \theta_L) \\ &= \varphi^{(L)} \left(\sum_{i_L=1}^{n_L} \varphi^{(L-1)} \left(\sum_{i_{L-1}=1}^{n_{L-1}} h_{i_{L-1}}^{(L-2)} * w_{i_{L-1}}^{(L-1)} + b^{(L-1)} \right) w_{i_L}^{(L)} + b^{(L)} \right) \\ &\rightarrow \text{remember as } \varphi^{(L)}(\varphi^{(L-1)}(h^{(L-2)}, \theta_{L-1}), \theta_L) \\ &= \dots = \varphi^{(L)}(\varphi^{(L-1)}(\dots \varphi^{(1)}(x, \theta_1) \dots, \theta_{L-1}), \theta_L) \rightarrow \text{remember as } f(x, \theta) \end{aligned} \quad (5.12)$$

In application, the train data set is:

$$\begin{cases} \{x^{(2)}, y^{(2)}\}_{n=1}^N \\ x^n \in R^m \\ y^n \in R^K \end{cases} \quad (5.13)$$

The optimized objective function (loss and regularization) is:

$$\text{min}_j(\theta) = L(\theta) + \lambda R(\theta) \quad (5.14)$$

inside $\hat{y}_n = f(x_n, \theta)$ and:

$$\begin{cases} l(y_n, \hat{y}_n) = \|y_n - \hat{y}_n\|_F^2 \\ L(\theta) = \frac{1}{N} \sum_{n=1}^N l(y_n, \hat{y}_n) \\ R(\theta) = \sum_{l=1}^L \|\theta_l\|_F^2 = \sum_{l=1}^L \|w^{(L)}\|_F^2 \end{cases} \quad (5.15)$$

Loss functions come in many forms: energy loss, cross-entropy, etc. The regular item R has coefficient regularities (analog biological characteristics) except for use of the Fubius norm (to prevent overfitting).

5.2.3 Back propagation algorithm

How do you solve the optimization target function formula? First, the convexity and non-convexity of the objective function are determined. If the parameter selection range is a convex set, the convex function defined on the set is convex optimization. The obtained solution does not depend on the initial value selection, and is the global optimal solution. The optimal objective function of the deep feedforward neural network is generally non-convex, so the solution of the parameters depends on the setting of the initial parameters. If the settings are reasonable, you can avoid prematurely falling into the local optimal solution. The specific description of the backpropagation algorithm [155] is as follows:

$$\begin{cases} \theta^k = \theta^{k-1} - \alpha * \nabla \theta|_{\theta=\theta^k} \\ \nabla \theta|_{\theta=\theta^k} = \frac{\partial L(\theta)}{\partial \theta} + \lambda \frac{\partial R(\theta)}{\partial \theta} \end{cases} \quad (5.16)$$

where α is the learning rate. Updating the parameters on each layer gives:

$$\begin{cases} \theta_l^k = \theta_l^{k-1} - \alpha * \nabla \theta_l|_{\theta_l=\theta_l^{k-1}} \\ \nabla \theta_l|_{\theta_l} = \theta_l^{k-1} = \frac{\partial L(\theta)}{\partial \theta} + \lambda \frac{\partial R(\theta)}{\partial \theta} \end{cases} \quad (5.17)$$

In this case, θ_l^k is the update of the k iteration of the l th layer. For the solution of the core gradient descent, the error propagation term is introduced. According to the chain rule, it is expanded as:

$$\frac{\varphi L(\theta)}{\varphi \theta_l} = \frac{\varphi h^l}{\varphi \theta_l} * \frac{\varphi h^{l+1}}{\varphi h^l} * \dots * \frac{\varphi h^L}{\varphi h^{L-1}} * \frac{\lambda L(\theta)}{\varphi h^L} \quad (5.18)$$

The error propagation is recorded as:

$$\delta^l = \frac{\varphi L(\theta)}{\varphi h^l} \quad (5.19)$$

Furthermore, using $\theta_l = (w^l + b^l)$, the implicit layer output is the derivative of the corresponding reciprocal:

$$\begin{cases} \frac{\partial h^l}{\partial w^l} = \frac{\partial \varphi^l((h^{l-1})^T * w^l + b^l)}{\partial w^l} = h^{l-1} \cdot (\varphi^l)' \\ \frac{\partial h^l}{\partial b^l} = \frac{\partial \varphi^l((h^{l-1})^T * w^l + b^l)}{\partial b^l} = 1 \cdot (\varphi^l)' \end{cases} \quad (5.20)$$

The \cdot is the Hadamard product from Eq. (5.18) and is the derivative of the loss with respect to the parameter. The derivative of the regular item about the parameter is:

$$\frac{\partial R(\theta)}{\partial \theta_l} = \frac{\partial}{\partial \theta_l} \left(\sum_{l=1}^L \|\theta_l\|_F^2 \right) = \frac{\partial \|\theta_l\|_F^2}{\partial \theta_l} \quad (5.21)$$

Normally, the constraints in the regular item are only for the weight matrix, and the binary values are not regular constraints. So, there are:

$$\begin{cases} \frac{\partial R(\theta)}{\partial w^l} = \frac{\partial \|w^l\|_F^2}{\partial w^l} = 2w^l \\ \frac{\partial R(\theta)}{\partial b^l} = \frac{\partial \|b^l\|_F^2}{\partial b^l} = 0 \end{cases} \quad (5.22)$$

The core meaning of the back propagation algorithm is to use the optimization objective function $J(\theta)$ to find the gradient of parameters θ in the l th hidden layer. The gradient is determined by the loss term $L(\theta)$, the regular term $R(\theta)$, parameter θ_l of the l th hidden layer, where the back propagation of the error is achieved by introducing an error propagation term.

So, a feed-forward neural network consists of two steps: one is based on the current parameter values, forward propagation on each layer in the process of the output value. Another step is to backpropagate the error propagation term on each layer by the difference between the actual output and the expected output, and combine the output with the partial derivative of the parameter of each layer to achieve an update of each layer parameter. Repeat these two steps until the process converges.

However, when the number of network layers is deep, the gradient increment of the parameters of each layer will attenuate with the propagation process from output to input, making it difficult for the whole network to obtain better results through training and fall into the local optimal solution. This is called diffusion problem.

5.3. Deep convolution neural network

A convolution feed-forward neural network [156, 157] has a special kind of depth to avoid a hierarchy between the connection parameters caused by redundancy, leading to the training of the network model, which depends on the amount of data that is equal to the number of parameters.

This local connection design choice, in line with the coefficient of response characteristics of biological neurons, can greatly reduce the scale of the network model parameters, and dependence on the amount of training data.

5.3.1 The mathematical representation of convolution

The base modules of convolution neural networks are convolution flows, including convolution (for dimension expansion), non-linearity (sparseness, saturation, side suppression), pooling (aggregation of control or feature types), and batch normalization (optimization operation, the purpose of which is to speed up the training process of convergence speed, while avoiding the local optimum).

A. Convolution

In mathematics, convolution is an important linear operation. There are three main types of convolution in digital signal processing: full convolution, same convolution and valid convolution. Suppose the input signal is a one-dimensional signal, $x \in R^n$; the filter is also one dimensional, $w \in R^m$.

1. Full convolution

$$\begin{cases} y = conv(x, w, 'full') = (y(1), \dots, y(t), \dots, y(n + m - 1)) \in R^{n+m+1} \\ y(t) = \sum_{i=1}^m x(t + i - 1)w(i) \end{cases} \quad (5.23)$$

where $t = 1, 2, \dots, n + m - 1$.

2. Same convolution

$$y = conv(x, w, 'same') = center(conv(x, w, 'full'), n) \in R^n \quad (5.24)$$

3. Valid convolution

$$\begin{cases} y = conv(x, w, 'valid') = (y(1), \dots, y(t), \dots, y(n - m + 1)) \in R^{n-m+1} \\ y(t) = \sum_{i=1}^m x(t + i - 1)w(i) \end{cases} \quad (5.25)$$

where $t = 1, 2, \dots, n - m + 1, n > m$.

In convolution flow, valid convolution is commonly used. Valid convolution is illustrated in **Figure 5.3**.

The core of the convolution operation is that some weight connections can be removed. The introduction of sparse or local connections reduces parameters

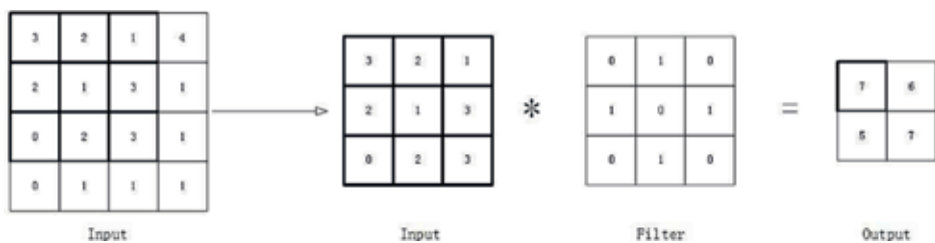


Figure 5.3.
Valid convolution operation.

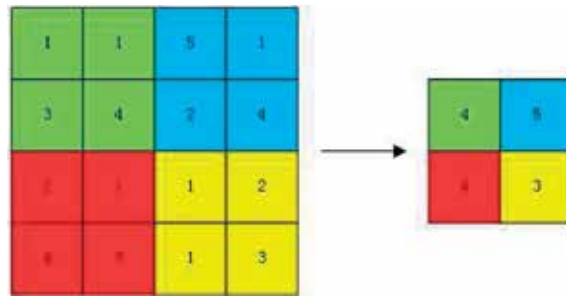


Figure 5.4.
Max pooling.

through the weight sharing strategy, thus. Since the convolution operation has translation invariance, the learned feature has the characteristics of topological correspondence and robustness.

B. Pooling

In essence, the pooling operation performs control of the feature type, and reduces spatial dimension. The main significance is to reduce the amount of computation to describe the translation invariance, to reduce the input dimension of the next layer and to control the risk of over-fitting. The operation of the pool varies between the average pool, the maximum pool, the number of pools and the probability of pooling, etc.; max pooling is commonly used (**Figure 5.4**).

In addition to the pooling methods listed above, there is spatial pyramid pooling, which is a multi-scale pooling method that can obtain input multi-scale information; another spatial pyramid pool can be any scale image of the convolution characteristics into the same latitude, which not only allows the convolution of the neural network to deal with any scale of the picture, but also avoids the cropping and wrapping operation caused by the loss of information, which has a very important significance.

The spatial pyramid pooling [158] method is used as much as possible in the final convolution stream to avoid the loss of information from the previous stretching or vectorization. Examples of spatial pyramid pooling can be found in **Figure 5.5**.

C. Activation function

The core of the activation function is that the complex non-linearity of the whole network is improved by simple non-linear mapping. If there is no non-linear operation in the network, there are more hierarchical combinations to consider: linear approximation, characterization or mining of high-level semantics whose features are limited (**Figures 5.6** and **5.7**).

In applications, the commonly used activation functions are: ReLU, which can accelerate convergence, sparse content, SoftMax (for the last layer, does not calculate probability response), softplus (ReLU smooth approximation) and the sigmoid function. The core of the traditional neural network includes the logistic-sigmoid function and the tanh-sigmoid function.

D. Batch normalization

With regard to the normalization operation, the aim is to avoid the tendency of information transmission to be attenuated step by step with the deepening of the hierarchy because the input of the data range in the pattern classification may be too large and the input range of the data range may be small. In short, the data range is too large or too small, and may lead to deep neural network convergence, slow and lengthy training time, a commonly used normalized operation with L2 normalization and sigmoid function normalization. It should be noted that the convolution

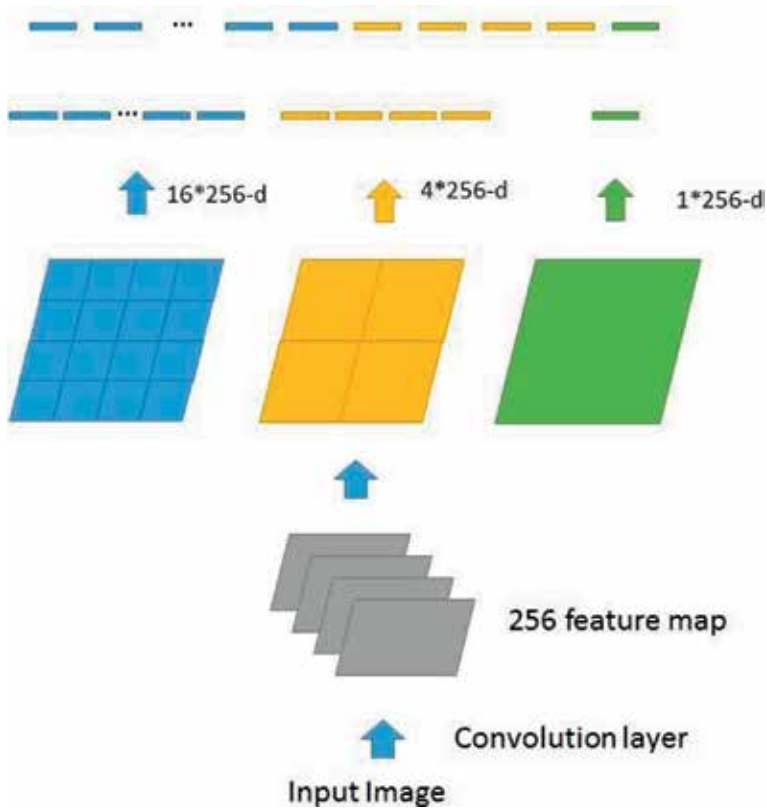


Figure 5.5.
Spatial pyramid pooling.

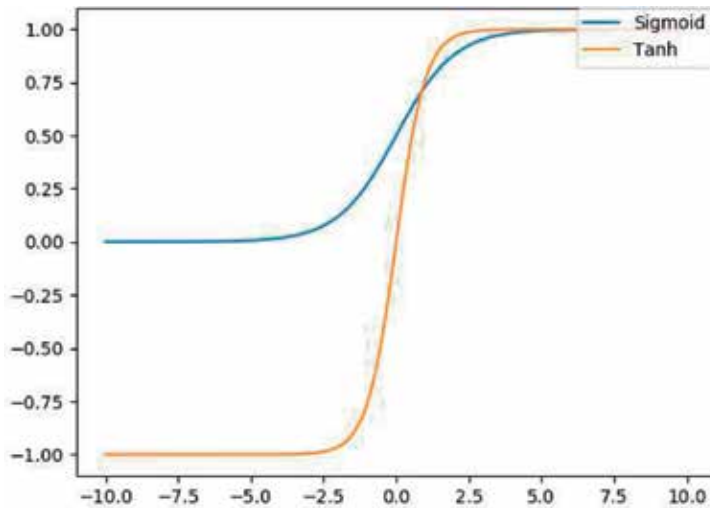


Figure 5.6.
Sigmoid—the core of the traditional neural network.

of the neural network sometimes uses a variety of normalized layers, but recent studies have shown that this level seems to help very little of the final result, so many networks have abandoned this operation.

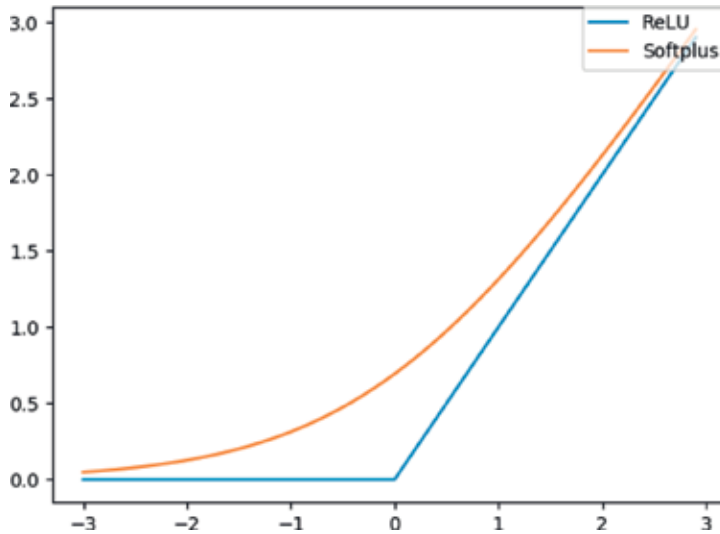


Figure 5.7.
Softplus function and ReLU function.

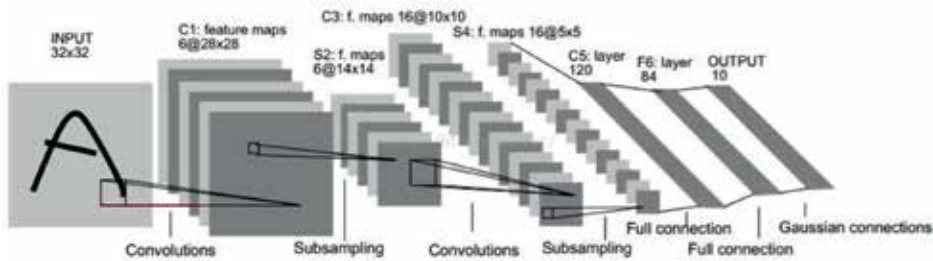


Figure 5.8.
LeNet [151].

5.3.2 Classical CNN structure

First, we need some of the classic CNN structures.

5.3.2.1 LeNet-5

LeNet-5 is a classic structure to a CNN, which is mainly used for handwriting recognition.

The input to the network is 32×32 handwritten font images, which contain 0 to 9 digits, which is equivalent to 10 categories. The output of the network is a number between 0 and 9. Therefore, we can see that this is a multi-classification problem, that is, the number of neurons in the output layer is 10, and the final output layer of the neural network must be a SoftMax problem. The structure of LeNet-5 is shown in **Figures 5.8**.

Input layer: a 32×32 picture, which is equivalent to 1024 neurons.

C1 layer: paper author, select the six characteristic convolution kernels, and then the volume of the convolution kernel. The size of the convolution kernel is 5×5 , so we can get six feature maps, and then the size of each feature graph is $32 - 5 + 1 = 28$. Also the number of neurons is reduced from 1024 to $28 \times 28 = 784$.

S2 layer: this is the downsampling layer, which is applied by using the max pooling. The pooled size is (2, 2), and the picture with size of 28 * 28 in the C1 layer is divided into blocks of size 2 * 2. This way we can get 14 * 14 blocks and then use the largest value in each 2 * 2 block as the new pixel. Then we can get the result of the S1 layer: a picture with a size of 14 * 14, a total of six.

C3 layer: convolution layer, with this layer we choose the size of the convolution kernel to be 5 * 5, so we can get a new picture size of 14 - 5 + 1 = 10, and then we hope to get a 16 feature map. But the problem here is this is the most difficult to understand; we know S2 contains six 14 * 14 sizes of the picture, and we hope that this layer is the result: 16 pictures of 10 * 10.

S4 layer: the lower sampling layer is relatively simple, but is also connected to the C3 16 10 10 picture for the maximum pool. The pool block size is 2 2, and is the last S4 layer for the 16 size of 5 5 pictures. At this point our number of neurons has been reduced to 16 * 5 * 5 = 400.

C5 layer: We continue to convolute with a 5 5 convolution kernel, and then we want to get 120 feature graphs. The size of the C5 layer image is 5 - 5 + 1 = 1, which is equivalent to one neuron, a 120 feature map, so there are only 120 neurons left. This time, the number of neurons is sufficient, and later we can directly use the fully connected neural network for 120-neuron follow-up processing.

5.3.2.2 AlexNet

AlexNet [21] is a historic network structure. Prior to AlexNet, research on network depth has been stalled for a long time. However, in 2012, AlexNet won the ImageNet Image Classification Competition that year, the top 5 error rate dropped by 10 percentage points comparing with the previous year's champion, and far exceeded the second place of the year.

AlexNet has shown that the depth of the network can greatly improve the performance of network.

AlexNet has four innovations which make the deeper network achieve better results: 1. Using non-linear activation function: ReLU. 2. Using following methods to prevent overfitting: dropout, data augmentation. 3. Applying a large amount of data in training: million ImageNet image data. 4. Other: Multiple GPUs for distributed computing, and local response normalization (LRN) layer to normalize the input. The structure of AlexNet is shown in **Figure 5.9**.

Here's a brief introduction to some of AlexNet's specifics.

A. Data augmentation

There is a view that the neural network is fed by the data. If there is an increase in training data, you can improve the accuracy of the algorithm, because it can avoid overfitting, and therefore increase your network structure. When the training

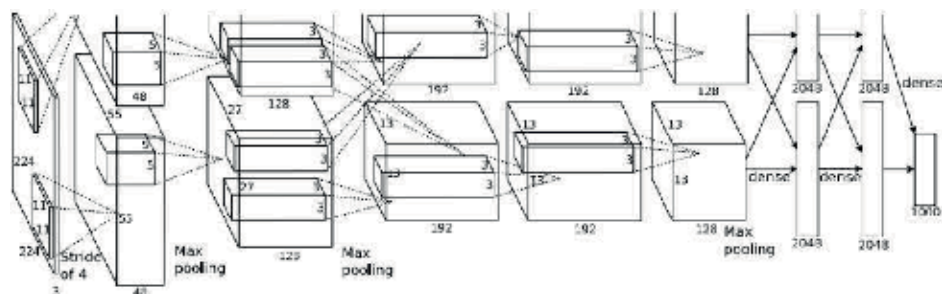


Figure 5.9.
AlexNet.

data is limited, some transformations can be used to generate new data from the existing training data set to expand the size of the training data.

Among them, the simplest way to achieve image data deformation is:

1. From the original image (256, 256), randomize some of the images (224, 224). (Translate)
2. Horizontal flip image. (Reflex)
3. Add some random light to the image. (Light, color transformation, color jittering)

B. ReLU activation function

Sigmoid is a commonly used non-linear activation function that compresses the input to a continuous real value between 0 and 1. In particular, if it is a very large negative number, then the output is 0; if it is a very large positive number, then the output is 1. But this has some fatal flaws.

Sigmoids saturate and kill gradients. Worse than this, when the input is very large or very small, there will be saturation; the gradient of these neurons is close to 0. If your initial value is large, the gradient is in the reverse propagation time because you need to multiply a sigmoid derivative. This will make the gradient smaller and smaller, which will lead to the network becoming difficult to learn.

The mean of the output of sigmoid is not 0, which is not desirable because it causes the neurons in the next layer to get the non-zero mean signal from the previous layer as input. One result is that if the data is positive when it is input to the neuron, then the calculated gradient will always be positive. But if you are training by batch, then each batch may get a different signal, so this problem can be alleviated. In general, the non-zero mean problem has some bad effects, but it is much better than the kill gradients mentioned above.

The mathematical expression of ReLU is:

$$f(x) = \max(0, x)$$

C. Dropout

Combining many different pretrained models for prediction is a very successful way to reduce test errors. But it takes time for each model to be trained, so it is too expensive for large-scale neural networks.

However, AlexNet [21] proposes a very effective combination of model versions, which only needs twice as much time as a single model in training. This technique is called Dropout [159], and is done with a probability of 0.5; the output of each hidden layer of neurons is set to zero. In this way, the “unfold” neurons are neither involved in the forward transmission nor participate in the reverse transmission.

So, each time you enter a sample, it is equivalent to the neural network trying a new structure, but all these structures share weight. Because neurons cannot rely on other specific neurons to exist, this technique reduces the complex relationship between neurons.

For this reason, the network needs to be forced to learn more robust features that are useful when combined with different random subsets of other neurons. At the time of the test, we multiply the output of all neurons by only 0.5, which is a reasonable approximation for obtaining the geometric mean of the predicted distribution of the exponential dropout network.

D. Local response normalization

In general, this layer is also intended to prevent the activation function from saturating.

Its principle is to make the activation function avoid saturation by regularization, and obtain a larger derivative value, and its function is similar to ReLU. From the experimental results, the LRN operation can improve the generalization ability of the network and reduce the error rate by about 1%.

5.4. Autoencoders and sparsity network

So far, we have described the application of neural networks to supervised learning, in which we have labeled training examples. Now suppose we have only a set of unlabeled training examples $\{x^1, x^2, x^3, \dots\}$ where $x^i \in \mathcal{R}^n$. An autoencoder neural network [160] is an unsupervised learning algorithm that applies back propagation, setting the target values to be equal to the inputs, i.e., it uses $y^i = x^i$.

Figure 5.10. is an autoencoder.

The autoencoder tries to learn a function $h_{W,b}(x) \approx x$. In other words, it tries to learn the approximate equality function of the output \hat{x} that is similar to x . An identification function seems to be a particularly important function to try to learn, but by limiting the network, for example by limiting the number of hidden units, we can find interesting structures about data. As a concrete example, suppose the inputs x are the pixel intensity values from a 10×10 image (100 pixels) so $n = 100$, and there are $s_2 = 50$ hidden units in layer L_2 . Note that we also have $y \in \mathcal{R}^{100}$. Since there are only 50 hidden units, the network is forced to learn the compressed representation of the input. That is, we only give vectors that activate hidden units $a^{(2)} \in \mathcal{R}^{50}$; it must try to reconstruct the 100-pixel input x . If the input were completely random—say, each x_i comes from an IID Gaussian independent of the other features—then this compression task would be very difficult. However, if there is structure in the data, for example if some of the input features are related, then the algorithm will be able to find some of these dependencies. In fact, the low-dimensional representation of this simple automatic encoder that is often learned by principal component analysis (PCA) is very similar.

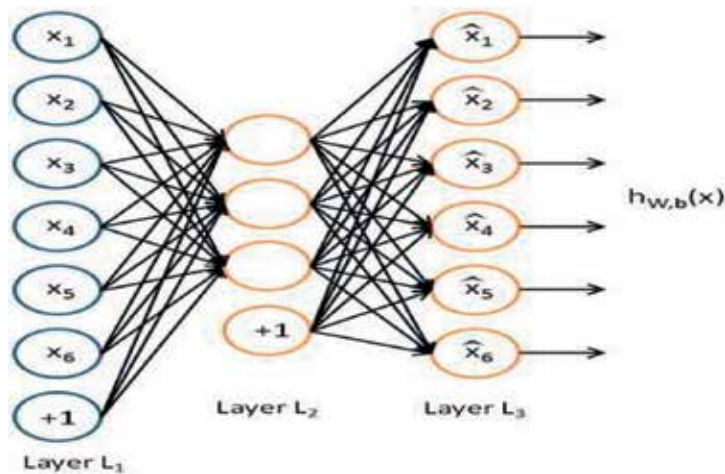


Figure 5.10.
Autoencoder.

Our argument relies on s_2 small hidden units. However, even though the number of hidden units is large (even larger than the number of input pixels), we can still find interesting structures by imposing other restrictions on the web. In particular, if we impose hidden units of sparse constraints, then self-coding will still find the interesting structure of the data, even if the number of hidden units is large.

To put it in a nutshell, we think neurons are “active” (or “triggered”) if the output is close to 1 or the output is close to 0. We want to limit neuronal inactivity most of the time. This discussion assumes a sigmoid activation function. If you are using a hyperbolic tangent function, then we think that the neuron is invalid, and the output value is close to 1.

Recall that $a_j^{(2)}$ denotes the activation of hidden unit j in the autoencoder. However, this notation doesn’t make explicit what the input x was that led to that activation. Thus, we will write $a_j^{(2)}(x)$ to denote the activation of this hidden unit when the network is given a specific input x . Furthermore, let:

$$\hat{\rho}_j = \frac{1}{m} \sum_{i=1}^m [a_j^{(2)}(x^{(i)})] \quad (5.26)$$

be the average activation of hidden unit j (averaged over the training set). We would like to (approximately) enforce the constraint:

$$\hat{\rho}_j = \rho,$$

where ρ is a sparsity parameter, typically a small value close to zero (say $\rho = 0.05$). In other words, we would like the average activation of each hidden neuron j to be close to 0.05 (say). To satisfy this constraint, the hidden unit’s activations must mostly be near 0.

To achieve this, we will add an extra penalty term to our optimization objective that penalizes $\hat{\rho}_j$ deviating significantly from ρ . Many choices of the penalty term will give reasonable results. We will choose the following:

$$\sum_{j=1}^{s_2} \rho \log \frac{\rho}{\hat{\rho}_j} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_j}.$$

Here, s_2 is the number of neurons in the hidden layer, and the index j is summing over the hidden units in our network. If you are familiar with the concept of Kullback–Leibler (KL) divergence, this penalty term is based on it, and can also be written:

$$\sum_{j=1}^{s_2} \text{KL}(\rho \parallel \hat{\rho}_j),$$

where $\text{KL}(\rho \parallel \hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_j}$ is the KL divergence between a Bernoulli random variable with mean ρ and a Bernoulli random variable with mean $\hat{\rho}_j$. KL divergence is a standard function for measuring how different two distributions are. (If you have not seen KL divergence before, don’t worry about it; everything you need to know about it is contained in this text.)

This penalty function has the property that $\text{KL}(\rho \parallel \hat{\rho}_j) = 0$ if $\hat{\rho}_j = \rho$, and otherwise it increases monotonically as $\hat{\rho}_j$ diverges from ρ . For example, in **Figure 5.11**, we have set $\rho = 0.2$, and plotted $\text{KL}(\rho \parallel \hat{\rho}_j)$ for a range of values of $\hat{\rho}_j$.

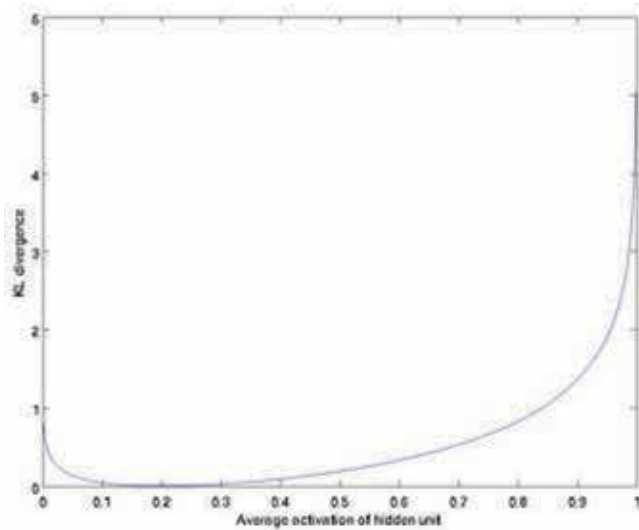


Figure 5.11.
KL penalty example.

We see that the KL divergence reaches its minimum of 0 at $\hat{\rho}_j = \rho$, and blows up (it actually approaches ∞) as $\hat{\rho}_j$ approaches 0 or 1. Thus, minimizing this penalty term has the effect of causing $\hat{\rho}_j$ to be close to ρ .

Our overall cost function now is:

$$J_{\text{sparse}}(W, b) = J(W, b) + \beta \sum_{j=1}^{s_2} \text{KL}(\rho || \hat{\rho}_j),$$

where $J(W, b)$ is as defined previously, and β controls the weight of the sparsity penalty term. The term $\hat{\rho}_j$ (implicitly) depends on W, b also, because it is the average activation of hidden unit j , and the activation of a hidden unit depends on the parameters W, b .

To incorporate the KL divergence term into your derivative calculation, there is a simple-to-implement trick involving only a small change to your code. Specifically, where previously for the second layer ($l = 2$), during back propagation you would have computed:

$$\delta_i^{(2)} = \sum_{j=1}^{s_2} W_{ji}^{(2)} \delta_j^{(3)} f'(\mathbf{z}_i^{(2)}),$$

you now instead compute:

$$\delta_i^{(2)} = \sum_{j=1}^{s_2} W_{ji}^{(2)} \delta_j^{(3)} \left(+ \beta \left(\left(\frac{\rho}{\hat{\rho}_i} + \frac{1 - \rho}{1 - \hat{\rho}_i} \right) \right) f'(\mathbf{z}_i^{(2)}) \right).$$

One subtlety is that you'll need to know $\hat{\rho}_i$ to compute this term. Thus, you'll need to compute a forward pass on all the training examples first to compute the average activations on the training set before computing back propagation on any example. If your training set is small enough to fit comfortably in computer memory (this will be the case for the programming assignment), you can compute forward passes on all your examples and keep the resulting activations in memory

and compute the $\hat{\rho}_i$ s. Then you can use your precomputed activations to perform back propagation on all your examples. If your data is too large to fit in memory, you may have to scan through your examples computing a forward pass on each to accumulate (sum up) the activations and compute $\hat{\rho}_i$ (discarding the result of each forward pass after you have taken its activations $a_i^{(2)}$ into account for computing $\hat{\rho}_i$). Then after having computed $\hat{\rho}_i$, you'd have to redo the forward pass for each example so that you can do back propagation on that example. In this latter case, you would end up computing a forward pass twice on each example in your training set, making it computationally less efficient.

The full derivation showing that the algorithm above results in gradient descent is beyond the scope of these notes. But if you implement the autoencoder using back propagation modified this way, you will be performing gradient descent exactly on the objective $J_{\text{sparse}}(W, b)$. Using the derivative checking method, you will be able to verify this for yourself as well.

5.5. Sparse deep neural network

5.5.1 Data sparsity

Data sparsity [161] has the following three properties: First, the data contains some topology characteristics, or the target's non-zero elements are relatively small relative to the data. Second, the corresponding representation of the data under some (linear or non-linear) adaptive or non-adaptive transform is sparse and has fewer non-zero elements. Third, as the size of the data set increases, data with some statistical or physical characteristics accounts for a small number of the entire data set. For example, samples with particularly good resolution or samples with particularly poor resolution have a small proportion in the entire data set.

5.5.2 Sparse regularity

The supervision and machine-learning problem will “minimize your error while regularizing your parameters,” namely the regularization parameter will minimize the error at the same time.

The minimization error allows our model to fit our training data, and the regularization parameter prevents our model from overfitting our training data. Because there are too many parameters, this can lead to an increase in model complexity, which is easy to fit, but means that our training error will be very small. However, the training error is not our ultimate goal; our goal is to ensure that the model's test error is small, which will predict the new sample accurately. So, we need to make sure that the “simple” model is based on minimizing the training error, therefore parameters will have a good generalization performance (i.e., the test error is small), and the model of “simple” is done by rule function.

In addition, the use of regular terms can also constrain the characteristics of our model. In this way, the prior knowledge of the model can be exploited during learning, and then the model can be artificially made with the characteristics we want, such as sparseness, low rank, and smoothness. There are several ways of looking at regularization. Regularization is consistent with Occam's razor. The idea is that in all possible models, we should choose to be able to interpret the known data very well with very simple models. From Bayesian estimation, rule-based entry corresponds to the prior probability of the model. Regularization is the realization of structural risk minimization strategy, which is to add a regularizer or penalty

term to the empirical risk. In general, supervised learning can be seen as minimizing the following objective function:

$$w^* = \operatorname{argmin}_i \sum L(y_i, f(x_o; w)) + \lambda \omega(w) \quad (5.27)$$

where the first term, $L(y_i, f(x_o; w))$, measures the error between our model (classification or regression) for the predicted value of the i th sample $f(x_i; W)$ and real label y_i of the previous error. Because the model is to maximally fit the training samples, this term should be minimized. But not only to ensure the training error of the model is the smallest, but also to ensure that the test error of model is small. So, we need to add the second item, which is the regularization function $\omega(w)$ of the parameter w , to constrain the model to make it as simple as possible.

Most of the models of machine learning are similar to this and are just the transformation of these two terms. For example, you can replace the first loss function with square loss, then it becomes a least squares problem.

The regulation function is $\omega(w)$ and there are plenty of options, In general, the more complex the model, the larger the regularization value. For example, the regularization term can be the norm of the model parameter vector. Different choices have different constraints on the parameter w , and the effects obtained are different. Common norms are: L0 norm, L1 norm, L2 norm, trace norm, Frobenius norm and norm.

Usually, we use L0 and L1 regularization to make the data sparse.

The L0 norm is the number of elements in the non-zero vector. If we use the L0 norm to rule a parametric matrix W , we hope that most of the elements of W are 0. Let the parameter W be sparse:

$$R(\theta) = \|W\|_0 \quad (5.28)$$

The L1 norm is the sum of the absolute values of each element in a vector, which is called the “sparse rule operator” (Lasso regularization):

$$R(\theta) = \|W\|_1 \quad (5.29)$$

The L2 norm refers to the square root to the sum of the squares of the elements in the vector

To further illuminate this problem: for any regularization operator, if it is not divisible at $W_i = 0$, and can be decomposed into a form of “summation”, then this regularization operator can achieve the effect of sparseness.

Although the L1 norm and the L0 norm can achieve sparsity, the application of L1 is more extensive.

There are two reasons for this. First, because the L0 norm is difficult to solve for optimization, and second, the L1 norm is the optimal convex approximation of the L0 norm, and it is easier to optimize than the L0 norm. Sparseness is widely used because it has two following advantages.

1. Feature selection

Sparse regularization enables automatic selection of features. In general, most of the elements (features) in the input x_i have nothing to do with the final output y_i . Considering these additional features in x_i while minimizing the objective function, although smaller training errors can be obtained, these features can cause interference when predicting new samples. The introduction of the sparse

regularization operator is to complete the feature selection. It removes the useless features, which is to reset the weights corresponding to these features to zero.

2. Interpretability

Another reason for sparseness is that the model is easier to explain. So, for example, the probability of getting a certain disease is y , and then the data that we're collecting is 1000 dimensions, which is what we need to look for in terms of how the 1000 factors affect the probability of getting the disease. So, let's say that this is a regression model: $y = w_1 * x_1 + w_2 * x_2 + \dots + w_{1000} * x_{1000} + b$ (of course, to limit y to $[0, 1]$, we have to add a logistic function).

5.5.3 Sparse connection

The convolution neural network includes local connection, weight sharing, translation invariance and all sparseness. First, for local connection, compared with the full connection strategy, Convolutional neural networks have local connections, weight sharing, and translation invariance. First of all, for local connections, compared with full connections, it has the characteristics of sparse response. Second is the weight sharing, the hidden layer has the same activation characteristics, which can reduce the number of parameters, and Translation invariance can be obtained by pooling. Moreover, the classic dropout operation, as shown in (Figure 5.12), can implement sparse connection. The way to implement it is to make some hidden layer node weights temporarily not working when training the model. The inactive nodes do not calculate, but retain their weights.

5.5.4 Sparse classifier

Common sparse classifiers are based on learning, such as sparse representation of the classifier. The core steps are as follows:

1. Construction of the dictionary:

$$D = [D_1, D_2, \dots, D_k] \quad (5.30)$$

where k is the number of categories, D_k is the construction of the k th class sample or data set and the sample x is represented as follows:

$$\min \frac{1}{2} \cdot \|x - D \cdot \alpha\|_2^2 + \lambda \cdot \|\alpha\|_1 = \frac{1}{2} \left\| x - \sum_{k=1}^K D_k \cdot \alpha_k \right\|_2^2 + \lambda \cdot \|\alpha\|_1 \quad (5.31)$$

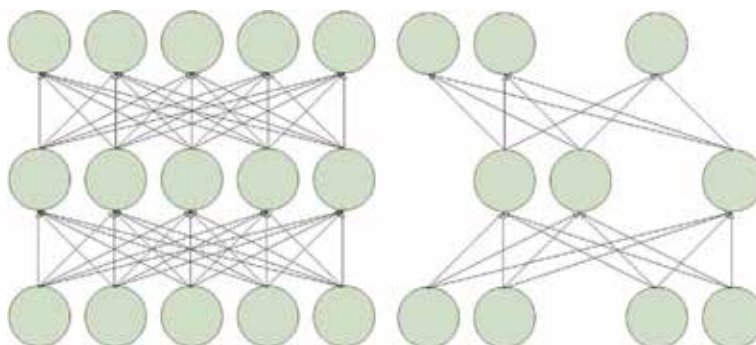


Figure 5.12.
 Dropout connection.

The indicated coefficients are as follows:

$$\alpha = [\alpha_1, \alpha_2, \dots, \alpha_k]^T \quad (5.32)$$

The final class is determined by the following formula:

$$\text{label}(x) = \arg \min_{1 \leq k \leq K} \{\|x - D_k \cdot \alpha_k\|_2^2\} \quad (5.33)$$

5.6. Generative adversarial nets

5.6.1 Generative model

Machine learning models can be broadly divided into two categories: generative model and discriminative model [162]. The discriminative model requires the input variable x to predict $p(y|x)$ by a certain model. The generative model is given some kind of implicit information to randomly generate the observed data. For example. Discriminant model: Determine whether the animal in a picture is a cat or a dog. Generative model: Give a picture of a series of cats and generate a new cat.

Using the well-known imagenet-1000 image classification, automatic driving of the image semantic segmentation, the prediction of human skeleton points belong to the discriminative model, that is, it gives the input to predict a feature. In fact, most of the work of the past 12 to 14 years belongs to the discriminative model. Why? Because one of the reasons is to determine the loss function of the model (loss) to facilitate the definition.

So what is machine learning? In a word, giving feedback during the training process, to make the results match our expectations. For the classification problem, we want the loss to not change after approaching bound, so we choose cross entropy as feedback. In the regression, we want the loss to remain the same only when the two are the same, so we choose the Euclidean distance between the points as the feedback. The choice of loss function (feedback) will obviously affect the quality of the training results, which is the top priority of the design model. In the past few years, there have been hundreds of variants of neural networks, but there very few loss functions are presented.

For discriminant models, the loss function is easy to define because the goal of the output is simple. But for generative models, the definition of loss functions is not that easy. For the task of generating a cat picture, if you simply define the loss function as “European distance from the original picture”, the result will be terrible. When we want the neural network to generate a picture of a cat, we hope that this picture has an animal’s outline, textured hair, rather than a simple Euclidean distance optimal solution. How do we put our expectations for cats into the model? This is the problem solved by the generative adversarial network (GAN).

5.6.2 Adversarial

For image classification, a bunch of RGB pixel points and N categories of probability distribution models are difficult to define from a traditional mathematical perspective. Then why not give the feedback of the generative model to the discriminant model? This is Goodfellow’s idea - he closely combines the two broad categories of machine learning, generative and discriminant.

A GAN has two parts in it: the generator “G” that generates images and the discriminator “D” that classifies real and fake images.

1. Input noise (hidden variable) z
2. Through generator G , we get fake data and refer to it as $x_{fake} = G(z)$
3. Take a part of data from real data set, and denoted by x_{real}
4. By mixing real data and fake data, we get $x = x_{fake} + x_{real}$
5. Enter data into discriminator D and tag the data as $x_{fake} = 0, x_{real} = 1$ (a simple binary classifier)
6. According to the classification results, return loss

In the process above, the discriminator D should try to make $D(G(z)) = 0$, and $D(x_{real}) = 1$, which is to correctly discriminate the fake images from real ones. However, generator G would try to make $D(G(z)) = 1$, which is to make the generated image as fake as possible. Therefore, the whole training process is like two players fighting each other, which is the source of the name Adversarial.

From a perspective of artificial intelligence, this model is the first to use two mutually constrained neural networks to guide neural networks. But the simple requirements make the training process in GAN easily lose direction.

These phenomena are actually three difficulties in the GAN problem caused by intertwining. One of the difficulties is that the depth of the neural network itself is difficult to train and is unstable. Although the original GAN theory does not require G and D to be neural networks, they only need to be able to fit the corresponding generation and identification function. But this is precisely the depth of the nerve network. So we all use the neural network as G and D . But the choice of neural network, design and training is also full of surprises and uncertainties, which has also brought difficulties for GAN training.

A. Mathematical expression of GAN

The following explains the mathematical principles of GAN. The symbol is described as random noise $z \in R^n$, generated data $\tilde{x} \in R^n$ and natural data $x \in R^n$. The discriminator has two categories, so $y \in [0, 1]^2$.

The following details comprise four aspects.

B. Data

$$\{(x^t, z^t), y^t\}_{t=1}^T \quad (5.34)$$

For the t th data (x^t, z^t) , the output is $y^t \in [0, 1]^t$. The probability that the natural data is true is 1. The probability that the generated data is false is 0.

C. Model

$$\left\{ \begin{array}{l} G : \tilde{x} = g(z, \theta^G) \in R^n \\ D : \left\{ \begin{array}{l} \text{Feature Learning} : \begin{cases} X = D^F(x, \theta^F) \\ \tilde{X} = D^F(\tilde{x}, \theta^F) \end{cases} \\ \text{Classifier Design} : y = \begin{pmatrix} P(L(x) = \text{real} | X, \theta^C) \\ P(L(\tilde{x}) = \text{real} | \tilde{X}, \theta^C) \end{pmatrix} \in R^2 \end{array} \right. \end{array} \right. \quad (5.35)$$

where G is generator. The parameter to be optimized is θ^G ; there is a need to further quantify the non-linear mapping function $g(\cdot)$. D is the discriminator, divided into two parts: one is feature learning (the parameter to be optimized is θ^F); the other is classifier design (the parameter to be optimized is θ^C). There is no need to further quantify the functions $D^F(\cdot)$ and $P(\cdot)$.

D. Optimization function

Usually, the discriminant model section (Eq. (5.41)) is written as:

$$y = \begin{pmatrix} D(x) \\ D(\tilde{x}) \end{pmatrix} = \begin{pmatrix} D(x) \\ D(G(z)) \end{pmatrix} \in R^2 \quad (5.36)$$

where $D(x) \in [0, 1]$ determines the probability that x is a true sample. The loss function of the corresponding discriminant model when the model is fixed is:

$$\begin{cases} \min_{\theta^D} \{ - [\sum_{x \sim P(x)} \log(D(x, \theta^D)) + \sum_{\tilde{x} \sim P(\tilde{x})} \log(D(\tilde{x}, \theta^D))] \} \\ \theta^D = (\theta^F, \theta^C) \end{cases} \quad (5.37)$$

The requirement to generate the model is when the discriminant model is fixed. The distribution of the generated data is as consistent as possible with the natural data. $P(\tilde{x})$ is the same as $P(x)$, maximizing the objective function:

$$\max_{\theta^G} \sum_{\tilde{x} \sim P(\tilde{x})} \log(D(\tilde{x})) = \sum_{\tilde{x} \sim P(x)} \log(D(\tilde{x})) \quad (5.38)$$

Put $x = G(z)$ into the formula:

$$\max_{\theta^G} \sum_{z \sim P(z)} \log(D(G(z), \theta^G)) \rightarrow d(P(\tilde{x}), P(x)) \quad (5.39)$$

under the $z \sim P(z)$ conditions. The greater the sum of $\log(D(G(z)))$ for z means:

$$(D(G(Z)) \sim P(\tilde{x})) \rightarrow d(G(z), x) \rightarrow (D(G(Z)) \sim P(x)) \quad (5.40)$$

The smaller the gap $d(G(Z), x)$ between the generated data and the natural data results in the generated data finally being judged to be 1.

The optimization function of the discriminant model is:

$$\min_{\theta^D} J(\theta^D) = -\frac{1}{T} \left[\begin{array}{l} \sum_{t=1}^T \delta(y^t(1) = real) \cdot \log(D(x^t)) \\ \sum_{t=1}^T \delta(y^t(2) = fake) \cdot \log(1 - D(\tilde{x}^t)) \end{array} \right] \quad (5.41)$$

Usually, because the natural data and the generated data correspond to the authenticity class, respectively, the formula contains:

$$\begin{cases} \delta(y^t(1) = real) = 1 \\ \delta(y^t(2) = real) = 1 \\ \delta(\cdot) \end{cases} \quad (5.42)$$

which is a Dirichlet function.

So the final optimization objective function is:

$$\min_{\theta^D} \min_{\theta^G} J(\theta^D, \theta^G) = \left\{ -\frac{1}{T} \begin{bmatrix} \sum_{t=1}^T \log(D(x^t, \theta^D)) \\ \sum_{t=1}^T \log(1 - D(G(z^t, \theta^G), \theta^D)) \end{bmatrix} \right. \quad (5.43)$$

E. Solution

Use the gradient descent method to optimize the parameter (θ^G, θ^D) alternately.

5.7. Deep recurrent neural networks

Although the traditional deep feedforward neural network has achieved excellent performance in tasks such as classification and target recognition. However, the application of computational models established by its “bionics” is still limited. For example, the deep feed-forward neural network is powerless for the analysis of the overall logical characteristic of the input sequences, which have time correlation between each other (e.g., consider learning a language model to predict the next word based on the previous ones). This section will introduce the deep recurrent neural network (RNN) [163], which is different from the deep neural network previously explored; it can better represent the overall logical characteristic of high dimensions through the introduction of loops within hidden layers.

5.7.1 Simple recurrent neural network

Figure 5.13 is a simple recurrent neural network (i.e., vanilla loop neural network).

As shown in **Figure 5.13(a)**, a chunk of neural network A looks at a series of inputs x_t and outputs a value h_t . The loop structure allows information to be passed from one step of the network to the next. Moreover, the chain-like structure (as shown in **Figure 5.13(b)**) will obviously be obtained while the loop is unfolded. This chain-like structure reveals that the deep recurrent neural network is intimately related to sequences and lists. They are the natural architecture of a neural network used for such data. The overall network model and the solution process are described in the following.

A. Training data

$$\{x_t \in \mathbb{R}^n, y_t \in \mathbb{R}^m\}_{t=1}^T \quad (5.44)$$

where x_t is the input of the moment t , and the length of whole time sequence is T . The output y_t is related to inputs before the moment t (including the moment t), i.e.:

$$\{x_1, x_2, \dots, x_t\} \xrightarrow{\text{relationship}} y_t \quad (5.45)$$

B. Overall model

$$\begin{cases} s_t = \sigma(U \cdot x_t + W \cdot s_{t-1} + b) \\ h_t = V \cdot s_t + c \in \mathbb{R}^m \\ y_t = \text{Softmax}(h_t) \in \mathbb{R}^m \end{cases} \quad (5.46)$$

Note that *Softmax* here is not a classifier, but is used as an activation function to compress an m -dimensional vector into another m -dimensional real number vector, where the value of each element of the vector is between 0 and 1. That is:

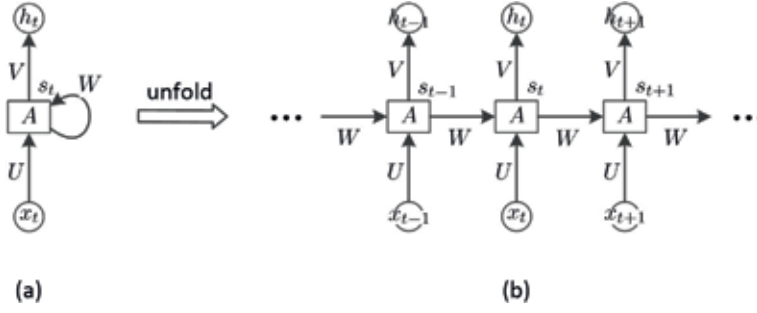


Figure 5.13.
A simple recurrent neural network.

$$\begin{cases} \text{Softmax}(h_t) = \frac{1}{Z} \cdot [e^{h_t(1)}, \dots, e^{h_t(m)}]^T \\ Z = \sum_{j=1}^m e^{h_t(j)} \end{cases} \quad (5.47)$$

where Z represents the normalization factor. In addition, parameters to be optimized in Eq. (5.52), including weighted connection U , W , V and bias b , c . $\sigma(\cdot)$, are the activation functions for the hidden layers.

C. Optimization objective function

Based on the relation within training data (Eq. (5.51)) and the overall model (Eq. (5.52)), the loss function is constructed by negative logarithmic-likelihood (i.e., cross-entropy), and the optimization objective function is as follows:

$$\min_{\theta} J(\theta) = \sum_{t=1}^T \text{loss}(\hat{y}_t, y_t) = \sum_{t=1}^T \left(- \left[\sum_{j=1}^m y_t(j) \cdot \log(\hat{y}_t(j)) + (1 - y_t(j)) \cdot \log(1 - \hat{y}_t(j)) \right] \right) \quad (5.48)$$

where $y_t(j)$ is the j th element of y_t , and the parameter θ is:

$$\theta = [U, V, W; b, c] \quad (5.49)$$

D. Solution

RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being depended on the previous computations. The corresponding loss term is simplified to:

$$J_t(\theta) = \text{loss}(\hat{y}_t, y_t) \quad (5.50)$$

As for the solution of optimization objective function (i.e., Eq. (5.54)), it can be realized through the time back propagation algorithm. The core is the solution to the following five partial derivatives:

$$\left[\frac{\partial J(\theta)}{\partial V}, \frac{\partial J(\theta)}{\partial c}, \frac{\partial J(\theta)}{\partial W}, \frac{\partial J(\theta)}{\partial U}, \frac{\partial J(\theta)}{\partial b} \right] \left(\quad (5.51)$$

The solution of the first two partial derivatives is based on the following error propagation terms:

$$\delta_{h_t} = \frac{\partial J_t(\theta)}{\partial h_t} \quad (5.52)$$

Notice that δ_{h_t} is the partial derivative of the output h_t of objective function at moment t . The solution of the remaining three partial derivatives is based on the error propagation:

$$\delta_{s_t} = \frac{\partial J_t(\theta)}{\partial s_t} \quad (5.53)$$

where δ_{s_t} represents the partial derivatives of the output s_t of the hidden layer activation function at moment t . The following are the partial derivatives of objective function with respect to W ; the remaining partial derivatives are similar:

$$\frac{\partial J(\theta)}{\partial W} = \sum_{t=1}^T \sum_{k=1}^t \frac{\partial s_k}{\partial W} \cdot \frac{\partial s_t}{\partial s_k} \cdot \frac{\partial J_t(\theta)}{\partial s_t} = \sum_{t=1}^T \sum_{k=1}^t \frac{\partial s_k}{\partial W} \cdot \frac{\partial s_t}{\partial s_k} \cdot \delta_{s_t} \quad (5.54)$$

Notice that the output s_t of the hidden layer at moment t is related to the previous outputs $s_k (k = 1, 2, \dots, t)$, as shown in Eq. (5.52) while the parameter W happens to be the intrinsic meaning of this relationship. The equation below can be easily obtained according to the chain rule:

$$\frac{\partial s_t}{\partial s_k} = \prod_{j=k+1}^t \frac{\partial s_j}{\partial s_{j-1}} = \prod_{j=k+1}^t (W^T \cdot \text{diag}(\sigma'(s_{j-1}))) \quad (5.55)$$

where $\sigma'(\cdot)$ is the partial derivative of the activation function $\sigma(\cdot)$. In addition, function $\text{diag}(\cdot)$ is the vector extension matrix, i.e., the resulting matrix takes the vectors as diagonal elements.

In practice, the activation function $\sigma(\cdot)$ of the hidden layer (Eq. (5.52)) often takes the *Tanh*(\cdot) function, so that the variation of the corresponding gradients of Eqs. (5.59) and (5.61) is very small in the later stage of training. Furthermore, the multiplication operation makes the gradient of Eq. (5.60) even smaller, which easily leads to the gradient vanishing phenomenon. The same is true if $\sigma(\cdot)$ takes the *Sigmoid*(\cdot) function. The common tricks used to avoid gradient vanishing include parameter initialization strategies, using the *ReLU*(\cdot) function as the activation function, and so on.

5.7.2 The problem of long-term dependencies

Reconsider the language model trying to predict the next word based on the previous ones. If the recurrent neural network is trying to predict the last word in “the clouds are in the sky,” it doesn’t need any further context; it’s pretty obvious the next word is going to be sky. In such cases, where the gap between the relevant information and the place that it is needed is small, RNNs can learn to use the previous information, and the process is shown in **Figure 5.14**.

But there are also cases where the recurrent neural network needs more context. Consider trying to predict the last word in the text “I grew up in France I speak fluent French.” Recent information suggests that the next word is probably the name of a language, but if the recurrent neural network wants to narrow down which language, it needs the context of France from further back. It’s entirely possible for the gap between the relevant information and the point where it is

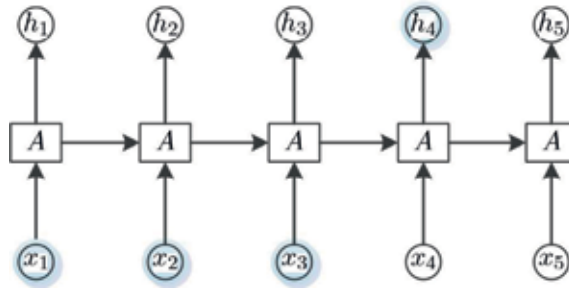


Figure 5.14.
The gap is small so that RNN can successfully use the past information.

needed to become very large. Unfortunately, as that gap grows, RNNs become unable to use the previous information as shown in **Figure 5.15**.

Despite RNNs being theoretically capable of establishing the dependencies between long time interval states, as a result of gradient vanishing or gradient explosion (i.e., the gradient tends to infinity after the multiplication operation, which makes the system unstable), they can only learn the short-term dependencies in a practical application, which limits the memory capacity of a recurrent neural network. This is the so-called long-term dependencies problem.

5.7.3 Long short-term memory networks

Long short-term memory networks [164], usually just called LSTMs, are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter and Schmidhuber [164], and were refined and popularized by many people [1]. They work tremendously well on a large variety of problems, and are now widely used. The core problem of the known simple recurrent neural network is that the gradient explosion or gradient vanishing phenomenon occurs easily as the time interval increases. To solve this problem effectively, LSTMs control the accumulation speed of information by introducing the gate mechanism, and they can choose to forget the accumulation of previous information.

A. Analysis of improvement motivation

In the simple recurrent neural network, we can see from the Eqs. (5.60) and (5.61) that if we define:

$$\zeta = W^T \cdot \text{diag}(\sigma'(s_{j-1})) \quad (5.56)$$

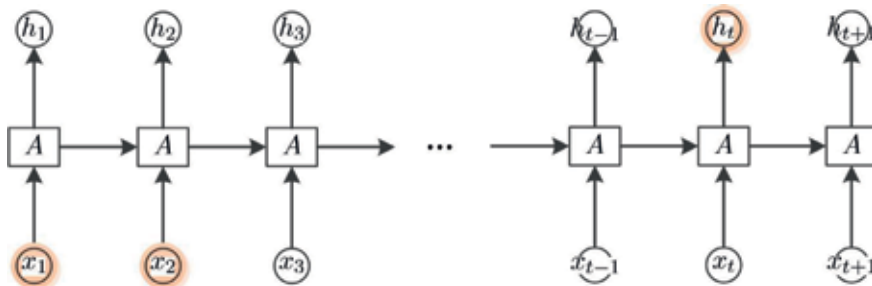


Figure 5.15.
A large gap makes RNN unable to use the past information.

there are:

$$\prod_{j=k+1}^t (W^T \cdot \text{diag}(\sigma'(s_{j-1}))) \rightarrow \zeta^{t-k} \quad (5.57)$$

As for ζ , if it's a spectral radius $\|\zeta\| > 1$, Eq. (5.64) will diverge and cause the system to appear as the so-called gradient explosion problem. On the contrary, the gradient vanishing problem will occur as the time difference increases infinitely if $\|\zeta\| < 1$.

To avoid the gradient explosion or gradient vanishing problem, the key is the spectral radius of ζ set as $\|\zeta\| = 1$. Generally, if W is set as a unit matrix and the spectral norm of $\sigma'(s_{j-1})$ is set to 1 at the same time, the relationship of hidden layers within the model degenerates to:

$$s_t = \sigma(U \cdot x_t + W \cdot s_{t-1} + b) \xrightarrow{\text{degenerate}} s_t = U \cdot x_t + s_{t-1} + b \quad (5.58)$$

But in this form the properties of non-linear activation are lost. Therefore, the improved approach is the introduction of cell state, recorded as c_t , to carry out the non-linear transmission of information, namely:

$$\begin{cases} c_t = c_{t-1} + U \cdot x_t \\ s_t = \text{Tanh}(c_t) \end{cases} \quad (5.59)$$

Note that the non-linear activation function here is $\text{Tanh}(\cdot)$.

B. The core idea behind LSTMs

Based on **Figure 5.16**, the key to LSTMs is the cell state to control the information changes. The cell state is like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It's very easy for information to flow along it unchanged. The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates. Note that in **Figure 5.16**, the input at the moment t consists of three gates, i.e., x_t , c_{t-1} and s_{t-1} ; the output consists of two gates, i.e., c_t and s_t .

Note that the core design of the network consists of three doors, i.e., forget gate, input gate and output gate. The overall network structure is shown in **Figure 5.17**.

C. Forget gate

The first step in LSTM is to decide what information its going to be thrown away from the cell state. This decision is made by a sigmoid layer called the forget gate layer. It looks at x_t and s_{t-1} , and outputs a number between 0 and 1 for each element

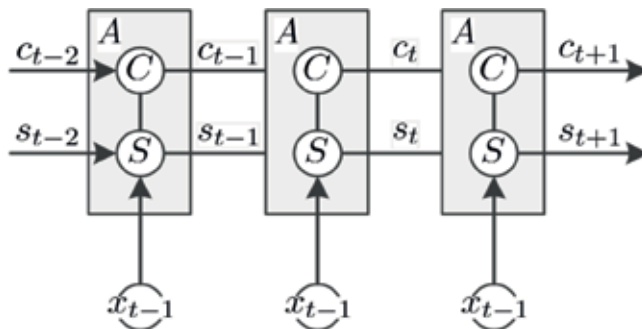


Figure 5.16.
 The improved structure.

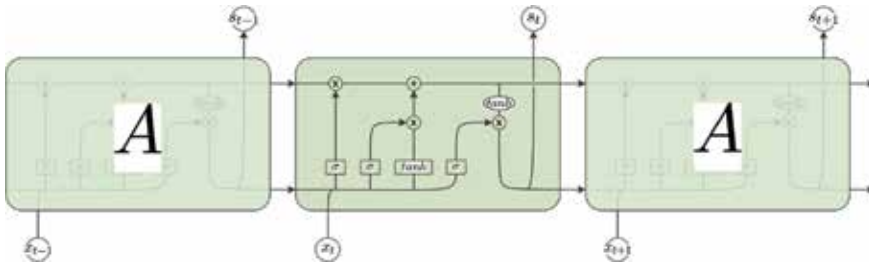


Figure 5.17.
An overview of LSTMs.

in the cell state c_{t-1} . A 1 represents completely keep this, while a 0 represents completely get rid of this. A detailed equation of the forget gate is as follows:

$$f_t = \sigma(W_f \cdot [s_{t-1}, x_t] + b_f) \quad (5.60)$$

where σ is the *Sigmoid* function, $[\cdot]$ represents the aggregation operation, the weights of s_{t-1} and x_t are merged in W_f and b_f is the bias of the forget gate. The specific structure of the forget gate is shown in **Figure 5.18**.

D. Input gate

The next step is to decide what new information LSTM is going to store in the cell state. This has two parts. First, a sigmoid layer called the input gate layer decides which values LSTM will update; then, a tanh layer creates a vector of new candidate values, \tilde{c}_t , that could be added to the state. The equation for this part is as follows:

$$\begin{cases} i_t = \sigma(W_i \cdot [s_{t-1}, x_t] + b_i) \\ \tilde{c}_t = \text{Tanh}(W_c \cdot [s_{t-1}, x_t] + b_c) \end{cases} \quad (5.61)$$

where the element values within i_t are between 0 and 1, indicating the proportion of each element being updated in the state. The value of \tilde{c}_t depends on s_{t-1}

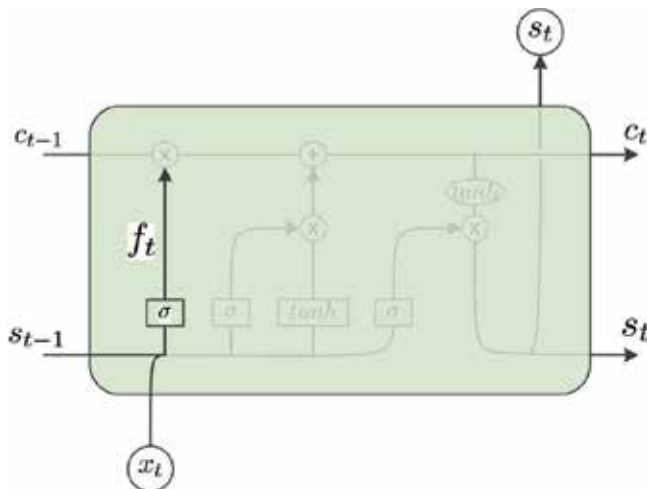


Figure 5.18.
The forget gate of LSTM.

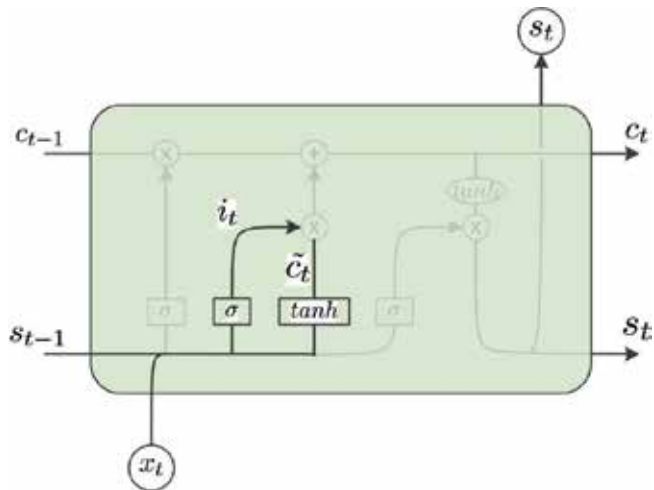


Figure 5.19.
 The input gate of LSTM.

and x_t . The rest of the parameters are similar to those of the forget gate. A schematic of this part is shown in **Figure 5.19**.

Next, LSTM will combine these two to create an update to the state. The detailed equation is as follows:

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t \quad (5.62)$$

The previous term is the old state multiplied by f_t , indicating that LSTM forgets elements that it has already decided. Then it adds $i_t * \tilde{c}_t$; this is the new candidate value, scaled by how much LSTM decided to update each state value. A schematic representation of the generation of the new state is shown in **Figure 5.20**.

E. Output gate

Finally, LSTM needs to decide what it's going to output. This output will be based on the cell state, but will be a filtered version. First, there is a sigmoid layer called the output gate layer, which decides what parts of the cell layer it's going to output. Then, LSTM will put the cell state through the Tanh layer (to push the

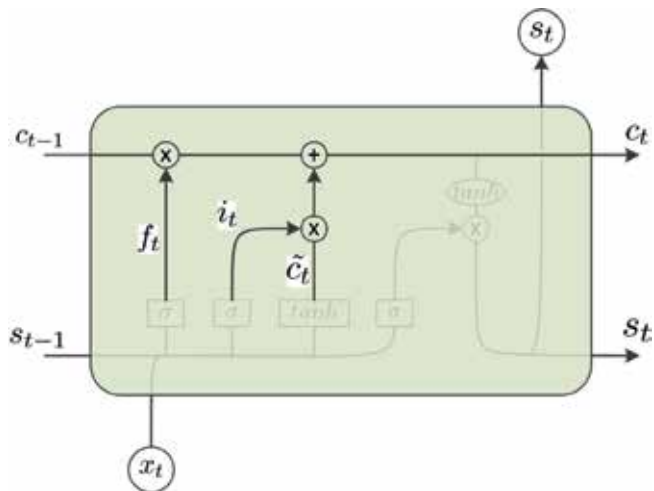


Figure 5.20.
 LSTM combines the two parts to create the update to the state.

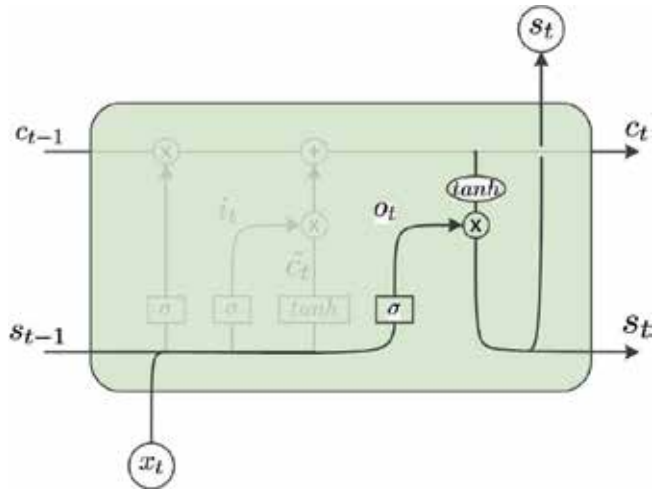


Figure 5.21.
The output gate of LSTM.

values between 0 and 1) and multiply it by the output of the output gate, so that LSTM only outputs the parts it has decided to. The equation for this step is as follows:

$$\begin{cases} o_t = \sigma(W_o \cdot [s_{t-1}, x_t] + b_o) \\ s_t = o_t * \text{Tanh}(c_t) \end{cases} \quad (5.63)$$

where o_t represents the output of the output gate, and s_t is the final output state after filtration. The specific diagram is shown in **Figure 5.21**.

5.8. Deep reinforce learning

5.8.1 Basic idea of deep reinforce learning

Deep reinforcement learning [165] combines the perception of deep learning with the decision-making ability of reinforcement learning, and can directly control the input image. It is an artificial intelligence method that is closer to human thinking. The criteria for measuring artificial intelligence are perception, awareness, and decision making. It can be considered that deep learning (deep neural network) [16] is to further enhance and break through the core technology of perceptual ability. At the same time, the learning mechanism of intensive learning shows that it is constantly interacting with the environment and constantly trying to get the best strategy. It is the key technology to make the decision-making ability to constantly obtain the benefit. The principle of deep reinforce learning is shown in **Figure 5.22**.

In addition, it is known that the information processing capacity of the brain is limited and it is impossible to perform a variety of operations in an instant. To process a large amount of information smoothly, people can only select specific information to process at each time according to a certain strategy, and organize a large number of operations of the whole process of cognition (including feeling,

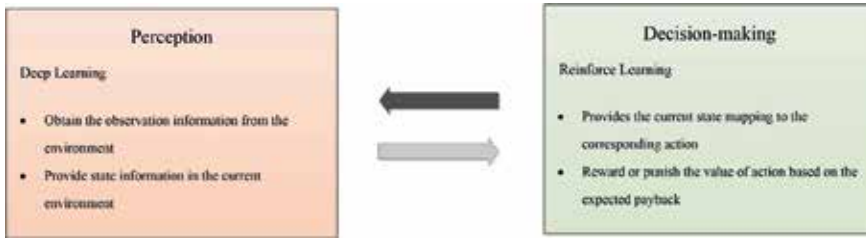


Figure 5.22.
 The deep reinforce learning framework.

perception, memory, thinking, imagination, speech, etc.). Therefore, perception, cognition and decision-making are important for the effective conduct and development of cognitive activities.

5.8.2 Deep Q network

Q learning was presented by Watkins [166], which is the earliest and one of the most important algorithms for online reinforcement learning. The deep Q network was the first deep reinforcement learning algorithm proposed by Google's DeepMind in 2015 [167] and was further refined in 2015; the results are published in the journal *Nature*. DeepMind applies the Deep Q network to a video game called Atari, which have not been used in the past. This method only uses video information as input, such as a video of a person playing the game, and the symbol "Q" indicates the score or quality obtained when the operation is performed in a certain stage.

5.8.2.1 Basic network model and framework

In the deep Q network, the evaluation module is represented only by the value network, and no action module is used because the evaluation module can select and perform the optimal action. The core idea is that based on the value network, a model can traverse the value of a variety of actions under a certain state, and then select the action that has the largest value as output, as shown in **Figure 5.23**.

Note that the deep Q network, as the first algorithm for deep reinforce learning, only uses the value network, which is characterized by low training efficiency and

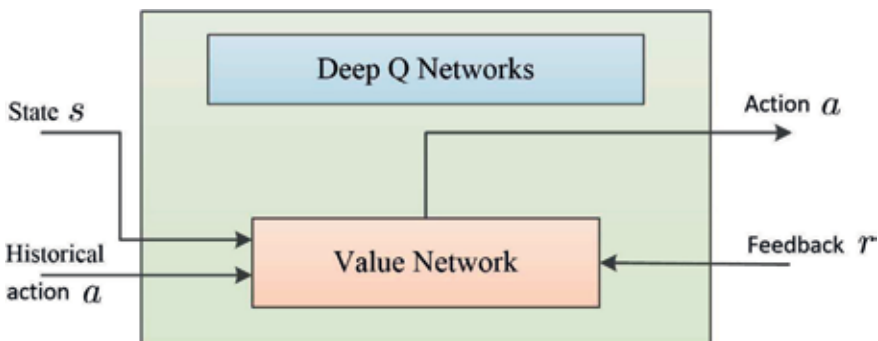


Figure 5.23.
 Basic framework of the deep Q network.

can only deal with low-dimension discrete control problems; in other words, its generality is limited. But since it's the first successful combination of deep learning and enhanced learning to solve the high-dimensional data input problem, and in the Atari game has made a major breakthrough, it is groundbreaking. As the deep convolution neural network has a natural advantage in the field of image processing, it is one of the current mainstreams to deal with image data in combination with Q learning in reinforce learning. In the DQN training process, the adjacent four-frame game screen is used as the input of the network, and the Q values of the optional actions in the current state are output through multiple convolution layers and allconnected layers, thereby realizing end-to-end learning control.

A. Mathematical analysis of the deep Q network

1. Analysis of Q learning

For example, a chess game can be divided into a series of steps (the last step is the result), each step is the action generated by the player based on their current observations. At the same time, it is expected that the cumulative reward (the sum of the behavioral reward and the previous behavior) brought by this action is better than the other. How do you quantify this process using mathematical analysis? First, the necessary assumption is that the observations or actions performed in each step of the game are limited. In addition to this, it is impossible to perform new actions only by observing the current steps, and it is necessary to combine the previous observations and actions.

For convenience, several symbols are given:

- The symbol $x_t \in \mathbb{R}^{m \times n}$ represents the corresponding observation in the $t(t = 1, 2, \dots, T)$ step of the game.
- The symbol $a_t \in \Lambda$ is the action performed under observation x_t , where Λ is the set of reasonable actions under the rules of the game.
- The symbol r_t is the reward (or penalty) acquired after the action a_t is taken under observation x_t , and:

$$R_t = \sum_{t'=t}^T \gamma^{t'-t} \cdot r_{t'} \quad (5.64)$$

where $\gamma \in (0, 1)$ is the discount factor, and R_t is the sum of cumulative reward obtained from step t to the end time T .

- The symbol $Q(s, a)$ is the state-action value function, where the state s at time t is:

$$s_t = (x_1, a_1, \dots, x_{t-1}, a_{t-1}, x_t) \quad (5.65)$$

Next, the main idea of Q learning is given, that is, according to the following iterative equation, to implement the optimization learning of the state-action value function:

$$\begin{cases} Q_{k+1}(s_t, a_t) = Q_k(s_t, a_t) + a_k \cdot \delta_k \\ \delta_k = r_{t+1} + \gamma \cdot \max_{a' \in \Lambda} Q_k(s_{t+1}, a') - Q_k(s_t, a_t) \end{cases} \quad (5.66)$$

where a_k is the learning rate, the other s_t and a_t are the state and action corresponding to the t th step, respectively, δ_k is the time difference, γ is the discount factor and a' are executable actions of the state-action value function under the k th iteration with respect to the state s_{t+1} within Λ . It has been shown that when Eq. (5.72) satisfies the following two conditions, one is:

$$\sum_k a_k^2 < +\infty, \quad \sum_k a_k = +\infty \quad (5.67)$$

and the other is that all states and actions can be traversed indefinitely, and there are:

$$\lim_{k \rightarrow \infty} Q_k = Q^* \quad (5.68)$$

That is, the state-action value function under the optimal control strategy is obtained when the number of iterations tends to infinity. Similar to the watermelon task, a set of optimal execution strategies from seeding to result are summarized.

Finally, based on the above description of the Bellman equation, it can be seen that the strategy of selecting the optimal executable action under a certain state is to maximize the following expectations:

$$Q^*(s, a) = \mathbb{E}_{s' \sim \xi}(r + \gamma \cdot \max_{a'} Q^*(s', a') | s, a) \quad (5.69)$$

where ξ is the environment (all states are included), s' is the state after the action a is executed for state s and a' are all possible actions for state s' under the game rules.

Note that Eq. (5.75) uses the expectation to analyze the state-action value function. In practice, the function approximation strategy is often used to estimate the state-action value function, i.e.:

$$Q(s, a, \theta) \approx Q^*(s, a) \quad (5.70)$$

In reinforce learning, the common approximation methods are linear and non-linear (e.g., the neural networks); note that the weight parameters to be optimized are θ . Assume that for each iteration k , the Q network implements the parameter update by minimizing the following objective functions:

$$L_k(\theta_k) = \mathbb{E}_{s, a \sim \rho(\cdot)} \left[(y_k - Q(s, a; \theta_k))^2 \right] \quad (5.71)$$

where $\rho(\cdot)$ is the behavior distribution, i.e., $\rho(s, a)$ is the probability distribution of state s and behavior a , and y_k is the objective (output) corresponding to the k th iteration, and there are:

$$y_k = \mathbb{E}_{s' \sim \xi} \left[r + \gamma \cdot \max_{a'} Q(s', a'; \theta_{k-1} | s, a) \right] \quad (5.72)$$

According to Eqs. (5.77) and (5.78), there are:

$$\theta_0 \xrightarrow{\min_{\theta} L_1(\theta_1)} \theta_1 \xrightarrow{\min_{\theta} L_2(\theta_2)} \dots \xrightarrow{\min_{\theta} L_k(\theta_k)} \theta_k \rightarrow \dots \quad (5.73)$$

That is, according to Eq. (5.78), the objective output y_1 can be obtained under the condition that parameter θ_0 is known, and the parameters are updated by

optimizing the objective function in Eq. (5.77) to get θ_1 . The rest can be done in the same manner, and the convergence of the parameters is finally realized, namely:

$$\lim_{k \rightarrow \infty} \theta_k = \theta_* \quad (5.74)$$

Among them, the gradient descent algorithm is used to update the parameters of Eq. (5.77), where the partial derivative is:

$$\nabla_{\theta_k} L_k(\theta_k) = \mathbb{E}_{s, a \sim \rho(\cdot); s' \sim \xi} \left[(r + \gamma \cdot \max_{a'} Q(s', a'; \theta_{k-1}) - Q(s, a; \theta_k)) \cdot \nabla_{\theta_k} Q(s, a; \theta_k) \right] \quad (5.75)$$

Note that the parameter θ_{k-1} is fixed here.

1. Analysis of deep Q learning

At present, the deep learning techniques that rely on a large number of training data sets to succeed have achieved many breakthroughs such as computer vision and voice processing. The most intuitive conclusion is: feature engineering (including feature extraction and feature selection) that relies on a priori knowledge or physical characteristics of data will be replaced by feature learning based on depth learning techniques.

In general, the success of Q learning depends on the selection of artificial features. Furthermore, the quality of the agent's learning is critically dependent on the quality of the feature selection. Can the artificial feature selection techniques in Q learning be replaced with feature learning under deep learning, such as feature learning based on convolution neural networks? The answer is yes; this is the motivation for deep Q learning. For the first time we introduce a concept, Experience Replay, which stores the experience of an agent in any step, such as:

$$e_t = (s_t, a_t, r_t, s_{t+1}) \quad (5.76)$$

The symbols appear the same as in Q learning, where e_t is the experience of the t th step. Assuming that the step at the end of this agent is N , then the experience replay can correspond to the set:

$$D = [e_1, e_2, \dots, e_N] \quad (5.77)$$

Second, the state-action value function in Q learning is revised as:

$$Q(s, a, \theta) \xrightarrow{\text{correction}} Q(\phi(s), a, \theta) \quad (5.78)$$

Here, $\phi(\cdot)$ is the feature learning based on deep learning. Pay attention to two points: the parameters θ here not only contain the parameters of Q learning, but also the parameters of deep learning and for the t th step in Q learning, there are:

$$s_{t+1} = (s_t, a_t, x_{t+1}) \quad (5.79)$$

where $s_1 = [x_1]$ and $t = 1, 2, \dots, T$. For Eq. (5.84), the left corresponds to Q learning, where the state input is s ; the right corresponds to deep Q learning, where the "state" input is $\phi(s)$, that is, the corresponding correction of the current experience replay is:

$$\begin{cases} D \xrightarrow{\text{correction}} \bar{D} = [\bar{e}_1, \bar{e}_2, \dots, \bar{e}_N] \\ \bar{e}_t = (\phi(s_t), a_t, r_t, \phi(s_{t+1})) \end{cases} \quad (5.80)$$

Moreover, in the corresponding Eq. (5.77) of the optimization objective function and the corresponding Eq. (5.78) of the expectation output y_k , the input state s of the state-action value function should be corrected to $\phi(s)$. Finally, the solution still uses the gradient decent algorithm

Section 3

Privacy Preserving for Data
Services in Mobile Internet

Attribute-Based Encryption for Flexible and Fine-Grained Access Control

6.1. Introduction

6.1.1 The motivation of ABE

With the development of the Internet and the distributed computing technology, there is a growing demand for data sharing and processing in an open distributed computing environment. The data provider needs to provide expressive access control and data confidentiality when communicating with customers. What is more is that it is urgent for large-scale distributed applications to support one-to-many communication mode to reduce the enormous costs of data encryption (Figure 6.1).

The traditional encryption mechanism based on public key infrastructure (PKI) [75] can achieve data confidentiality; however, it has disadvantages. On the one hand, in order to encrypt data, the data provider needs firstly to obtain the public keys of authorized users and then sends the encrypted data separately to the corresponding user, which increases the processing overhead and the bandwidth demand [168]. On the other hand, although broadcast encryption [169] can solve the efficiency problem mentioned above, the data provider must obtain the user's list before encryption. In addition, if the data provider wants the recipient to be the one with certain identity not the one who is specified, the public key encryption will not work anymore. Therefore, more applicable encryption mechanisms are required. That's why the ABE scheme was created.

In ABE scheme, attribute plays a very important role. Attributes have been exploited to generate a public key for encryption data and have been used as an access policy to control users' access.

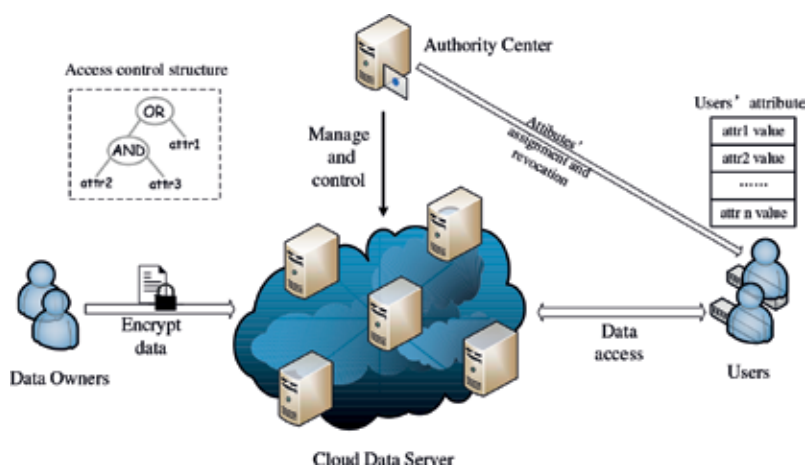


Figure 6.1. System model.

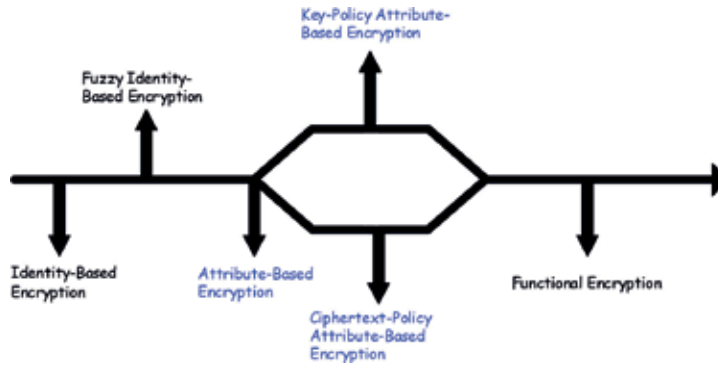


Figure 6.2.
Evolution of ABE.

6.1.2 History of ABE

After the introduction of fuzzy identity-based encryption, the notion of ABE was initially invented by Nali et al. [170] who proposed a threshold ABE scheme. Although this scheme can prevent the collusion attacks, it introduces new disadvantage that the threshold semantics are limited in designing more general systems which need expressive access control (**Figure 6.2**).

In ABE scheme, attribute plays a very important role. Attributes have been exploited to generate a public key for encryption data and have been used as an access policy to control users' access. Based on the access policy, subsequent researches can be roughly categorized as either key policy or ciphertext policy. The first KP-ABE scheme that allows any monotone access structures was proposed by Goyal et al. [171], and the first CP-ABE scheme was presented by Bethencourt et al. [172]. After that, several KP-ABE [173–175] and CP-ABE schemes [176–184] were proposed. Goyal et al. [176] presented a bounded CP-ABE scheme in the standard model, but the first fully expressive CP-ABE scheme in the standard model was proposed by Waters [177]. Subsequently, there are some other versions of ABE that have been proposed such as the outsourced ABE, the ciphertext-policy attribute-based proxy re-encryption (CP-ABPRE), and so on.

Organization. The remainder of this chapter is organized as follows. Section 2 presents the formal definitions and security models of attribute-based encryption (ABE). Design philosophy is given in Section 3, together with concrete construction and security proof of several representative attribute-based encryption schemes. Comparison and observations are discussed in Section 4.

6.2. Framework and security model of ABE

In this chapter, we present the formal definitions and security models of attribute-based encryption (ABE) via considering two distinct varieties of attribute-based encryption: ciphertext policy (CP-ABE) and key policy (KP-ABE).

6.2.1 The formal definition of ABE

Definition 1. (Access structure). Let $\{P_1, P_2, \dots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1, P_2, \dots, P_n\}}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}$. An access structure (respectively, monotone access structure) is a collection (respectively, monotone

collection) \mathbb{A} of non-empty subsets of $\{P_1, P_2, \dots, P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1, P_2, \dots, P_n\}} \setminus \{\emptyset\}$. The sets in \mathbb{A} are called the authorized sets, and the sets not in \mathbb{A} are called the unauthorized sets.

In our context, the role of the parties is taken by the attributes. Thus, the access structure \mathbb{A} will contain the authorized sets of attributes. We restrict our attention to monotone access structures. However, it is also possible to (inefficiently) realize general access structures using our techniques by having the negative attribute as the separate attribute altogether. Thus, the number of attributes in the system will be doubled. From now on, unless stated otherwise, by an access structure, we mean a monotone access structure.

In CP-ABE, an access structure (policy) is embedded into the ciphertext during encryption, and each private key is based on an attribute set S . On the contrary, KP-ABE inverts this relationship, embedding S into the ciphertext and a policy into the key. But in both two varieties of ABE, each interior node of the tree is a threshold gate, and the leaves are associated with attributes. (We note that this setting is very expressive. For example, we can represent a tree with “and” and “or” gates by using, respectively, two of two and one of two threshold gates.) In CP-ABE, a user will be able to decrypt a ciphertext with a given key if and only if there is an assignment of attributes from the private key (it is the ciphertext in KP-ABE) to nodes of the tree such that the tree is satisfied.

Definition 2. (Access tree). Let \mathcal{T} be a tree representing an access structure. Each non-leaf node of the tree represents a threshold gate, described by its children and a threshold value. If num_x is the number of children of a node x and k_x is its threshold value, then $0 < k_x < num_x$. When $k_x = 1$, the threshold gate is an “or” gate, and when $k_x = num_x$, it is an AND gate. Each leaf node x of the tree is described by an attribute and a threshold value $k_x = 1$.

To facilitate working with the access trees, we define a few functions. We denote the parent of the node x in the tree by $parent(x)$. The function $att(x)$ is defined only if x is a leaf node and denotes the attribute associated with the leaf node x in the tree. The access tree \mathcal{T} also defines an ordering between the children of every node; that is, the children of a node are numbered from 1 to num_x . The function $index(x)$ returns such a number associated with the node x , where the index values are uniquely assigned to nodes in the access structure for a given key in an arbitrary manner.

Satisfying an access tree. Let \mathcal{T} be an access tree with root r . Denote by \mathcal{T}_x the subtree of \mathcal{T} rooted at the node x . Hence, \mathcal{T} is the same as \mathcal{T}_r . If a set of attributes γ satisfies the access tree \mathcal{T}_x , we denote it as $\mathcal{T}_x(\gamma) = 1$. We compute $\mathcal{T}_x(\gamma)$ recursively as follows. If x is a non-leaf node, evaluate $\mathcal{T}_{x'}(\gamma)$ for all children x' of node x . $\mathcal{T}_x(\gamma)$ returns 1 if and only if at least k_x children return 1. If x is a leaf node, then $\mathcal{T}_x(\gamma)$ returns 1 if and only if $att(x) \in \gamma$.

1. Ciphertext-policy attribute-based encryption (CP-ABE)

A ciphertext-policy attribute-based encryption scheme consists of four fundamental algorithms: Setup, Encrypt, KeyGen, and Decrypt.

- **Setup:** The setup algorithm takes no input other than the implicit security parameter. It outputs the public parameters PK and a master key MK .
- **Encrypt(PK, M, \mathbb{A}):** The encryption algorithm takes as input the public parameters PK , a message M , and an access structure \mathbb{A} over the universe of attributes. The algorithm will encrypt M and produce a ciphertext CT

such that only a user that possesses a set of attributes that satisfies the access structure will be able to decrypt the message. We will assume that the ciphertext implicitly contains \mathbb{A} .

- **KeyGen**(MK, S): The key generation algorithm takes as input the master key MK and a set of attributes S that describe the key. It outputs a private key SK .
- **Decrypt**(PK, CT, SK): The decryption algorithm takes as input the public parameters PK ; a ciphertext CT , which contains an access policy \mathbb{A} ; and a private key SK , which is a private key for a set S of attributes. If the set S of attributes satisfies the access structure \mathbb{A} , then the algorithm will decrypt the ciphertext and return a message M .

2. Key-policy attribute-based encryption (KP-ABE)

Similar to CP-ABE, a key-policy attribute-based encryption scheme consists of four fundamental algorithms: Setup, Encrypt, KeyGen, and Decrypt.

- **Setup**: The setup algorithm takes no input other than the implicit security parameter. It outputs the public parameters PK and a master key MK .
- **Encrypt**(PK, M, S): The encryption algorithm takes as input the public parameters PK , a message M , and a set of attributes S . It encrypts M and outputs the ciphertext CT .
- **KeyGen**(MK, \mathbb{A}, PK): The key generation algorithm takes as input the master key MK , an access structure \mathbb{A} , and the public parameters PK . It outputs a private key SK .
- **Decrypt**(CT, SK, PK): The decryption takes as input the ciphertext CT that was encrypted under the set S of attributes, the private key SK for access control structure \mathbb{A} , and the public parameters PK . It decrypts the ciphertext and outputs the message M if the set S of attributes satisfies the access structure \mathbb{A} .

6.2.2 Security model of ABE

1. Ciphertext-policy attribute-based encryption (CP-ABE)

- **Setup**: The challenger runs the setup algorithm and gives the public parameters PK to the adversary.
- **Phase 1**: The adversary makes repeated private keys corresponding to sets of attributes S_1, \dots, S_{q_1} .
- **Challenge**: The adversary submits two equal-length messages M_0 and M_1 . In addition the adversary gives a challenge access structure \mathbb{A}^* such that none of the sets S_1, \dots, S_{q_1} from Phase 1 satisfy the access structure. The challenger flips a random coin b and encrypts M_b under \mathbb{A}^* . The ciphertext CT^* is given to the adversary.

- **Phase 2:** Phase 1 is repeated with the restriction that none of sets of attributes S_{q_1+1}, \dots, S_q satisfy the access structure corresponding to the challenge.
- **Guess:** The adversary outputs a guess b' of b .

The advantage of an adversary \mathcal{A} in this game is defined as $\Pr[b' = b] - \frac{1}{2}$.

We note that the model can easily be extended to handle chosen-ciphertext attacks by allowing for decryption queries in Phase 1 and Phase 2.

Definition 3. *A ciphertext-policy attribute-based encryption scheme is secure if all polynomial-time adversaries have at most a negligible advantage in the above game.*

2. Key-policy attribute-based encryption (KP-ABE)

- **Init:** The adversary declares the set of attributes S , that he wishes to be challenged upon.
- **Setup:** The challenger runs the setup algorithm and gives the public parameters PK to the adversary.
- **Phase 1:** The adversary is allowed to issue queries for private keys for many access structures \mathbb{A}_j , where $S \notin \mathbb{A}_j$ for all j .
- **Challenge:** The adversary submits two equal-length messages M_0 and M_1 . The challenger flips a random coin b and encrypts M_b with S . The ciphertext is passed to the adversary.
- **Phase 2:** Phase 1 is repeated.
- **Guess:** The adversary outputs a guess b' of b .

The advantage of an adversary \mathcal{A} in this game is defined as $\Pr[b' = b] - \frac{1}{2}$.

We note that the model can easily be extended to handle chosen-ciphertext attacks by allowing for decryption queries in Phase 1 and Phase 2.

Definition 4. *An attribute-based encryption scheme is secure in the selective-set model of security if all polynomial-time adversaries have at most a negligible advantage in the selective-set game.*

6.3. State of the art

6.3.1 The first ABE

In 2006, Goyal et al. [171] proposed a new cryptosystem for fine-grained sharing of encrypted data called key-policy attribute-based encryption (KP-ABE). In the scheme, ciphertexts are labeled with sets of attributes, and private keys are associated with access structures that control which ciphertexts a user is able to decrypt. In addition, their construction supported delegation of private keys that subsume hierarchical identity-based encryption (HIBE).

6.3.1.1 Concrete construction

Let \mathbb{G}_1 be a bilinear group of prime order p , and let g be a generator of \mathbb{G}_1 . In addition, let $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ denote the bilinear map. A security parameter, k , will

determine the size of the groups. We also define the Lagrange coefficient $\Delta_{i,S}$ for $i \in \mathbb{Z}_p$ and a set, S , of elements in \mathbb{Z}_p : $\Delta_{i,S(x)} = \prod_{j \in S, j \neq i} \frac{x-j}{i-j}$. We will associate each attribute with a unique element in \mathbb{Z}_p^* . Our construction follows:

- **Setup:** Define the universe of attributes $\mathcal{U} = \{1, 2, \dots, n\}$. Now, for each attribute $i \in \mathcal{U}$, choose a number t_i uniformly at random from \mathbb{Z}_p . Finally, choose y uniformly at random in \mathbb{Z}_p . The published public parameters PK are

$$T_1 = g^{t_1}, \dots, T_{|\mathcal{U}|} = g^{t_{|\mathcal{U}|}}, Y = e(g, g)^y. \quad (6.1)$$

The master key MK is

$$t_1, \dots, t_{|\mathcal{U}|}, y. \quad (6.2)$$

- **KeyGen**(\mathcal{T}, MK): The algorithm outputs a key that enables the user to decrypt a message encrypted under a set of attributes γ if and only if $T(\gamma) = 1$. The algorithm proceeds as follows. First, choose a polynomial q_x for each node x (including the leaves) in the tree \mathcal{T} . These polynomials are chosen in the following way in a top-down manner, starting from the root node r .
- **Encrypt**(M, γ, PK): To encrypt a message $M \in \mathbb{G}_2$ under a set of attributes γ , choose a random value $s \in \mathbb{Z}_p$ and publish the ciphertext as

$$E = (\gamma, E' = MY^s, \{E_i = T_i^s\}_{i \in \gamma}). \quad (6.3)$$

For each node x in the tree, set the degree d_x of the polynomial q_x to be 1 less than the threshold value k_x of that node, that is, $d_x = k_x - 1$. Now, for the root node r , set $q_r(0) = y$ and d_r other points of the polynomial q_r randomly to define it completely. For any other node x , set $q_x(0) = q_{parent(x)}(\text{index}(x))$ and choose d_x other points randomly to completely define q_x .

Once the polynomials have been decided, for each leaf node x , we give the following secret value to the user:

$$D_x = g^{\frac{q_x(0)}{t_i}} \text{ where } i = \text{att}(x). \quad (6.4)$$

The set of above secret values is the decryption key D .

- **Decrypt**(E, D): We specify our decryption procedure as a recursive algorithm. For ease of exposition, we present the simplest form of the decryption algorithm and discuss potential performance improvements in the next subsection.

We first define a recursive algorithm $DecryptNode(E, D, x)$ that takes as input the ciphertext $E = (\gamma, E', \{E_i\}_{i \in \gamma})$, the private key D (we assume that the access tree \mathcal{T} is embedded in the private key), and a node x in the tree. It outputs a group element of \mathbb{G}_2 or \perp .

Let $i = \text{att}(x)$. If the node x is a leaf node, then

$$\text{DecryptNode}(E, D, x) = \begin{cases} e(D_x, E_i) = e(g^{\frac{q_x(0)}{t_i}}, g^{s \cdot t_i}) \\ = e(g, g)^{s \cdot q_x(0)} & \text{if } i \in \gamma \\ \perp & \text{otherwise.} \end{cases} \quad (6.5)$$

We now consider the recursive case when x is a non-leaf node. The algorithm $\text{DecryptNode}(E, D, x)$ then proceeds as follows: For all nodes z that are children of x , it calls $\text{DecryptNode}(E, D, z)$ and stores the output as F_z . Let S_x be an arbitrary k_x -sized set of child nodes z such that $F_z \neq \perp$. If no such set exists, then the node was not satisfied and the function returns \perp .

Otherwise, we compute

$$\begin{aligned} F_x &= \prod_{z \in S_x} F_z^{A_i, s'_x(0)}, \text{ where } i = \text{index}(z) \\ &\quad S'_x = \{\text{index}(z) : z \in S_x\} \\ &= \prod_{z \in S_x} (e(g, g)^{s \cdot q_z(0)})^{A_i, s'_x(0)} \\ &= \prod_{z \in S_x} (e(g, g)^{s \cdot q_{\text{parent}(z)}(\text{index}(z))})^{A_i, s'_x(0)} \text{ (by constr.)} \\ &= \prod_{z \in S_x} e(g, g)^{s \cdot q_x(i)} \cdot \Delta_{i, s'_x(0)} \\ &= e(g, g)^{s \cdot q_x(0)} \text{ (using polynomial interpolation)} \end{aligned} \quad (6.6)$$

and return the result.

Now that we have defined our function DecryptNode , the decryption algorithm simply calls the function on the root of the tree. We observe that $\text{DecryptNode}(E, D, x) = e(g, g)^{Y^s} = Y^s$ if and only if the ciphertext satisfies the tree. Since $E' = MY^s$, the decryption algorithm simply divides out Y^s and recovers the message M .

We discuss how to optimize the decryption procedure in the full version of this paper [171].

6.3.1.2 Security proof

We prove that the security of our scheme in the attribute-based selective-set model reduces to the hardness of the decisional BDH assumption. The intuition of security proof in the scheme is illustrated in **Figure 6.3**.

Theorem 1. *If an adversary can break our scheme in the attribute-based selective-set model, then a simulator can be constructed to play the decisional BDH game with a non-negligible advantage.*

Proof: Suppose that there exists a polynomial-time adversary \mathcal{A} that can attack our scheme in the selective-set model with advantage ϵ . We build a simulator \mathcal{B} that can play the decisional BDH game with advantage $\epsilon/2$. The simulation proceeds as follows:

We first let the challenger set the groups \mathbb{G}_1 and \mathbb{G}_2 with an efficient bilinear map e and generator g . The challenger flips a fair binary coin μ , outside of \mathcal{B}' 's view. If $\mu = 0$, the challenger sets $(A, B, C, Z) = (g^a, g^b, g^c, e(g, g)^{abc})$; otherwise, it sets $(A, B, C, Z) = (g^a, g^b, g^c, e(g, g)^z)$ for random a, b, c, z . We assume that the universe, \mathcal{U} , is defined. The whole game is described in **Figure 6.4**.

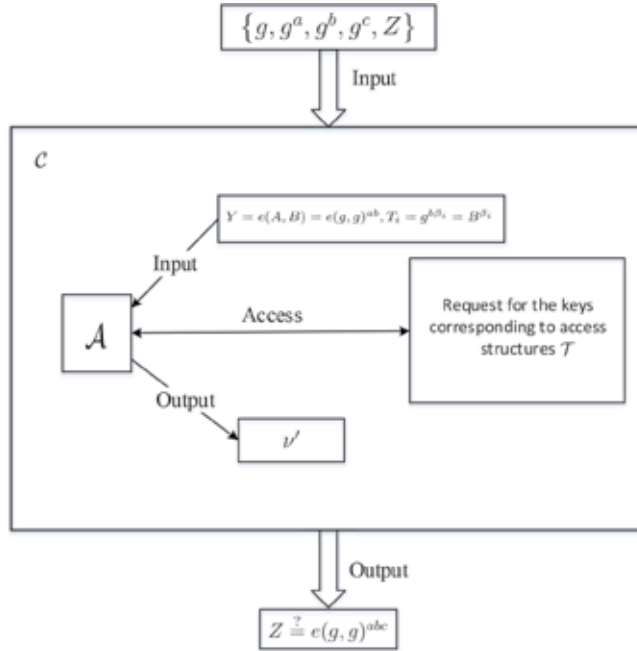


Figure 6.3. Intuition of security proof in the KP-ABE scheme.

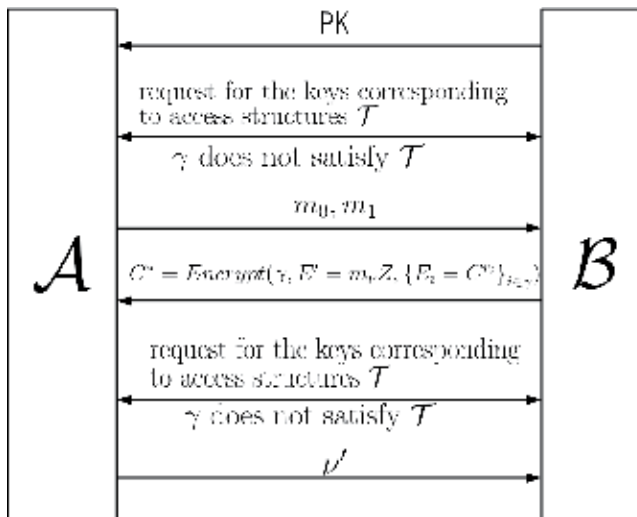


Figure 6.4. Game in the first KP-ABE scheme.

- **Init:** The simulator B runs A . A chooses the set of attributes γ it wishes to be challenged upon.
- **Setup:** The simulator sets the parameter $Y = e(A, B) = e(g, g)^{ab}$. For all $i \in \mathcal{U}$, it sets T_i as follows: if $i \in \gamma$, it chooses a random $r_i \in \mathbb{Z}_p$ and sets $T_i = g^{r_i}$ (thus, $t_i = r_i$); otherwise, it chooses a random $\beta_i \in \mathbb{Z}_p$ and sets $T_i = g^{b\beta_i} = B^{\beta_i}$ (thus, $t_i = b\beta_i$). It then gives the public parameters to A .

- **Phase 1:** \mathcal{A} adaptively makes requests for the keys corresponding to any access structures \mathcal{T} such that the challenge set γ does not satisfy \mathcal{T} . Suppose that \mathcal{A} makes a request for the secret key for an access structure \mathcal{T} where $\mathcal{T}(\gamma) = 0$. To generate the secret key, \mathcal{B} needs to assign a polynomial Q_x of degree d_x for every node in the access tree \mathcal{T} .

We first define the following two procedures: PolySat and PolyUnsat.

- **PolySat**($\mathcal{T}_x, \gamma, \lambda_x$): This procedure sets up the polynomials for the nodes of an access subtree with satisfied root node, that is, $\mathcal{T}_x(\gamma) = 1$. The procedure takes an access tree \mathcal{T}_x (with root node x) as input along with a set of attributes γ and an integer $\lambda_x \in \mathbb{Z}_p$.

It first sets up a polynomial q_x of degree d_x for the root node x . It sets $q_x(0) = \lambda_x$ and then sets rest of the points randomly to completely fix q_x . Now, it sets polynomials for each child node x' of x by calling the procedure PolySat($\mathcal{T}_{x'}$, $\gamma, q_x(\text{index}(x'))$). Notice that in this way, $q_{x'}(0) = q_x(\text{index}(x'))$ for each child node x' of x .

- **PolyUnsat**($\mathcal{T}_x, \gamma, g^{\lambda_x}$): This procedure sets up the polynomials for the nodes of an access tree with unsatisfied root node, that is, $\mathcal{T}_x(\gamma) = 0$. The procedure takes an access tree \mathcal{T}_x (with root node x) as input along with a set of attributes γ and an element $g^{\lambda_x} \in \mathbb{G}_1$ (where $\lambda_x \in \mathbb{Z}_p$).

It first defines a polynomial q_x of degree d_x for the root node x such that $q_x(0) = \lambda_x$. Because $\mathcal{T}_x(\gamma) = 0$, no more than d_x children of x are satisfied. Let $h_x \leq d_x$ be the number of satisfied children of x . For each satisfied child x' of x , the procedure chooses a random point $\lambda_{x'} \in \mathbb{Z}_p$ and sets $q_x(\text{index}(x')) = \lambda_{x'}$. It then fixes the remaining $d_x - h_x$ points of q_x randomly to completely define q_x . Now, the algorithm recursively defines polynomials for the rest of the nodes in the tree as follows. For each child node x' of x , the algorithm calls:

- PolySat($\mathcal{T}_{x'}$, $\gamma, q_x(\text{index}(x'))$), if x' is a satisfied node. Notice that $q_x(\text{index}(x'))$ is known in this case.
- PolyUnsat($\mathcal{T}_{x'}$, $\gamma, g^{q_x(\text{index}(x'))}$), if x' is not a satisfied node. Notice that only $g^{q_x(\text{index}(x'))}$ can be obtained by interpolation as only $g^{q_x(0)}$ is known in this case.

Notice that in this case also, $q_{x'}(0) = q_x(\text{index}(x'))$ for each child node x' of x .

To give keys for access structure \mathcal{T} , the simulator first runs PolyUnsat($\mathcal{T}, \gamma, \mathcal{A}$) to define a polynomial q_x for each node x of \mathcal{T} . Notice that for each leaf node x of \mathcal{T} , we know q_x completely if x is satisfied; if x is not satisfied, then at least $g^{q_x(0)}$ is known (in some cases q_x might be known completely). Furthermore, $q_r(0) = a$.

The simulator now defines the final polynomial $Q_x(\cdot) = bq_x(\cdot)$ for each node x of \mathcal{T} . Notice that this sets $y = Q_r(0) = ab$. The key corresponding to each leaf node is given using its polynomial as follows. Let $i = \text{att}(x)$:

$$D_x = \begin{cases} g^{\frac{Q_x(0)}{i}} = g^{\frac{bq_x(0)}{i}} = g^{\frac{g_x(0)}{i}} & \text{if } i \in \gamma \\ g^{\frac{Q_x(0)}{i}} = g^{\frac{bq_x(0)}{i}} = g^{\frac{g_x(0)}{i}} & \text{otherwise} \end{cases} \quad (6.7)$$

Therefore, the simulator is able to construct a private key for the access structure \mathcal{T} . Furthermore, the distribution of the private key for \mathcal{T} is identical to that in the original scheme.

- **Challenge:** The adversary \mathcal{A} will submit two challenge messages m_0 and m_1 to the simulator. The simulator flips a fair binary coin ν and returns an encryption of m_ν . The ciphertext outputs as

$$E = (\gamma, E' = m_\nu Z, \{E_i = C^{r_i}\}_{i \in \gamma}). \quad (6.8)$$

If $\mu = 0$, then $Z = e(g, g)^{abc}$. If we let $s = c$, then we have $Y^s = (e(g, g)^{ab})^c = e(g, g)^{abc}$ and $E_i = (g^{r_i})^c = C^{r_i}$. Therefore, the ciphertext is a valid random encryption of message m_ν .

Otherwise, if $\mu = 1$, then $Z = e(g, g)^z$. We then have $E' = m_\nu e(g, g)^z$. Since z is random, E' will be a random element of \mathbb{G}_2 from the adversary view, and the message contains no information about m_ν .

- **Phase 2:** The simulator acts exactly as it did in Phase 1.
- **Guess:** \mathcal{A} will submit a guess ν' of ν . If $\nu' = \nu$, the simulator will output $\mu' = 0$ to indicate that it was given a valid BDH-tuple; otherwise, it will output $\mu' = 1$ to indicate it was given a random four-tuple.

As shown in the construction, the simulator's generation of public parameters and private keys is identical to that of the actual scheme.

In the case where $\mu = 1$, the adversary gains no information about ν . Therefore, we have $\Pr[\nu \neq \nu' | \mu = 1] = \frac{1}{2}$. Since the simulator guesses $\mu' = 1$ when $\nu \neq \nu'$, we have $\Pr[\mu' = \mu | \mu = 1] = \frac{1}{2}$.

If $\mu = 0$, then the adversary sees an encryption of m_ν . The adversary's advantage in this situation is ϵ by definition. Therefore, we have $\Pr[\nu = \nu' | \mu = 0] = \frac{1}{2} + \epsilon$. Since the simulator guesses $\mu' = 0$ when $\nu = \nu'$, we have $\Pr[\mu' = \mu | \mu = 0] = \frac{1}{2} + \epsilon$.

The overall advantage of the simulator in the decisional BDH game is $\frac{1}{2} \Pr[\mu' = \mu | \mu = 0] + \frac{1}{2} \Pr[\mu' = \mu | \mu = 1] - \frac{1}{2} = \frac{1}{2} (\frac{1}{2} + \epsilon) + \frac{1}{2} \frac{1}{2} - \frac{1}{2} = \frac{1}{4} \epsilon$.

Chosen-ciphertext security. Our security definitions and proofs have been in the chosen-plaintext model. Similar to [26], we notice that our construction can be extended to the chosen-ciphertext model by applying the technique of using simulation-sound NIZK proofs to achieve chosen-ciphertext security [185]. However, in Section 9 we describe how our delegation mechanism can be used with the techniques of Canetti, et al. [186] to achieve a much more efficient CCA-2 system.

6.3.2 The CP-ABE

Bethencourt et al. [172] presented a scheme for realizing complex access control on encrypted data called ciphertext-policy attribute-based encryption (CP-ABE). In the scheme, attributes are used to describe user's credentials, and a party encrypting data determines a policy for who can decrypt. By using the scenario, encrypted data can be kept confidential even if the storage server is untrusted; moreover, their scenario is secure against collusion attacks.

6.3.2.1 Concrete construction

Let \mathbb{G}_0 be a bilinear group of prime order p , and let g be a generator of \mathbb{G}_0 . In addition, let $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$ denote the bilinear map. A security parameter, k , will determine the size of the groups. We also define the Lagrange coefficient $\Delta_{i, S}$ for $i \in \mathbb{Z}_p$ and a set, S , of elements in \mathbb{Z}_p : $\Delta_{i, S(x)} = \prod_{j \in S, j \neq i} \frac{x-j}{i-j}$. We will additionally

employ a hash function $H: \{0, 1\}^* \rightarrow \mathbb{G}_0$ that we will model as a random oracle. The function will map any attribute described as a binary string to a random group element. Our construction follows.

- **Setup:** The setup algorithm will choose a bilinear group \mathbb{G}_0 of prime order p with generator g . Next, it will choose two random exponents $\alpha, \beta \in \mathbb{Z}_p$. The public key is published as

$$\text{PK} = (\mathbb{G}_0, g, h = g^\beta, f = g^{1/\beta}, e(g, g)^\alpha) \quad (6.9)$$

and the master key (MK) is (β, g^α) . (Note that f is used only for delegation.)

- **Encrypt**(PK, M , T): The encryption algorithm encrypts a message M under the tree access structure T . The algorithm first chooses a polynomial q_x for each node x (including the leaves) in the tree T . These polynomials are chosen in the following way in a top-down manner, starting from the root node R . For each node x in the tree, set the degree d_x of the polynomial q_x to be one less than the threshold value k_x of that node, that is, $d_x = k_x - 1$.

Starting with the root node R , the algorithm chooses a random $s \in \mathbb{Z}_p$ and sets $q_R(0) = s$. Then, it chooses d_R other points of the polynomial q_R randomly to define it completely. For any other node x , it sets $q_x(0) = q_{\text{parent}(x)}(\text{index}(x))$ and chooses d_x other points randomly to completely define q_x .

Let Y be the set of leaf nodes in T . The ciphertext is then constructed by giving the tree access structure T and computing

$$CT = (T, \tilde{C} = Me(g, g)^\alpha, C = h^s, \forall y \in Y : C_y = g^{q_y(0)}, C'_y = H(\text{att}(y))^{q_y(0)}). \quad (6.10)$$

- **KeyGen**(MK, S): The key generation algorithm will take as input a set of attributes S and output a key that identifies with that set. The algorithm first chooses a random $r \in \mathbb{Z}_p$ and then random $r_j \in \mathbb{Z}_p$ for each attribute $j \in S$. Then, it computes the key as

$$SK = (D = g^{(\alpha+r)}/\beta, \forall j \in S : D_j = g^r \cdot H(j)^{r_j}, D'_j = g^{r_j}). \quad (6.11)$$

- **Delegate**(SK, \tilde{S}): The delegation algorithm takes in a secret key SK , which is for a set S of attributes and another set \tilde{S} such that $\tilde{S} \subseteq S$. The secret key is of the form $SK = (D, \forall j \in S : D_j, D'_j)$. The algorithm chooses random \tilde{r} and $\tilde{r}_k \forall k \in \tilde{S}$. Then, it creates a new secret key as

$$\tilde{SK} = (\tilde{D} = Df\tilde{r}, \forall k \in \tilde{S} : \tilde{D}_k = D_k g^{\tilde{r}} H(k)^{\tilde{r}_k}, \tilde{D}'_k = D'_k g^{\tilde{r}_k}). \quad (6.12)$$

The resulting secret key \tilde{SK} is a secret key for the set \tilde{S} . Since the algorithm re-randomizes the key, a delegated key is equivalent to one received directly from the authority.

- **Decrypt**(CT, SK): We specify our decryption procedure as a recursive algorithm. For the ease of exposition, we present the simplest form of the decryption algorithm and discuss potential performance improvements in the next subsection.

We first define a recursive algorithm $\text{DecryptNode}(CT, SK, x)$ that takes as input a ciphertext $CT = (T, \tilde{C}, C, \forall y \in Y : C_y, C'_y)$; a private key SK , which is associated with a set S of attributes; and a node x from T .

If the node x is a leaf node, then we let $i = \text{att}(x)$ and define as follows: If $i \in S$, then

$$\begin{aligned} \text{DecryptNode}(CT, SK, x) &= \frac{e(D_i, C_x)}{e(D'_i, C'_x)} \\ &= \frac{e(g^r \cdot H(i)^{r_i}, g^{q_x(0)})}{e(g^{r_i}, H(i)^{q_x(0)})} \\ &= e(g, g)^{r q_x(0)}. \end{aligned} \quad (6.13)$$

If $i \notin S$, then we define $\text{DecryptNode}(CT, SK, x) = \perp$.

We now consider the recursive case when x is a non-leaf node. The algorithm $\text{DecryptNode}(CT, SK, x)$ then proceeds as follows: For all nodes z that are children of x , it calls $\text{DecryptNode}(CT, SK, z)$ and stores the output as F_z . Let S_x be an arbitrary k_x -sized set of child nodes z such that $F_z \neq \perp$. If no such set exists, then the node was not satisfied and the function returns \perp . Otherwise, we compute

$$\begin{aligned} F_x &= \prod_{z \in S_x} F_z^{A_{i, s'_x(0)}}, \text{ where } \begin{matrix} i = \text{index}(z) \\ S'_x = \{\text{index}(z) : z \in S_x\} \end{matrix} \\ &= \prod_{z \in S_x} (e(g, g)^{r \cdot q_z(0)})^{A_{i, s'_x(0)}} \\ &= \prod_{z \in S_x} (e(g, g)^{r \cdot q_{\text{parent}(z)}(\text{index}(z))})^{A_{i, s'_x(0)}} \quad (\text{by construction}) \\ &= \prod_{z \in S_x} e(g, g)^{r \cdot q_x(i)} \cdot \Delta_{i, s'_x(0)} \\ &= e(g, g)^{r \cdot q_x(0)} \quad (\text{using polynomial interpolation}) \end{aligned} \quad (6.14)$$

and return the result.

Now that we have defined our function DecryptNode , we can define the decryption algorithm. The algorithm begins by simply calling the function on the root node R of the tree T . If the tree is satisfied by S , we set $A = \text{DecryptNode}(CT, SK, r) = e(g, g)^{r q_R(0)} = e(g, g)^{rs}$. The algorithm now decrypts by computing

$$\tilde{C} / (e(C, D) / A) = \tilde{C} / (e(h^s, g^{(\alpha+r)/\beta}) / e(g, g)^{rs}) = M. \quad (6.15)$$

6.3.2.2 Security proof

In this section, we use the generic bilinear group model of [187, 188] and the random oracle model [189] to argue that no efficient adversary that acts generically on the groups underlying our scheme can break the security of our scheme with any reasonable probability. At an intuitive level, this means that if there are any vulnerabilities in our scheme, then these vulnerabilities must exploit specific

mathematical properties of elliptic curve groups or cryptographic hash functions used when instantiating our construction.

While from a security standpoint, it would be preferable to have a proof of security that reduces the problem of breaking our scheme to a well-studied complexity-theoretic problem; there is a reason to believe that such reductions will only exist for more complex (and less efficient) schemes than the one we give here. We also stress that ours is the first construction which offers the security properties we are proposing here; we strongly encourage further research that can place this kind of security on a firmer theoretical foundation.

The generic bilinear group model. We follow [187] here: We consider two random encodings ψ_0, ψ_1 of the additive group \mathbb{F}_p , that is, injective maps $\psi_0, \psi_1: \mathbb{F}_p \rightarrow \{0, 1\}^m$, where $m > 3\log(p)$. For $i = 0, 1$ we write $\mathbb{G}_i = \{\psi_i(x) : x \in \mathbb{F}_p\}$. We are given oracles to compute the induced group action on $\mathbb{G}_0, \mathbb{G}_1$ and an oracle to compute a nondegenerate bilinear map $e: \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$. We are also given a random oracle to represent the hash function H . We refer to \mathbb{G}_0 as a generic bilinear group.

The following theorem gives a lower bound on the advantage of a generic adversary in breaking our CP-ABE scheme.

Theorem 1. *Let $\psi_0, \psi_1, \mathbb{G}_0, \mathbb{G}_1$ be defined as above. For any adversary A , let q be a bound on the total number of group elements it receives from queries it makes to the oracles for the hash function, groups \mathbb{G}_0 and \mathbb{G}_1 , and the bilinear map e and from its interaction with the CP-ABE security game. Then, we have presented that the advantage of the adversary in the CP-ABE security game is $O(q^2/p)$.*

Proof. We first make the following standard observation, which follows from a straightforward hybrid argument: In the CP-ABE security game, the challenge ciphertext has a component \tilde{C} which is randomly either $M_0e(g, g)^\alpha$ or $M_1e(g, g)^\alpha$. We can instead consider a modified game in which \tilde{C} is either $e(g, g)^\alpha$ or $e(g, g)^\theta$, where θ is selected uniformly at random from \mathbb{F}_p , and the adversary must decide which is the case. It is clear that any adversary that has advantage ϵ in the CP-ABE game can be transformed into an adversary that has advantage at least $\epsilon/2$ in the modified CP-ABE game. (To see this consider two hybrids: one in which the adversary must distinguish between $M_0e(g, g)^\alpha$ and $e(g, g)^\theta$ and another in which it must distinguish between $e(g, g)^\theta$ and $\text{textit}M_1e(g, g)^\alpha$. Clearly, both of these are equivalent to the modified game above.) From now on, we will bind the adversary's advantage in the modified game. The intuition of security proof in the scheme is illustrated in **Figure 6.5**.

We now introduce some notation for the simulation of the modified CP-ABE game. Let $g = \psi_0(1)$ (we will write g^x to denote $\psi_0(x)$ and $e(g, g)^y$ to denote $\psi_1(y)$ in the future). The whole game is described in **Figure 6.6**.

At setup time, the simulation chooses α, β at random from \mathbb{F}_p (which we associate with the integers from 0 to $p - 1$). Note that if $\beta = 0$, an event that happens with probability $1/p$, then setup is aborted, just as it would be in the actual scheme. The public parameters $h = g^\beta, f = g^{1/\beta}$, and $e(g, g)^\alpha$ are sent to the adversary.

When the adversary (or simulation) calls for the evaluation of H on any string i , a new random value t_i is chosen from \mathbb{F}_p (unless it has already been chosen), and the simulation provides g^{t_i} as the response to $H(i)$.

When the adversary makes its j th key generation query for the set S_j of attributes, a new random value $r^{(j)}$ is chosen from \mathbb{F}_p , and for every $i \in S_j$, new random values $r_i^{(j)}$ are chosen from \mathbb{F}_p . The simulator then computes $D = g^{(\alpha+r^{(j)})/\beta}$, and for each $i \in S_j$, we have $D_i = g^{r^{(j)}+t_i r_i^{(j)}}$ and $D'_i = g^{r_i^{(j)}}$. These values are passed onto the adversary.

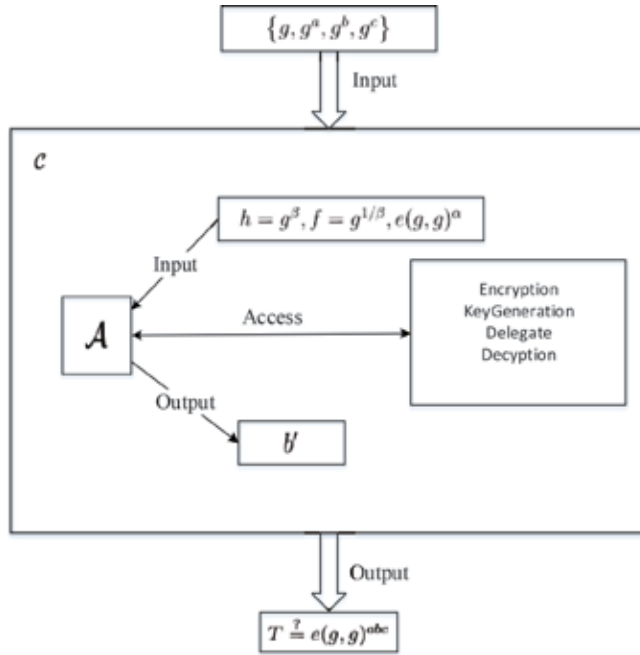


Figure 6.5.
Intuition of security proof in the CP-ABE scheme.

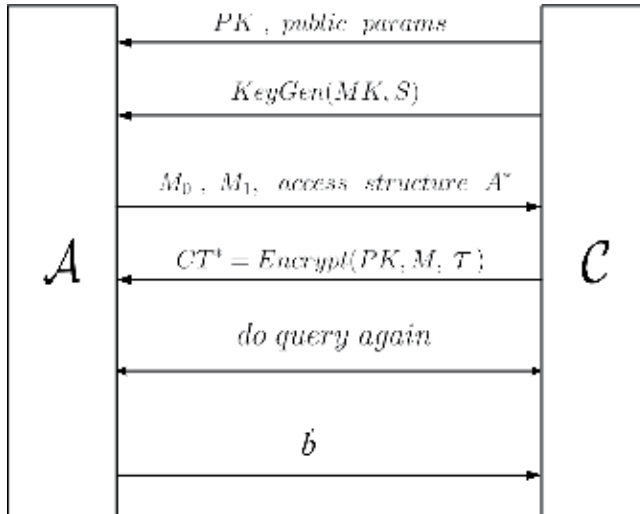


Figure 6.6.
Game in the CP-ABE scheme.

When the adversary asks for a challenge, giving two messages $M_0, M_1 \in \mathbb{G}_1$, and the access tree \mathbb{A} , the simulator does the following. First, it chooses a random s from \mathbb{F}_p . Then, it uses the linear secret-sharing scheme associated with \mathbb{A} (as described in Section 4) to construct shares λ_i of s for all relevant attributes i . We stress again that λ_i is all chosen uniformly and independently at random from \mathbb{F}_p subject to the linear conditions imposed on them by the secret-sharing scheme. In particular, the choice of the λ_i s can be perfectly simulated by choosing l random values μ_1, \dots, μ_l uniformly and independently from \mathbb{F}_p , for some value of l , and then letting λ_i be fixed public

linear combinations of the μ_k s and s . We will often think of λ_i as written as such linear combinations of these independent random variables later.

Finally, the simulation chooses a random $\theta \in \mathbb{F}_p$ and constructs the encryption as follows: $\tilde{C} = e(g, g)^\theta$ and $C = h^s$. For each relevant attribute i , we have $C_i = g^{\lambda_i}$ and $C'_i = g^{t_i \lambda_i}$. These values are sent to the adversary.

(Note, of course, that if the adversary asks for a decryption key for a set of attributes that pass the challenge access structure, then the simulation does not issue the key; similarly, if the adversary asks for a challenge access structure such that one of the keys already issued pass the access structure, then the simulation aborts and outputs a random guess on behalf of the adversary, just as it would in the real game.)

We will show that with probability $1 - O(q^2/p)$, taken over the randomness of the choice of variable values in the simulation, the adversary's view in this simulation is identically distributed to what its view would have been if it had been given $\tilde{C} = e(g, g)^{\alpha s}$. We will therefore conclude that the advantage of the adversary is at most $O(q^2/p)$, as claimed.

When the adversary makes a query to the group oracles, we may condition on the event that (1) the adversary only provides as input values it received from the simulation, or intermediate values it already obtained from the oracles, and (2) there are p distinct values in the ranges of both ψ_0 and ψ_1 . (This event happens with overwhelming probability $1 - O(1/p)$.) As such, we may keep track of the algebraic expressions being called for from the oracles, as long as no "unexpected collisions" happen. More precisely, we think of an oracle query as being a rational function $\nu = \eta/\xi$ in the variables $\theta, \alpha, \beta, t_i$'s, $r_i^{(j)}$'s, s, s, s , and μ_k 's. An unexpected collision would be when two queries correspond to two distinct formal rational functions $\eta/\xi \neq \eta'/\xi'$, but due to the random choices of these variables' values, we have presented that the values of η/ξ and η'/ξ' coincide.

We now condition on the event that no such unexpected collisions occur in either group \mathbb{G}_0 or \mathbb{G}_1 . For any pair of queries (within a group) corresponding to distinct rational functions η/ξ and η'/ξ' , a collision occurs only if the nonzero polynomial $\eta\xi' - \xi\eta'$ evaluates to zero. Note that the total degree of $\eta\xi' - \xi\eta'$ is in our case at most 5. By the Schwartz-Zippel lemma [190, 191], the probability of this event is $O(1/p)$. By a union bound, the probability that any such collision happens is at most $O(q^2/p)$. Thus, we can condition on no such collision happening and still maintain $1 - O(q^2/p)$ of the probability mass.

Now, we consider what the adversary's view would have been if we had set $\theta = \alpha s$. We will show that subject to the conditioning above, the adversary's view would have been identically distributed. Since we are in the generic group model where each group element's representation is uniformly and independently chosen, the only way that the adversary's view can differ in the case of $\theta = \alpha s$ is if there are two queries ν and ν' into \mathbb{G}_1 such that $\nu \neq \nu'$ but $\nu|_{\theta=\alpha s} = \nu'|_{\theta=\alpha s}$. We will show that this never happens. Suppose not.

Recall that since θ only occurs as $e(g, g)^\theta$, which lives in \mathbb{G}_1 , the only dependence that ν or ν' can have on θ is by having some additive terms of the form $\gamma'\theta$, where γ' is a constant. Therefore, we must have that $\nu - \nu' = \gamma\alpha s - \gamma\theta$, for some constant $\gamma \neq 0$. We can then artificially add the query $\nu - \nu' + \gamma\theta = \gamma\alpha s$ to the adversary's queries. But we will now show that the adversary can never construct a query for $e(g, g)^{\gamma\alpha s}$ (subject to the conditioning we have already made), which will reach a contradiction and establish the theorem.

What is left now is to do a case analysis based on the information given to the adversary by the simulation. For the sake of completeness and ease of reference for

$t_i t_{i'}$	$\lambda_i t_{i'}$	$t_i t_{i'} \lambda_{i'}$	$t_i r^{(j)} + t_i t_{i'} r_{i'}^{(j)}$
$t_i r_i^{(j)}$	t_i	$\alpha + r^{(j)}$	$\alpha s + s r^{(j)}$
$\lambda_i \lambda_{i'}$	$t_i \lambda_i \lambda_{i'}$	$\lambda_{i'} r^{(j)} + \lambda_{i'} t_i r_i^{(j)}$	$\lambda_i r_i^{(j)}$
λ_i	$t_i t_{i'} \lambda_i \lambda_{i'}$	$t_i \lambda_i r_i^{(j)} + t_i t_{i'} \lambda_i r_{i'}^{(j)}$	$t_i \lambda_i r_{i'}^{(j)}$
$t_i \lambda_i$	$(r^{(j)} + t_i r_i^{(j)})(r^{(j)} + t_{i'} r_{i'}^{(j)})$	$(r^{(j)} + t_i r_i^{(j)}) r_{i'}^{(j)}$	$r^{(j)} + t_i r_i^{(j)}$
$r_i^{(j)} r_{i'}^{(j)}$	$r_i^{(j)}$	s	

Table 6.1.
Possible query types from the adversary.

the reader, in **Table 6.1**, we enumerate overall rational function queries possible into \mathbb{G}_1 by means of the bilinear map and the group elements given the adversary in the simulation, except those in which every monomial involves the variable β , since β will not be relevant to constructing a query involving αs . Here, the variables i and i' are possible attribute strings, and the variables j and j' are the indices of secret key queries made by the adversary. These are given in terms of λ_i s, not μ_k s. The reader may check the values given in **Table 6.1** against the values given in the simulation above.

In the group \mathbb{G}_1 , in addition to the polynomials in the table above, the adversary also has access to 1 and α . The adversary can query for arbitrary linear combinations of these, and we must show that none of these polynomials can be equal to a polynomial of the form $\gamma \alpha s$. Recall that $\gamma \neq 0$ is a constant.

As seen above, the only way that the adversary can create a term containing αs is by pairing $s\beta$ with $(\alpha + r^{(j)})/\beta$ to get the term $\alpha s + s r^{(j)}$. In this way, the adversary could create a query polynomial containing $\gamma \alpha s + \sum_{j \in T} \gamma_j s r_j^{(j)}$, for some set T and constants $\gamma, \gamma_j \neq 0$.

In order for the adversary to obtain a query polynomial of the form $\gamma \alpha s$, the adversary must add other linear combinations in order to cancel the terms of the form $\sum_{j \in T} \gamma_j s r_j^{(j)}$.

We observe (by referencing the table above) that the only other term that the adversary has access to that could involve monomials of the form $s r^{(j)}$ is obtained by pairing $r^{(j)} + t_i r_i^{(j)}$ with some $\lambda_{i'}$, since the $\lambda_{i'}$ terms are linear combinations of s and the μ_k s.

In this way, for sets T'_j and constants $\gamma(i, j, i') \neq 0$, the adversary can construct a query polynomial of the form

$$\gamma \alpha s + \sum_{j \in T} (\gamma_j s r^{(j)} + \sum_{(i, i') \in T'_j} \gamma(i, j, i') (\lambda_{i'} r^{(j)} + \lambda_{i'} t_i r_i^{(j)})) + \text{other terms.} \quad (6.16)$$

Now, to conclude this proof, we do the following case analysis:

- **Case 1:** There exists some $j \in T$ such that the set of secret shares $L_j = \{\lambda_{i'} : \exists i : (i, i') \in T'_j\}$ do not allow for the reconstruction of the secret s .

If this is true, then the term $s r^{(j)}$ will not be canceled, and so the adversary's query polynomial cannot be of the form $\gamma \alpha s$.

- **Case 2:** For all $j \in T$, the set of secret shares $L_j = \{\lambda_{i'} : \exists i : (i, i') \in T'_j\}$ does allow for the reconstruction of the secret s .

Fix any $j \in T$. Consider S_j , the set of attributes belonging to the j th adversary key request. By the assumption that no requested key should pass the challenge access structure, and the properties of the secret-sharing scheme, we know that the set $L'_j = \lambda_i : i \in S_j$ cannot allow for the reconstruction of s .

Thus, there must exist at least one share $\lambda_{i'}$ in L_j such that $\lambda_{i'}$ is linearly independent of L'_j when written in terms of s and the $\mu_k s$. By the case analysis, this means that in the adversary's query there is a term of the form $\lambda_{i'} t_i r_i^{(j)}$ for some $i \in S_j$. However, (examining the table above), there is no term that the adversary has access to that can cancel this term. Therefore, any adversary query polynomial of this form cannot be of the form $\lambda \alpha s$.

6.3.3 The ABE with outsourced decryption

6.3.3.1 Concrete construction

1. Ciphertext-policy attribute-based encryption (CP-ABE)

- **Setup**(λ, U): The setup algorithm takes as input a security parameter and a universe description U . To cover the most general case, we let $U = \{0, 1\}^*$. It then chooses a group \mathbb{G} of prime order p , a generator g , and a hash function F that maps $0, 1^*$ to \mathbb{G}^a .

In addition, it chooses random exponents $\alpha, a \in \mathbb{Z}_p$. The authority sets $MSK = (g^\alpha, P)$ as the master secret key. It publishes the public parameters as

$$PK = g, e(g, g)^\alpha, g^a, F \quad (6.17)$$

Encrypt($PK, \mathcal{M}, (M, \rho)$): The encryption algorithm takes as input the public parameters PK and a message \mathcal{M} to encrypt. In addition, it takes as input an LSSS access structure (M, ρ) . The function ρ associates rows of M to attributes. Let M be an $l \times n$ matrix. The algorithm first chooses a random vector $\vec{v} = (s, y_2, \dots, y_n) \in \mathbb{Z}_p^n$. These values will be used to share the encryption exponent s . For $i = 1$ to l , it calculates $\lambda_i = \vec{v} \cdot M_i$, where M_i is the vector corresponding to the i th row of M . In addition, the algorithm chooses random $r_1, \dots, r_l \in \mathbb{Z}_p$. The ciphertext is published as $CT =$

$$C = \mathcal{M} \cdot e(g, g)^{\alpha s}, C' = g^s, \quad (6.18)$$

$$(C_1 = G^{a\lambda_1} \cdot F(\rho(1))^{(-r_1)}, D_1 = G^{r_1}), \dots, (C_l = g^{a\lambda_l} \cdot F(\rho(l)))^{-r_l}, D_l = g^{r_l}$$

along with a description of (M, ρ) .

- **KeyGen_{out}**(MSK, S): The key generation algorithm runs **KeyGen** (MSK, S) to obtain $SK' = (PK, K' = g^\alpha g^{at'}, L' = g^{t'}, \{K'_x\} = F(x)_{x \in S}^t)$. It chooses a random value $z \in \mathbb{Z}_p^*$. It sets the transformation key TK as

$$PK, K = K'^{1/z} = g^{(\alpha/z)} g^a (t'/z) = g^{(\alpha/z)} g^{at}, L = L'^{1/z} = g^{t'/z} = g^t, \{K_x\}_{x \in S} = \{K'\} \quad (6.19)$$

and the private key SK as (z, TK) .

- **Transform_{out}**(TK, CT): The transformation algorithm takes as input a transformation key $TK = (PK, K, L, \{K_x\}_{x \in S})$ for a set S and a ciphertext $CT = (C, C', C_1, \dots, C_l)$ for access structure (M, ρ) . If S does not satisfy the access structure, it outputs \perp . Suppose that S satisfies the access structure, and let $I \subset \{1, 2, \dots, l\}$ be defined as $I = \{i : \rho(i) \in S\}$. Then, let $\omega_i \in \mathbb{Z}_{i \in I}$ be a set of constants such that if $\{\lambda_i\}$ is a valid share of any secret s according to M , then $\sum_{i \in I} \omega_i \lambda_i = s$. The transformation algorithm computes

$$\begin{aligned} & e(C', K) / (e(\prod_{i \in I} C_i^{\omega_i}, L) \cdot \prod_{i \in I} e(D_i^{\omega_i, K_{\rho(i)}})) \\ & = e(g, g)^{s\alpha/z} e(g, g)^{ast} / (\prod_{i \in I} e(g, g)^{t\lambda_i \omega_i}) = e(g, g)^{s\alpha/z}. \end{aligned} \quad (6.20)$$

It outputs the partially decrypted ciphertext CT' as $(C, e(g, g)^{s\alpha/z})$, which can be viewed as the ElGamal ciphertext $(\mathcal{M} \cdot G^{zd}, G^d)$ where $G = e(g, g)^{1/z} \in \mathcal{G}_T$ and $d = s\alpha \in \mathbb{Z}_p$.

- **Decrypt_{out}**(SK, CT): The decryption algorithm takes as input a private key $SK = (z, TK)$ and a ciphertext CT . If the ciphertext is not partially decrypted, then the algorithm first executes transformout (TK, CT). If the output is \perp , then this algorithm outputs \perp as well. Otherwise, it takes the ciphertext (T_0, T_1) and computes $T_0/T_1^z = \mathcal{M}$.

Notice that if the ciphertext is already partially decrypted for the user, then she needs to only compute one exponentiation and no pairings to recover the message.

2. Key-policy attribute-based encryption (KP-ABE)

- **Setup**(λ, U): The setup algorithm takes as input a security parameter and a universe description U . To cover the most general case, we let $U = \{0, 1\}^*$. It then chooses a group \mathbb{G} of prime order p , a generator g , and a hash function F that maps $\{0, 1\}^*$ to \mathbb{G}^a . In addition, it chooses random exponents $\alpha \in \mathbb{Z}_p$ and $h \in \mathbb{G}$. The authority sets $MSK = (\alpha, PK)$ as the master secret key. It publishes the public parameters as

$$PK = g, g^\alpha, h, F. \quad (6.21)$$

- **Encrypt**(PK, \mathcal{M}, S): The encryption algorithm takes as input the public parameters PK , a message \mathcal{M} to encrypt, and a set of attributes S . It chooses a random $s \in \mathbb{Z}_p$. The ciphertext is published as $CT = (S, C)$ where

$$C = \mathcal{M} \cdot e(g, h)^\alpha, C' = g^s, \{C_x = F(x)^s\}_{x \in S}. \quad (6.22)$$

- **KeyGen_{out}**($MSK, (M, \rho)$): Parse $MSK = (\alpha, \alpha PK)$. The key generation algorithm runs **KeyGen** $((\alpha, PK), (M, \rho))$ to obtain $SK' = (PK, (D'_1 = h^{\lambda_1} \cdot F(\rho(1))^{r'_1}, R'_1 = g^{r'_1}), \dots, (D'_l, R'_l))$. Next, it chooses a random value $z \in \mathbb{Z}_p$, computes the transformation key TK as below, and outputs the private key as (z, TK) . Denoting r'_i/z as r_i , TK is computed as

$$PK, (D_1 = D'_1^{1/z} = h^{\lambda_1/z} \cdot F(\rho(1))^{r_1}, R_1 = R'_1^{1/z} = g^{r_1}), \dots, (D_l = D'_l^{1/z}, R_l = R'_l^{1/z}). \quad (6.23)$$

- **Transform_{out}**(TK, CT): The transformation algorithm takes as input a transformation key $TK = (PK, (D_1, R_1), \dots, (D_l, R_l))$ for access structure (M, ρ) and a ciphertext $CT = (C, C', C_{xx \in S})$ for set S . If S does not satisfy the access structure, it outputs \perp . Suppose that S satisfies the access structure, and let $I \subset 1, 2, \dots, l$ be defined as $I = i : \rho(i) \in S$. Then, let $\omega_i \in \mathbb{Z}_p$ be a set of constants such that if λ_i is a valid share of any secret s according to M , then $\sum_{i \in I} \omega_i \lambda_i = s$. The transformation algorithm computes

$$\begin{aligned} e(C', \prod_{i \in I} D_i^{\omega_i}) / (\prod_{i \in I} e(R_i, C_{\rho(i)}^{\omega_i})) &= e(g^s, \prod_{i \in I} h^{\lambda_i \omega_i / z} \cdot F(\rho(i))^{r_i \omega_i}) / (\prod_{i \in I} e(g^{r_i}, F(\rho(i))^{s \omega_i})) \\ &= e(g, h)^{s \alpha / z} \cdot \prod_{i \in I} e(g^s, F(\rho(i))^{r_i \omega_i}) / (\prod_{i \in I} e(g^{r_i}, F(\rho(i))^{s \omega_i})) = e(g, h)^{s \alpha / z} \end{aligned} \quad (6.24)$$

It outputs the partially decrypted ciphertext CT' as $(C, e(g, h)^{s \alpha / z})$, which can be viewed as the ElGamal ciphertext $(\mathcal{M} \cdot G^{zd}, G^d)$ where $G = e(g, h)^{1/z} \in \mathcal{G}_T$ and $d = s \alpha \in \mathbb{Z}_p$.

- **Decrypt_{out}**(SK, CT): The decryption algorithm takes as input a private key $SK = (z, TK)$ and a ciphertext CT . If the ciphertext is not partially decrypted, then the algorithm first executes **Transform_{out}**(TK, CT). If the output is \perp , then this algorithm outputs \perp as well. Otherwise, it takes the ciphertext (T_0, T_1) and computes $T_0 / T_1^z = \mathcal{M}$.

6.3.3.2 Security proof

Proof. Suppose that there exists a polynomial-time adversary \mathcal{A} that can attack our scheme in the selective RCCA-security model for outsourcing with advantage ε . We build a simulator \mathcal{B} that can attack the Waters scheme of [177] in the selective CPA-security model with advantage ε minus a negligible amount. In [177] the Waters scheme is proven secure under the decisional q-parallel BDHE assumption. The intuition of security proof in the scheme is illustrated in **Figure 6.7**.

- **Init:** The simulator \mathcal{B} runs \mathcal{A} . \mathcal{A} chooses the challenge access structure (M^*, ρ^*) , in which \mathcal{B} passes on to the Waters challenger as the structure on which it wishes to be challenged.
- **Set up:** The simulator \mathcal{B} obtains the Waters public parameters $PK = g, e(g, g)^\alpha, g^a$ and a description of the hash function F . It sends these to \mathcal{A} as the public parameters.
- **Phase 1:** The simulator \mathcal{B} initializes empty tables T, T_1, T_2 , an empty set D , and an integer $j = 0$. It answers the adversary's queries as follows:
 - Random oracle hash $H_1(R, \mathcal{M})$: If there is an entry (R, \mathcal{M}, s) in T_1 , return s . Otherwise, choose a random $s \in \mathbb{Z}_p$, record (R, \mathcal{M}, s) in T_1 , and return s .
 - Random oracle hash $H_2(R)$: If there is an entry (R, r) in T_2 , return r . Otherwise, choose a random $r \in \{0, 1\}^k$, record (R, r) in T_2 , and return r .

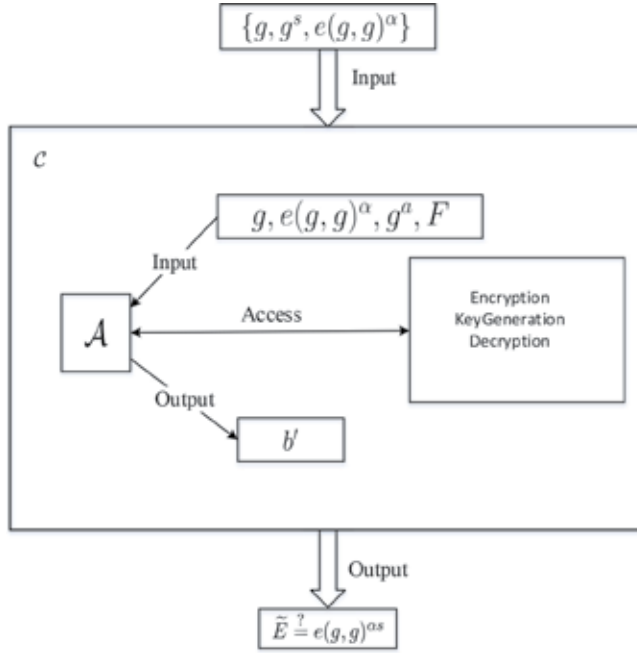


Figure 6.7.
Intuition of security proof in the ABE with outsourced decryption.

- Create $((S))$: \mathcal{B} sets $j := j + 1$. It now proceeds one of the two ways.
 - If S satisfies (M^*, ρ^*) , then it chooses a “fake” transformation key as follows: Choose a random $d \in \mathbb{Z}_p$ and run $\text{KeyGen}((d, PK), S)$ to obtain SK' . Set $TK = SK'$ and set $SK = (d, TK)$. Note that the pair (d, TK) is not well formed, but that TK is properly distributed if d was replaced by the unknown value $z = \alpha/d$.
 - Otherwise, it calls the Waters key generation oracle on S to obtain the key $SK' = (PK, K', L', \{K'_x\}_{x \in S})$. (Recall that in the non-outsourcing CP-ABE game, the create and corrupt functionalities are combined in one oracle.) The algorithm chooses a random value $z \in \mathbb{Z}_p$ and sets the transformation key TK as $(PK, K = K'^{1/z}, L = L'^{1/z}, \{K_x\}_{x \in S} = \{K'_x\}_{x \in S})$ and the private key as (z, TK) .

Finally, store (j, S, SK, TK) in table T and return TK to \mathcal{A} .

- Corrupt (i) : \mathcal{A} cannot ask to corrupt any key corresponding to the challenge structure (M^*, ρ^*) . If there exists an i th entry in table T , then \mathcal{B} obtains the entry (i, S, SK, TK) and sets $D := D \cup \{S\}$. It then returns SK to \mathcal{A} or \perp if no such entry exists.
- Decrypt (i, CT) : Without loss of generality, we assume that all ciphertexts input to this oracle are already partially decrypted. Recall that both \mathcal{B} and \mathcal{A} have access to the TK values for all keys created, so either can execute the transformation operation. Let $CT = (C_0, C_1, C_2)$ be associated with structure (M^*, ρ^*) . Obtain the record (i, S, SK, TK) from table T . If it is not there or $S \notin (M, \rho)$, return \perp to \mathcal{A} (**Figure 6.8**).

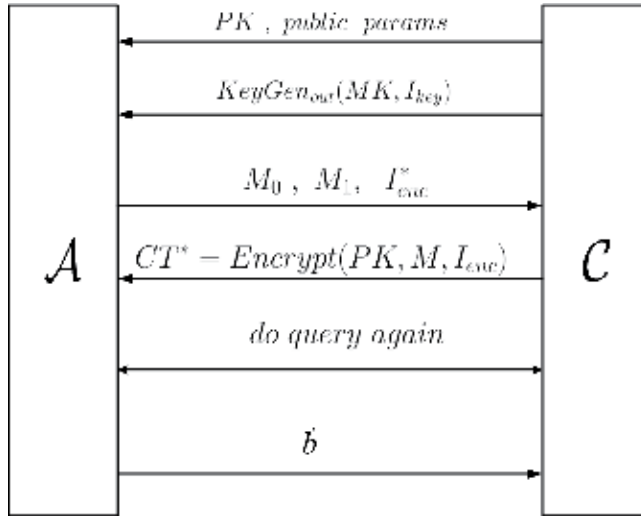


Figure 6.8.
 Game in the ABE with outsourced decryption scheme.

If key i does not satisfy the challenge structure (M^*, ρ^*) , proceed as follows:

- Parse $SK = (z, TK)$. Compute $R = C_0/C_2^z$.
- Obtain the records (R, \mathcal{M}_i, s_i) from table T_1 . If none exist, return \perp to \mathcal{A} .
- If in this set, there exists indices $y \neq x$ such that (R, \mathcal{M}_y, s_y) and (R, \mathcal{M}_x, s_x) are in table T_1 , $\mathcal{M}_y \neq \mathcal{M}_x$, and $s_y = s_x$, then \mathcal{B} aborts the simulation.
- Otherwise, obtain the record (R, r) from table T_2 . If it does not exist, \mathcal{B} outputs \perp .
- For each i , test if $C_0 = R \cdot e(g, g)^{\alpha s_i}$, $C_1 = \mathcal{M}_i \oplus r$ and $C_2 = e(g, g)^{\alpha s_i/z}$.
- If there is an i that passes the above test, output the message \mathcal{M}_i ; otherwise, output \perp . (Note: at most one value of s_i , and thereby one index i , can satisfy the third check of the above test.)

If key i does satisfy the challenge structure (M^*, ρ^*) , proceed as follows:

- Parse $SK = (d, TK)$. Compute $\beta = C_2^{1/d}$.
- For each record $(R_i, \mathcal{M}_i, s_i)$ in table T_1 , test if $\beta = e(g, g)^{s_i}$.
- If zero matches are found, \mathcal{B} outputs \perp to \mathcal{A} .
- If more than one matches are found, \mathcal{B} aborts the simulation.
- Otherwise, let (R, \mathcal{M}, s) be the sole match. Obtain the record (R, r) from table T_2 . If it does not exist, \mathcal{B} outputs \perp .
- Test if $C_0 = R \cdot e(g, g)^{\alpha s}$, $C_1 = \mathcal{M} \oplus r$ and $C_2 = e(g, g)^{ds}$.
- If all tests pass, output \mathcal{M} ; else, output \perp .

- **Challenge:** Eventually, \mathcal{A} submits a message pair $(\mathcal{M}_0^*, \mathcal{M}_1^*) \in \{0, 1\}^{2 \times k}$. \mathcal{B} acts as follows:
 - \mathcal{B} chooses random “messages” $(\mathcal{R}_0, \mathcal{R}_1) \in \mathbb{G}_T^2$ and passes them on to the Waters challenger to obtain a ciphertext $CT = (C, C', \{C_i\}_{i \in [1, l]})$.
 - \mathcal{B} chooses a random value $C'' \in \{0, 1\}^k$.
 - \mathcal{B} sends to \mathcal{A} the challenge ciphertext $CT^* = (C, C', C'', \{C_i\}_{i \in [1, l]})$.
- **Phase 2:** The simulator \mathcal{B} continues to answer queries as in Phase 1, except that if the response to a Decrypt query would be either \mathcal{M}_0^* or \mathcal{M}_1^* , then \mathcal{B} responds with the message *test* instead.
- **Guess:** Eventually, \mathcal{A} must either output a bit or abort, either way \mathcal{B} ignores it. Next, \mathcal{B} searches through tables T_1 and T_2 to see if the values \mathcal{R}_0 and \mathcal{R}_1 appear as the first element of any entry (i.e., that \mathcal{A} issued a query of the form $H_1(\mathcal{R}_i \cdot)$ or $H_2(\mathcal{R}_i)$.) If neither or both values appear, \mathcal{B} outputs a random bit as its guess. If only value \mathcal{R}_b appears, then \mathcal{B} outputs b as its guess.

6.3.4 The ABE with outsourced encryption

In 2012, Li et al. [28] formulized the paradigm of outsourcing encryption of ABE in cloud computing. They utilized MapReduce to propose a security-enhanced construction which is secure under the assumption that the master node as well as at least one of the slave nodes in cloud is honest. Another advantage of the proposed construction is that it is able to delegate encryption for any access policy, instead of a special hybrid access policy.

6.3.4.1 Concrete construction

- **Setup**(λ): The setup algorithm is executed by authority. Select a bilinear group \mathbb{G} of prime order q with generator g and two random integers $\alpha, \beta \in \mathbb{Z}_q$, and define a hash function $H: \{0, 1\}^* \rightarrow \mathbb{G}$ modeled as a random oracle. Finally, output the public key $PK = (\mathbb{G}, H(\cdot), g, h = g^\beta, e(g, g)^\alpha)$ and the master key $MK = (\beta, g^\alpha)$.
- **KeyGen**(ω, MK): For each user’s private key request, the authority runs the key generation algorithm. Choose $r \in {}_R\mathbb{Z}_q$ and $r_j \in {}_R\mathbb{Z}_q$ for each attribute $j \in \omega \cup \{attr(\theta)\}$. Finally, compute and output the private key as

$$SK = (d = g^{\frac{\alpha+r}{\beta}}, d_{j0} = g^r \cdot H(j)^{r_j}, d_{j1} = g^{r_j})_{j \in \omega \cup \{attr(\theta)\}}. \quad (6.25)$$

- **Encrypt** $_U(\mathcal{T}_U, M)$: To encrypt a message M with access policy \mathcal{T}_U , firstly pick an integer $s \in {}_R\mathbb{Z}_q$, and randomly select a 1-degree polynomial $q_R(\cdot)$ such that $q_R(0) = s$. Furthermore, let $s_1 = q_R(1)$ and $s_2 = q_R(2)$. Then, make n -splits on s_1 by randomly selecting $s_{11}, \dots, s_{1n} \in \mathbb{Z}_q$ with $s_{11} + \dots + s_{1n} = s_1$, and set $OEK_i = s_{1i}$ for $i = 1, \dots, n$. Finally, it outputs the partially encrypted ciphertext at local $CT_U = (\mathcal{T}_U \wedge \mathcal{T}_\theta, \tilde{E} = Me(g, g)^\alpha, E = h^s, (E_{\theta 0} = g^{s_2}, E_{\theta 1} = H(attr(\theta))^{s_2}))$ and the set of outsourcing keys $\{OEK_i\}_{i=1}^n$.

- **Encrypt_{MR}**($\mathcal{T}_U, \{OEK_i\}_{i=1}^n$): The delegated encryption algorithm is executed by the MapReduce cloud. User uploads $(\mathcal{T}_U, \{OEK_i\})$ which is scanned as the key-value pair to the i th slave node. Then, the MapReduce is triggered.
 - **Map**: The slave node i finishes the partial encryption task by choosing a $(k_x - 1)$ -degree polynomial $q_x^{(i)}(\cdot)$ for each node x (including the leaves) in the tree \mathcal{T}_U in a top-down manner. The selected polynomial $q_x^{(i)}(\cdot)$ must satisfy the restriction that $q_x^{(i)}(0) = s_{i1}$ if x is the root node in \mathcal{T}_U ; otherwise, $q_x^{(i)}(0) = q_{parent(x)}^{(i)}(index(x))$. Let Y_U denote the set of leaf nodes in \mathcal{T}_U ; then, the partially encrypted ciphertext at slave node i is computed as $CT_{MR}^{(i)} = (\{E_{y0}^{(i)} = g^{q_y^{(i)}(0)}, E_{y1}^{(i)} = H(attr(y))^{q_y^{(i)}(0)}\}_{y \in Y_U})$. The map function for outsourced encryption is described as $Map(\mathcal{T}_U, OEK_i) \rightarrow (\mathcal{T}_U, CT_{MR}^{(i)})$.
 - **Reduce**: Let $q_y(x) = \sum_{i=0}^n q_y^{(i)}(x)$ for $y \in Y_U$. The master node is selected as the reducer. Then, after gathering all the intermediate key-value pairs $\{(\mathcal{T}_U, CT_{MR}^{(i)})\}_{i=1}^n$ sent from the other slave nodes, reducer computes $CT_{MR} = (\{E_{y0} = \prod_{i=1}^n E_{y0}^{(i)} = g^{q_y(0)}, E_{y1} = \prod_{i=1}^n E_{y1}^{(i)} = H(attr(y))^{q_y(0)}\}_{y \in Y_U})$. The reduce function for outsourced encryption can be described as $Reduce(\{(\mathcal{T}_U, CT_{MR}^{(i)})\}_{i=1}^n) \rightarrow CT_{MR}$.

Finally, such ciphertext CT_{MR} is sent back to user.

- **Decrypt**(CT, SK): The recursive decryption algorithm is executed by user. Suppose that $CT = (CT_U, CT_{MR})$ is encrypted under the policy corresponding to SK ; then, the decryption is followed in a down-top manner.
 - For each leaf node y in the hybrid access tree $\mathcal{T}_U \wedge \mathcal{T}_\theta$, let $i = attr(y)$ and the decryption is presented as follows:

$$F_y = \frac{e(E_{y0}, d_{i0})}{e(d_{i1}, E_{y1})} = \frac{e(g^{q_y(0)}, g^r H(i)^{r_i})}{g^{r_i}, H(attr(y))^{q_y(0)}} = e(g, g)^{rq_y(0)}. \quad (6.26)$$
 - For each interior node y , let S_y be an arbitrary k_y -sized set of child nodes z such that $F_z \neq \perp$. If no such set exists, then the node is not satisfied and the function returns \perp . Then, the decryption is presented as follows:

$$\begin{aligned}
 F_y &= \left(\prod_{z \in S_y} F_z \right)^{\Delta_{i, S_y}(0)} \\
 &= (e(g, g)^{\sum_{z \in S_y} rq_z(0) \Delta_{i, S_y}(0)}) \\
 &= (e(g, g)^{\sum_{z \in S_y} rq_{parent(z)}(index(z)) \Delta_{i, S_y}(0)}) \\
 &= (e(g, g)^{r \sum_{z \in S_y} q_y(i) \Delta_{i, S_y}(0)}) \\
 &= e(g, g)^{rq_y(0)}
 \end{aligned} \quad (6.27)$$

Finally, we are able to decrypt the root node by computing $F_R = e(g, g)^{rq_R(0)} = e(g, g)^{rs}$. Then, the ciphertext can be decrypted by computing

$$M = \frac{\tilde{E}}{\frac{e(E, d)}{F_R}} = \frac{Me(g, g)^{\alpha s}}{\frac{e(h^s, g^{\frac{\alpha+r}{r}})}{e(g, g)^{rs}}}. \quad (6.28)$$

6.3.4.2 Security proof

Adversary model. The authors follow the replayable chosen-ciphertext attack (RCCA) security and define RCCA security for their setting. The intuition of security proof in the scheme is illustrated in **Figure 6.9**. The whole game is described in **Figure 6.10**.

- **Setup:** The challenger runs setup algorithm and gives the public key PK to the adversary.
- **Phase 1:** The challenger initializes an empty set D_{key} . Proceedingly, adversary is allowed to make the following queries for several times.
 - *Private key query:* Upon receiving I_{key} , challenger runs key generation algorithm on I_{key} to obtain SK and returns SK after setting $D_{key} = D_{key} \cup \{I_{key}\}$.
 - *Encryption query:* Upon receiving M, j and I_{enc} , challenger runs encryption algorithm totally to obtain CT and $\{OEK_i\}_{i=1}^n$. But only return CT and $\{OEK_i\}_{i=1, i \neq j}^n$ to adversary.
 - *Decryption query:* Upon receiving CT encrypted under I_{enc} , challenger generates SK for I_{key} and performs decryption on CT to obtain M . Finally, return M .

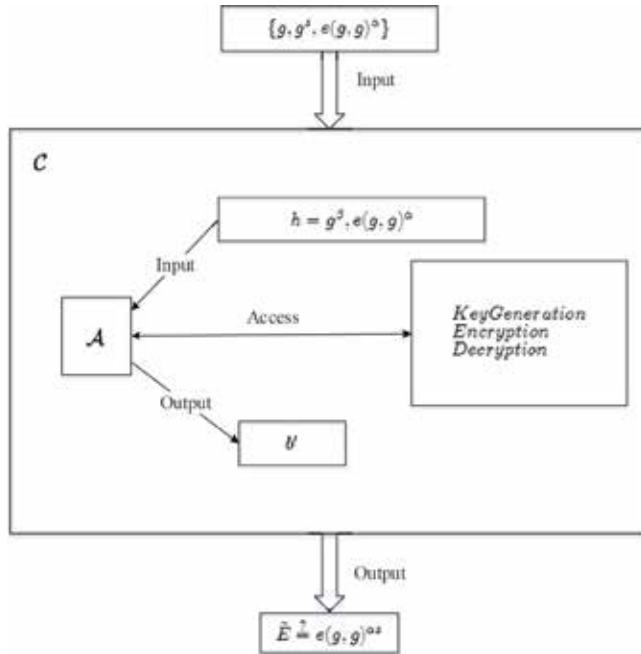


Figure 6.9. Intuition of security proof in the ABE with outsourced encryption scheme.

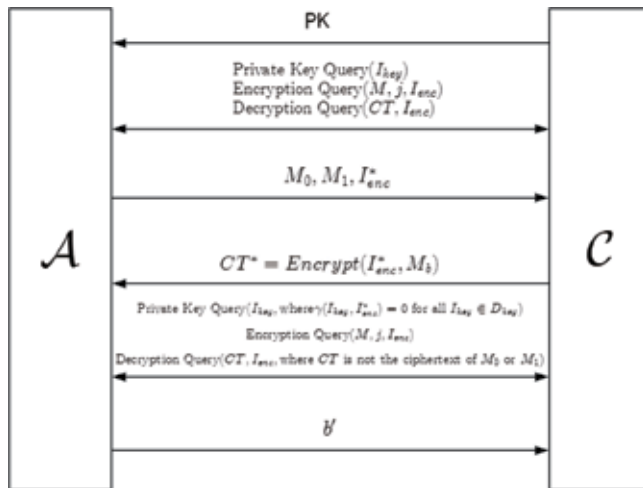


Figure 6.10. Game in the ABE with outsourced encryption scheme.

- **Challenge:** Adversary submits two messages M_0 and M_1 . In addition the adversary gives I_{enc}^* satisfying that $\gamma(I_{key}, I_{enc}^*) = 0$ for all $I_{key} \in D_{key}$. Challenger flips a random coin b and encrypts M_b under I_{enc}^* totally. Finally, return the resulting ciphertext CT^* .
- **Phase 2:** Phase 1 is repeated with the restrictions.
 - Adversary cannot issue private key query on iKey where $\gamma(I_{key}, I_{enc}^*) = 1$.
 - Decryption query will be answered normally except that the response would be either M_0 or M_1 ; then, challenger responds with a special message instead.
- **Guess:** Adversary outputs a guess b' of b .

Definition. A CP-ABE or KP-ABE scheme with outsourced encryption is secure against replayable chosen-ciphertext attack if all polynomial-time adversaries have at most a negligible advantage in the game defined above.

Theorem. The proposed construction is CPA-security model under the assumption that the master node as well as at least one of the slave nodes in MapReduce cloud is “honest but curious” in the generic group model.

Proof. We observe that in the security game shown, the challenge ciphertext has a component \tilde{E} which is randomly either $M_0e(g, g)^{\alpha}$ or $M_1e(g, g)^{\alpha}$. We can instead consider a modified game in which \tilde{E} is either $e(g, g)^{\alpha}$ or $e(g, g)^{\nu}$, where ν is selected uniformly at random from \mathbb{Z}_q , and the adversary must decide which is the case. It is clear that any adversary has advantage ϵ in the original game that can be transformed into an adversary that has advantage at least $\frac{\epsilon}{5}$ in the modified game. Then, we would like to bound the adversary’s advantage in the modified game.

Then, we are to denote $\phi_0 : \mathbb{Z}_q \rightarrow \{0, 1\}^{\log q}$ as a random encoding for the group operation $g^x \in \mathbb{G}$ and $\phi_1 : \mathbb{Z}_q \rightarrow \{0, 1\}^{\log q}$ as a random encoding for $e(g, g)^x \in \mathbb{G}_T$. Let $g = \phi_0(1)$.

At setup time, simulator chooses $\alpha, \beta \in_R \mathbb{F}_q$. Note that if $\beta = 0$ which happens with probability $\frac{1}{q}$, then setup is aborted. Finally, publish public parameters $h = g^\beta$ and $e(g, g)^\alpha$ to adversary.

When the adversary calls for the evaluation of H on any string i , simulator maintains a list \mathcal{L} to store the response to hash query. Upon receiving a string i , if the entry (i, g^{t_i}) exists in \mathcal{L} , straightforwardly return g^{t_i} . Otherwise, pick $t_i \in_R \mathbb{R}\mathbb{F}_q$ and return g^{t_i} after adding the entry (i, g^{t_i}) into \mathcal{T} .

When the adversary makes its j th private key request on ω_j of attributes, challenger picks $r^{(j)} \in_R \mathbb{R}\mathbb{F}_q$ and computes $d = g^{\frac{a+r^{(j)}}{p}}$, and we have $d_{i0} = g^{r^{(j)}+t_i r_i^{(j)}}$ and $d_{i1} = g^{r_i^{(j)}}$ for $i \in \omega_j \cup \{\text{attr}(\theta)\}$ and $r_i^{(j)} \in_R \mathbb{Z}_q$.

When the adversary makes encryption request on M, j as well as \mathcal{T}_U , the simulator chooses $s \in_R \mathbb{R}\mathbb{F}_q$. Then, it splits s into s_1 and s_2 with linear secret sharing. (i) For s_1 , the simulator continues to split it into s_{11}, \dots, s_{1n} and uses the linear secret-sharing scheme associated with \mathcal{T}_U to construct shares $\lambda_i^{(j)}$ of s_{1j} ($j = 1, \dots, n$) for all relevant attributes i . Then, the simulator makes a reduction by computing $E_{i0} = g^{\sum_{j=1}^k \lambda_i^{(j)}}$ and $E_{i1} = g^{t_i \sum_{j=1}^k \lambda_i^{(j)}}$ for each relevant attribute i . (ii) For s_2 , the simulation performs computation to obtain $E_{\theta 0} = g^{s^2}$ and $E_{\theta 1} = g^{t_{\theta} s^2}$. Finally, these values along with $\tilde{E} = Me(g, g)^{as}$, $E = g^s$ and $\{s_{1i}\}_{i=1, i \neq j}^n$ are sent to adversary.

When adversary asks for a challenge on M_0, M_1 , and \mathcal{T}^* , simulator's action is identical to the response to encryption query, except that it picks $u \in_R \mathbb{Z}_q$ and constructs the encryption as follows: $\tilde{E} = e(g, g)^u$ and $E = h^s$. For each relevant attribute i , $E_{i0} = g^{\lambda_i}$ and $E_{i1} = g^{t_i \lambda_i}$.

Therefore, using the generic bilinear group model, it is able to be shown that with probability $1 - O(\frac{p}{q})$ taken over the randomness of the choice of variable values in the simulation, adversary's view in this simulation is identically distributed to what its view would have been if it had been given $\tilde{E} = Me(g, g)^{as}$, where p is the bound on the total number of group elements received from queries to hash functions, group \mathbb{G}, \mathbb{G}_T , and the bilinear map e and from its interaction with security game. Therefore, the proposed construction is secure in the proposed model.

In the construction, the reduced operation is run by the master node which is honest. Moreover, a further split on s_1 is performed to "map" the "partial encryption" task onto n slave nodes to allow for concurrent execution. Since at least one of the slave nodes is honest, they are not able to recover s_1 to fake access policy even if $n - 1$ slave nodes collude.

Finally, we specify that though the proposed construction is secure against chosen-plaintext attack, it is allowed to be extended to the stronger RCCA-security guarantee by using simulation-sound NIZK proofs [173]. Alternatively, if we are willing to use random oracle, then we can use standard techniques such as the Fujisaki-Okamoto transformation.

6.3.5 The ABE with updated policy

The algorithms include AttributeAuthoritySetUp (AASetUp), UserKeyGen, Root Decryption Key Generation (RDKGen), Encryption (ENC), and Decryption (DEC). Basically, bilinear map is a major construct in our system setup and user key generation protocol. In our scheme, the proposed cryptographic algorithms are developed based on the extension of the original CP-ABE by introducing more system parameters based on the role-based access control management and dual encryption (symmetric encryption and ABE). In practice, AttributeAuthoritySetUp (AASetUp), UserKeyGen, and RDKGen are run by the AA, while ENC and DEC are executed by data owners or users.

6.3.5.1 Concrete construction

- **AASetup:** Each AA_k ($k \in$ set of all authority S_A). Let S be a set of attributes issued and managed by the authority AA_k . The AA setup (AA_k) chooses two random numbers $\alpha, \beta \in \mathbb{Z}_p$. Then, the public key (PK_k) = $\mathbb{G}_1, g, h = g\beta_k, f = g^{1/\beta_k}, e(g, g)^{\alpha_k}$, and the secret key SK_k is (β_k, g^{α_k}) .
- **UserKeyGen**($S_{uid, k}, SK_k, Cert_{uid}$) \rightarrow $EDK_{uid, k}, RDK_{oid}$. The UserKeyGen algorithm consists of the UDKGen and EDKGen algorithms to produce the $UDK_{uid, k}$ and $EDK_{uid, k}$, respectively.
 - UDKGen takes as input a set of attributes $S_{uid, k}$, attribute authority's secret key SK_k , and user's certificate $Cert_{uid}$, and then it returns a set of user decryption keys $UDK_{uid, aid}$. The algorithm first chooses a random $r \in \mathbb{Z}_p$ and then random $r_i \in \mathbb{Z}_p$ for each attribute $i \in S_{uid, k}$. Then, the key is computed as $UDK_{uid, k} = (D = g^{\frac{\alpha_k + r}{\beta_k}}, A_i \in S : D_i = g^r \cdot H(i)^{r_i}, D'_t = g^{r_i})$.
 - EDKGen then encrypts the $UDK_{uid, k}$ with the public key of the user $Cert_{uid}$ and outputs an encrypted decryption key $EDK_{uid, k}$. The encryption function is expressed as $UDK_{uid, k} \rightarrow ENC_{RSA}(Cert_{uid}, UDK_{uid, aid}) \equiv EDK_{uid, k}$.

The derived $EDK_{uid, k}$ is then stored in the cloud, and it can be requested anytime by the legitimate user who has the matched GSK_{uid} .

- **RDKGen**($SK_k, oid, Owner'sDS$) \rightarrow RDK_{oid} . RDKGen takes as inputs an attribute authority's secret key SK_k , an identity attribute of data owner oid , and a data owner's digital signature ($Owner'sDS$); then, it produces a root decryption key RDK_{oid} for further use in re-encryption key generation process.
- **ENC**($PK_k, SS, M, ACP, Cert_{uid}$) \rightarrow SCT . Before the file is encrypted, it is compressed into a gzip format. Then, the encryption algorithm is applied over the compressed file with two consecutive steps as follows:
 - Encrypt message M :

$$M \rightarrow ENC_{C-CP-ARBE}(PK_{aid}, ACP, M) \equiv CT. \quad (6.29)$$

The algorithm takes as inputs authority public key PK_k , access control policy ACP , and data M . Then, it returns a ciphertext CT .

- Encrypt ciphertext CT and SS :

$$\begin{aligned} CT &\rightarrow ENC_{AES}(SS, CT) \equiv SCT \\ SS &\rightarrow ENC_{RSA}(Cert_{uid}, SS) \equiv ESS. \end{aligned} \quad (6.30)$$

Then, the algorithm takes group role parameter GRP as a key together with AES algorithm to generate the session key referred to as a secret seal SS to encrypt the ciphertext CT . Then, the algorithm returns sealed ciphertext SCT .

Finally, the SS is encrypted with user's public key $Cert_{uid}$, the algorithm will return encrypted $SS(ESS)$, and it is stored in a cloud server.

- $\text{DEC}(PK_k, SCT, GSK_{uid}, EDK_{uid,k}) \rightarrow M$. The decryption algorithm performs two consecutive steps as follows:
 - Decrypt SS and SCT:

$$\begin{aligned} SS &= \text{DEC}_{\text{RSA}}(GSK_{uid}, \text{ENC}_{\text{RSA}}(\text{Cert}_{uid}, SS)) \\ CT &= \text{DEC}_{\text{AES}}(SS, SCT). \end{aligned} \quad (6.31)$$

The algorithm takes user's global secret key GSK_{uid} , and it returns the session key SS. Then, the algorithm takes SS to decrypt SCT and gets the CT.

- Decrypt CT

The user (uid) requests a cloud $EDK_{uid,k}$ for decryption. The decryption step is conducted as follows:

$$\begin{aligned} UDK_{uid,k} &= \text{DEC}(GSK_{uid}, EDK_{uid,k}) \\ M &= \text{DEC}_{\text{C-CP-ARBE}}(UDK_{uid,k}, CT) \equiv \text{DEC}(UDK_{uid,k}, \text{ENC}(PK_k, ACP, M)). \end{aligned} \quad (6.32)$$

The algorithm takes user's global secret key GSK_{uid} to decrypt the encrypted user decryption key $EDK_{uid,k}$. Then, the algorithm returns $UDK_{uid,k}$. Finally, the $UDK_{uid,k}$ is used to decrypt the ciphertext CT. Together with the PK_k , if the set of attribute S satisfies the ACP structure, the algorithm returns the compressed message M. Finally, the algorithm decompresses M and returns the original M.

6.3.5.2 Security model

Our proposed PRE process embedded in the C-CP-ARBE is secure under the random oracle model as shown in the following security game. The whole game is described in **Figure 6.11**.

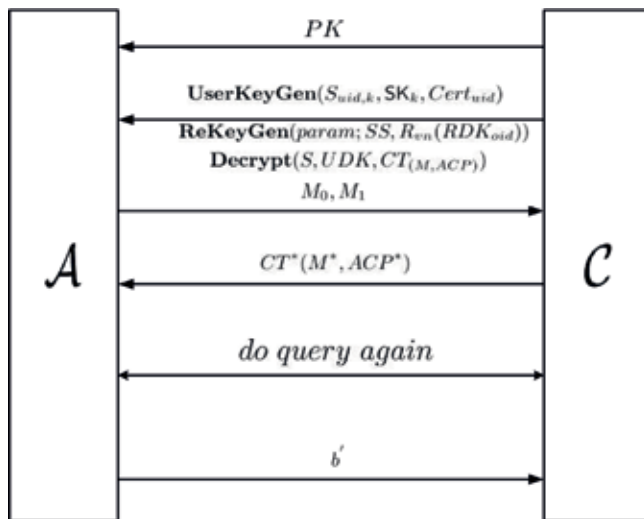


Figure 6.11. Game in the ABE with updated policy scheme.

- **Init:** Adversary \mathcal{A} outputs a challenge access policy ACP^C to challenger \mathcal{C} .
- **Setup:** \mathcal{C} runs the CreateAttributeAuthority algorithm and gives a public keys PK to the adversary \mathcal{A} . For corrupted authorities S'_A , the challenger sends both the public keys and secret keys to adversary.
- **Query Phase 1:**
 - *Private key extraction* : \mathcal{C} runs UserKeyGen on the attribute set $S(S_{uid,k})$ of the corrupted AA and returns UDK to \mathcal{A} .
 - *Re-encryption key extraction oracle* $O_{rk}(S, ACP^C)$: With attribute set S , and an access control policy ACP^C , \mathcal{C} returns $reKeyGen(param; SS, R_{vn}(RDK_{oid}), (ACP^C)) \rightarrow rk_{s,2,vn \rightarrow (M', ACP')}$ to \mathcal{A} , where $rk_{s,2,vn}$ is a generated re-encryption key and $(S, SK_{aid}) \rightarrow UDK$.
 - *RE-encryption oracle* $O_{rk}(S, ACP^C, CT_{(M, ACP)})$: With the input an attribute set S , an access control policy ACP^C , and an original ciphertext $CT_{(M, ACP)}$, \mathcal{C} returns $rk_{s,2 \rightarrow (M', ACP')}$, $CT_{(M, ACP)} \rightarrow CT_{(M', ACP')}^R$, where $reKeyGen(param, SS, R_{vn}(RDK_{oid}), (ACP_{v_n}^{R_{vn}})) \rightarrow rk_{s,2 \rightarrow (M', ACP')}$, $(S, SK_k) \rightarrow UDK$, and $S \models ACP$.
 - *Original ciphertext decryption oracle* $O_{d1}(S, CT_{(M, ACP)})$: With the input an attribute set S and an original ciphertext $CT_{(M, ACP)}$, \mathcal{C} returns Decrypt $(S, UDK, CT_{(M, ACP)}) \rightarrow M$ to \mathcal{A} , where $(S, SK_k) \rightarrow UDK$ and $S \models ACP$.
 - *Re-encrypted ciphertext decryption oracle* $O_{d2}(S', CT_{(M', ACP')}^R)$: With the input an attribute set S' and a re-encrypted ciphertext $CT_{(M', ACP')}^R$, \mathcal{C} returns Decrypt $(S', UDK', CT_{(M', ACP')}^R) \rightarrow M'$, where $(S', SK_k) \rightarrow UDK'$ and $S' \models ACP'$.

Note that if the ciphertexts queried to oracles O_{rk} , O_{d2} , and O_{d1} are invalid, \mathcal{C} simply outputs a \perp .

- *Challenge.* \mathcal{A} outputs two equal-length messages M_0 and M_1 to \mathcal{C} . \mathcal{C} returns $CT^*(M^*, ACP^*) = ENC_{C-CP-ARBE}(PK_k, ACP^*, M_b)$ to \mathcal{A} , where $b \in \{0, 1\}$. *Query Phase 2.* \mathcal{A} performs as it did in Phase 1.
- *Guess.* \mathcal{A} submits a guess bit $b' \in \{0, 1\}$. If $b' = b$, \mathcal{A} wins. The advantage of \mathcal{A} in this game is defined as $Pr[b' = b | \mu = 0] = \frac{1}{2}$.

In the security point of view of VL-PRE, we generate random as a key used with AES algorithm to encrypt re-encryption key component, while our core access control enforcement is based on CP-ABE. The detailed security proof is as presented in the original CP-ABE [172].

6.3.6 The attribute-based proxy re-encryption

The description of our new CP-ABPRE scheme with adaptive CCA security is as follows. Unless stated otherwise, we let \mathcal{U} be an attribute universe and S be an attribute set, $S \subseteq \mathcal{U}$.

6.3.6.1 Concrete construction

- Setup**($1^k, \mathcal{U}$): The setup algorithm runs $(N, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^k)$, where $N = p_1 p_2 p_3$ is the order of group \mathbb{G} and p_1, p_2, p_3 are distinct primes. Let \mathbb{G}_{p_i} denote the subgroup of order p_i in group \mathbb{G} . It chooses $a, \alpha, \kappa, \beta, \varepsilon \in_R \mathbb{Z}_N, g, \hat{g}_1 \in_R \mathbb{G}_{p_1}$, two TCR hash functions $TCR_1 : \mathbb{G}_T \rightarrow \mathbb{Z}_N, TCR_2 : \mathbb{G}_T \rightarrow \{0, 1\}^{\text{poly}(1^k)}$, a CCA-secure one-time symmetric encryption system SYM , and a strongly existential unforgeable one-time signature system OTS . For each attribute $i \in \mathcal{U}$, it chooses $h_i \in_R \mathbb{Z}_N$. The public parameters param are $(N, g, \hat{g}_1, g^a, g^\kappa, g^\beta, g^\varepsilon, e(g, g)^\alpha, \forall i \in \mathcal{U} H_i = g^{h_i}, TCR_1, TCR_2, SYM, OTS)$, and the master secret key msk is (g^a, g^3) , where g_3 is a generator of \mathbb{G}_{p_3} .
- KeyGen**(param, msk, S): The key generation algorithm chooses $t, u \in_R \mathbb{Z}_N$, and $R, R', R'', \{R_i\}_{i \in S} \in_R \mathbb{G}_{p_3}$. The secret key sk_S is $(S, K = g^a g^{at} g^{\kappa u} R, K' = g^u R', K'' = g^t R'', \forall i \in S K_i = H_i R_i)$.
- Encrypt**(param, $(\mathbb{A}, \rho), M$): Given an LSSS access structure (\mathbb{A}, ρ) and a message $M \in \mathbb{G}_T$ in which \mathbb{A} is an $l \times n$ matrix and ρ is a map from each row \mathbb{A}_j to an attribute $\rho(j)$, the encryption algorithm works as follows.
 - Choose a random vector $v = (s, v_2, \dots, v_n) \in_R \mathbb{Z}_N^n$.
 - For each row \mathbb{A}_j , choose $r_j \in_R \mathbb{Z}_N$, run $(ssk, svk) \leftarrow OTS.KeyGen(1^k)$, and set $B_0 = M \cdot e(g, g)^\alpha, B_1 = g^\varepsilon, B_2 = (g^\kappa)^s, B_3 = (\hat{g}_1^{svk} g^\beta)^s, B_4 = (g^\varepsilon)^s, \forall j \in [1, l] \leftarrow (C_j = (g^a)^{\mathbb{A}_{ij}} H_{\rho(j)}^{-r_j}, D_j = g^{r_j}), E = \text{Sign}(ssk, (B_0, B_1, B_3, \forall j \in [1, l](C_j, D_j)))$.
 - Output the original ciphertext $C = (svk, B_0, B_1, B_2, B_3, B_4, \forall j \in [1, l](C_j, D_j), E)$. Note that $\{\rho(j) | 1 \leq j \leq l\}$ are the attributes used in (\mathbb{A}, ρ) .
- ReKeyGen**(param, $sk_S, (\mathbb{A}', \rho')$): Given a secret key $sk_S = (S, K, K', K'', \forall i \in S K_i)$ and an LSSS access structure (\mathbb{A}', ρ') , the re-encryption key $rk_S \rightarrow (\mathbb{A}', \rho')$ is generated as follows.
 - Choose $\theta_1, \theta_2, \theta_3 \in_R \mathbb{Z}_N, \delta \in_R \mathbb{G}_T$, and set $rk_1 = (K g^{\theta_1} g^{a\theta_2})^{TCR_1(\delta)} g^{\theta_3}, rk_2 = (K' g^{\theta_1})^{TCR_1(\delta)}, rk_3 = (K'' g^{\theta_2})^{TCR_1(\delta)}, rk_4 = g^{\theta_3}, \forall i \in S rk_{5,i} = (K_i H_i^{\theta_2})^{TCR_1(\delta)}$.
 - Choose a random vector $v^{(rk)} = (s^{(rk)}, v_2^{(rk)}, \dots, v_n^{(rk)}) \in_R \mathbb{Z}_N^n$. For each row \mathbb{A}'_j of \mathbb{A}' , choose $r_j^{(rk)} \in_R \mathbb{Z}_N$, run $(ssk^{(rk)}, svk^{(rk)}) \leftarrow OTS.KeyGen(1^k)$, and set rk_6 as $svk^{(rk)}, B_0^{(rk)} = e(g, g)^{\alpha(rk)}, B_1^{(rk)} = g^{\varepsilon(rk)}, B_2^{(rk)} = (g^\kappa)^{s^{(rk)}}, B_3^{(rk)} = (\hat{g}_1^{svk^{(rk)}} g^\beta)^{s^{(rk)}}, \forall j \in [1, l] \leftarrow (C_j^{(rk)} = (g^a)^{\mathbb{A}'_{ij}} H_{\rho'(j)}^{-r_j^{(rk)}}, D_j^{(rk)} = g^{r_j^{(rk)}}, E^{(rk)} = \text{Sign}(ssk^{(rk)}, (B_0^{(rk)}, B_1^{(rk)}, B_3^{(rk)}, \forall j \in [1, l](C_j^{(rk)}, D_j^{(rk)})))$.
 - Output the re-encryption key $rk_{S \rightarrow (\mathbb{A}', \rho')} = (rk_1, rk_2, rk_3, rk_4, \forall i \in S rk_{5,i}, rk_6)$.
- ReEnc**(param, $rk_{S \rightarrow (\mathbb{A}', \rho')}, C$): Parse the original ciphertext C under (\mathbb{A}, ρ) as $(svk, B_0, B_1, B_2, B_3, B_4, \forall j \in [1, l](C_j, D_j), E)$ and the re-encryption key $rk_{S \rightarrow (\mathbb{A}', \rho')}$ as $(rk_1, rk_2, rk_3, rk_4, \forall i \in S rk_{5,i}, rk_6)$.

- Check the validity of the original ciphertext C as $e(B_1, g^k) \stackrel{?}{=} e(B_2, g)$,
 $e(B_1, \hat{g}_1^{svk} g^\beta) \stackrel{?}{=} e(B_3, g)$, $e(B_1, g^e) \stackrel{?}{=} e(B_4, g)$, $e(\prod_{\rho(j) \in S} C_j^{w_j}, g) \stackrel{?}{=} e(B_1, g^a) \times \leftarrow$
 $\prod_{\rho(j) \in S} (e(D_j^{-1}, H_{\rho(j)}^{w_j}))$, $S \stackrel{?}{=} (\mathbb{A}, \rho)$
 $Verify(svk, (E, (B_0, B_1, B_3, \forall j \in [1, l](C_j, D_j)))) \stackrel{?}{=} \mathbb{A}$

where w_j are the constants chosen by the proxy such that the following holds $\sum_{\rho(j) \in S} w_j \mathbb{A}_j = (1, 0, \dots, 0)$. If Eq. (1) does not hold, output \perp . Otherwise, proceed.

- Compute $F = \leftarrow \frac{e(B_1, rk_1) e(B_2, rk_2)^{-1} e(B_4, rk_4)^{-1}}{(\prod_{\rho(j) \in S} (e(C_j, rk_3) e(D_j, rk_{s_j}))^{w_j})}$, and further run $\sigma_1 \leftarrow \text{SYM}$.
 $Enc(TCR_2(key), G)$, where $G = (C || rk_6 || F)$ and key $\in \mathbb{R} \mathbb{G}_T$.
- Choose a random vector $v^{(re)} = (s^{(re)}, v_2^{(re)}, \dots, v_n^{(re)}) \in \mathbb{R} \mathbb{Z}_N$. For each row of $\mathbb{A}' \leftarrow$
choose $r_j^{(re)} \in \mathbb{R} \mathbb{Z}_N$, and run $(ssk^{(re)}, svk^{(re)}) \leftarrow OTS.KeyGen(1^k)$ and set σ_2 as
 $svk^{(re)}$, $B_0^{(re)} \leftarrow key \cdot e(g, g)^{as^{(re)}}$, $B_1^{(re)} = g^{s^{(re)}}$, $B_2^{(re)} = (g^k)^{s^{(re)}}$, $B_3^{(re)} = (\hat{g}_1^{svk^{(re)}} g^\beta)^{s^{(re)}}$,
 $\forall j \in [1, l](C_j^{(re)} = (g^a)^{\mathbb{A}'_{j, \rho(j)} v^{(re)}} H_{\rho'(j)}^{-r_j^{(re)}})$, $D_j^{(re)} = g^{r_j^{(re)}}$, $E^{(re)} = \text{Sign}(ssk^{(re)}$,
 $(B_0^{(re)}, B_1^{(re)}, B_3^{(re)}, \forall j \in [1, l](C_j^{(re)}, D_j^{(re)}))$.
- Output the re-encrypted ciphertext $C_R = (\sigma_1, \sigma_2)$ under (\mathbb{A}', ρ') .
- **Dec**(param, sk_S, C): Parse the original ciphertext C under (\mathbb{A}, ρ) as $(svk, B_0,$
 $B_1, B_2, B_3, B_4, \forall j \in [1, l](C_j, D_j), E)$ and the secret key sk_S as $(S, K, K', K'',$
 $\forall i \in S K_i)$. The decryption algorithm chooses a set of constants $w_j \in \mathbb{R} \mathbb{Z}_N$
such that $\sum_{\rho(j) \in S} w_j \mathbb{A}_j = (1, 0, \dots, 0)$, and next recovers the message as
follows.
- If Eq. (1) does not hold, output \perp . Otherwise, proceed.
- Compute $e(B_1, K) e(B_2, K')^{-1} / (\prod_{\rho(j) \in S} (e(C_j, K'') e(D_j, K_{\rho(j)}))^{w_j}) = e(g, g)^{as}$, and
output the message $M = B_0 / e(g, g)^{as}$.

- **Dec_R**(param, sk_S, C_R): Parse the re-encrypted ciphertext C_R under (\mathbb{A}', ρ') as
 (σ_1, σ_2) and the secret key sk_S as $(S, K, K', K'', \forall i \in S K_i)$.
- Check the validity of σ_2 as $e(B_1^{(re)}, g^k) \stackrel{?}{=} e(B_2^{(re)}, g)$, $e(B_1^{(re)}, \hat{g}_1^{svk^{(re)}} g^\beta) \stackrel{?}{=} e(B_3^{(re)}, g)$, $S \stackrel{?}{=} (\mathbb{A}', \rho')$, $e(\prod_{\rho(j) \in S} (C_j^{(re)})^{w_j^{(re)}}, g) \stackrel{?}{=} e(B_1^{(re)}, g^a) \times \leftarrow$
 $\prod_{\rho'(j) \in S} (e((D_j^{(re)})^{-1}, H_{\rho'(j)}^{w_j^{(re)}}))$
 $Verify(svk^{(re)}, (E^{(re)}, (B_0^{(re)}, B_1^{(re)}, B_3^{(re)}, \forall j \in [1, l](C_j^{(re)}, D_j^{(re)}))) \stackrel{?}{=} \mathbb{A}$

where $w_j^{(re)}$ are the constants chosen by the decryptor such that $\sum_{\rho'(j) \in S} w_j^{(re)} \mathbb{A}'_j = (1, 0, \dots, 0)$. If Eq. (2) does not hold, output \perp . Otherwise, proceed.

- Compute $e(B_1^{(re)}, K)e(B_2^{(re)}, K')^{-1} / (\prod_{\rho'(j) \in S} (e(C_j^{(re)}, K'')e(D_j^{(re)}, K_{\rho'(j)}))^{w_j^{(re)}}) = \mathfrak{e}(g, g)^{\alpha^{(re)}}$, and output $key = \mathfrak{B}_0 / \mathfrak{e}(g, g)^{\alpha^{(re)}}$.
- Run $G = \mathfrak{SYM}.Dec(TCR_2(key), \sigma_1)$.
- Parse G as (C, rk_ϵ, F) . If either Eq. (1) or the following verification for rk_ϵ does not hold, output \perp . Otherwise, proceed.

$$e(B_1^{(rk)}, g^k) \stackrel{?}{=} \mathfrak{e}(B_2^{(rk)}, g), e(B_1^{(rk)}, g_1^{svk^{(rk)-\beta}} g^\beta) \stackrel{?}{=} \mathfrak{e}(B_3^{(rk)}, g),$$

$$e(\prod_{\rho'(j) \in S} (C_j^{(rk)})^{w_j^{(rk)-}} g^{a_j}) \stackrel{?}{=} \mathfrak{e}(B_1^{(rk)}, g^a) \cdot \prod_{\rho'(j) \in S} (e((D_j^{(rk)})^{-1}, H_{\rho'(j)}^{w_j^{(rk)}}))$$

$$Verify(sv k^{(rk)}, (E^{(rk)}, (B_0^{(rk)}, B_1^{(rk)}, B_3^{(rk)}, \forall j \in [1, l](C_j^{(rk)}, D_j^{(rk)}))) \stackrel{?}{=} \mathfrak{A}_j \stackrel{?}{=} (A', \rho'),$$

where $w_j^{(rk)}$ are the constants chosen by the decryptor such that $\sum_{\rho'(j) \in S} w_j^{(rk)} \mathfrak{A}_j = (1, 0, \dots, 0)$.

- Compute $e(B_1^{(rk)}, K)e(B_2^{(rk)}, K')^{-1} / (\prod_{\rho'(j) \in S} (e(C_j^{(rk)}, K'')e(D_j^{(rk)}, K_{\rho'(j)}))^{w_j^{(rk)}}) = \mathfrak{e}(g, g)^{\alpha^{(rk)}}$ and then $B_0^{(rk)} / \mathfrak{e}(g, g)^{\alpha^{(rk)}} = \mathfrak{e}$. Compute $F^{TCR_1(\delta)^{-1}} = \mathfrak{e}(g, g)^\alpha$, and finally output $M = \mathfrak{B}_0 / \mathfrak{e}(g, g)^\alpha$.

6.3.6.2 Security model

A single-hop unidirectional CP-ABPRE scheme is IND-CCA secure at original ciphertext if no probabilistic polynomial-time (PPT) adversary \mathcal{A} can win the game below with non-negligible advantage. In the game, \mathcal{C} is the challenger, and k and \mathcal{U} are the security parameter and attribute universe. The intuition of security proof in the scheme is illustrated in **Figure 6.12**. The whole game is described in **Figure 6.13**.

- **Setup.** \mathcal{C} runs $Setup(1^k, \mathcal{U})$ and sends param to \mathcal{A} .
- **Phase 1.**
 - *Secret key extraction oracle* $\mathcal{O}_{sk}(S)$: On input an attribute set S , \mathcal{C} runs $sk_S = KeyGen(param, msk, S)$ and returns sk_S to \mathcal{A} .
 - *Re-encryption key extraction oracle* $\mathcal{O}_{rk}(S, (\mathbb{A}, \rho'))$: On input S , and an access structure (\mathbb{A}, ρ') , \mathcal{C} returns $rk_{S \rightarrow (\mathbb{A}, \rho')} = ReKeyGen(param, sk_S, (\mathbb{A}, \rho')) \in \mathfrak{R}$, where $sk_S = KeyGen(param, msk, S)$.
 - *Re-encryption oracle* $\mathcal{O}_{re}(S, (\mathbb{A}, \rho'), C)$: On input $S, (\mathbb{A}, \rho')$ and an original ciphertext C under (\mathbb{A}, ρ) , \mathcal{C} returns $C_R = ReEnc(param, rk_{S \rightarrow (\mathbb{A}, \rho')}, C)$ to \mathcal{A} , where $rk_{S \rightarrow (\mathbb{A}, \rho')} = ReKeyGen(param, sk_S, (\mathbb{A}, \rho')), sk_S = KeyGen(param, msk, S)$ and $S = (\mathbb{A}, \rho)$.

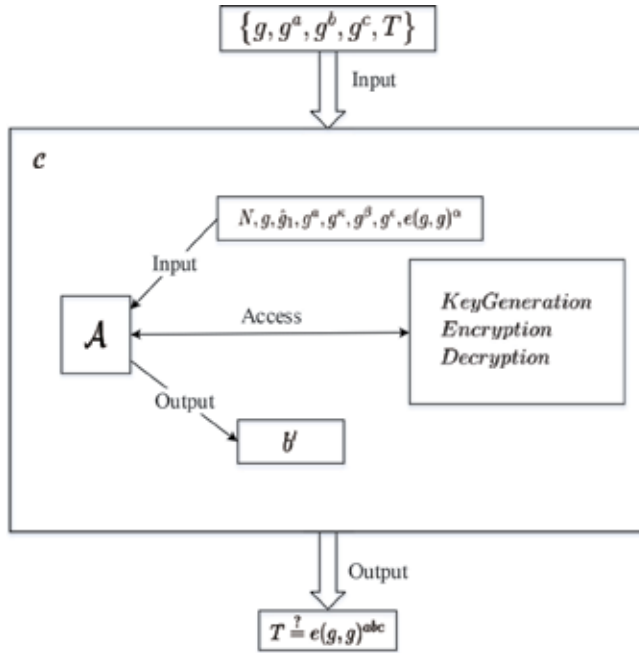


Figure 6.12.
 Intuition of security proof in the attribute-based proxy re-encryption.

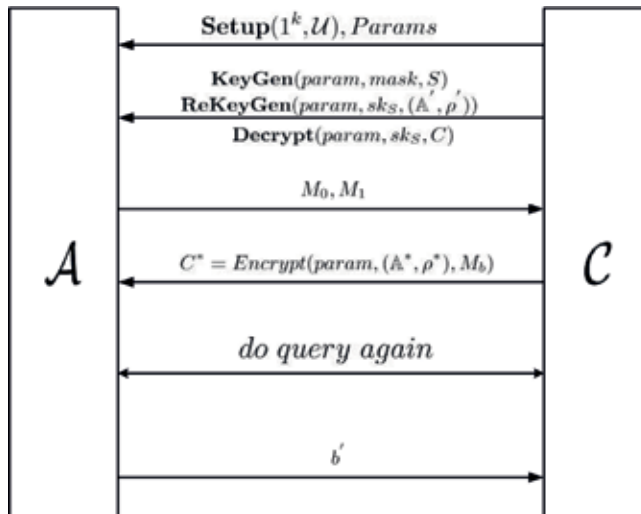


Figure 6.13.
 Game in the attribute-based proxy re-encryption scheme.

- *Original ciphertext decryption oracle* $\mathcal{O}_{dec}(S, C)$: On input S and an original ciphertext C under (\mathbb{A}, ρ) , \mathcal{C} returns $M = Dec(param, sk_S, C) \in \mathcal{A}$, where $sk_S = KeyGen(param, msk, S)$ and $S = (\mathbb{A}, \rho)$.
- *Re-encrypted ciphertext decryption oracle* $\mathcal{O}_{dec}(S, C_R)$: On input S and a re-encrypted ciphertext C_R under (\mathbb{A}, ρ) , \mathcal{C} returns $M = Dec_R(param, sk_S, C_R)$, where $sk_S = KeyGen(param, msk, S)$ and $S = (\mathbb{A}, \rho)$.

If the ciphertexts issued to \mathcal{O}_{re} , \mathcal{O}_{dec} , and \mathcal{O}_{dec_R} are invalid, \mathcal{C} simply outputs \perp .

- **Challenge.** \mathcal{A} outputs two equal-length messages M_0 and M_1 and a challenge access structure $(\mathbb{A}^*, \rho^*) \in \mathcal{C}$. If the following queries

$\mathcal{O}_{sk}(S)$ for any $S \in (\mathbb{A}^*, \rho^*)$, $\mathcal{O}_{rk}(S, (\mathbb{A}', \rho'))$ for any $S \in (\mathbb{A}^*, \rho^*)$, and

$\mathcal{O}_{sk}(S')$ for any $S' \in (\mathbb{A}', \rho')$

are never made, \mathcal{C} returns $C^* = \text{Encrypt}(param, (\mathbb{A}^*, \rho^*), M_b) \in \mathcal{A}$, where $b \in_R \{0, 1\}$.

- **Phase 2.** \mathcal{A} continues making queries as in Phase 1 except the following:
 - $\mathcal{O}_{sk}(S)$ for any $S \in (\mathbb{A}^*, \rho^*)$.
 - $\mathcal{O}_{rk}(S, (\mathbb{A}', \rho'))$ for any $S \in (\mathbb{A}^*, \rho^*)$ and $\mathcal{O}_{sk}(S')$ for any $S' \in (\mathbb{A}', \rho')$.
 - $\mathcal{O}_{re}(S, (\mathbb{A}', \rho'), C^*)$ for any $S \in (\mathbb{A}^*, \rho^*)$ and $\mathcal{O}_{sk}(S')$ for any $S' \in (\mathbb{A}', \rho')$.
 - $\mathcal{O}_{dec}(S, C^*)$ for any $S \in (\mathbb{A}^*, \rho^*)$.
 - $\mathcal{O}_{dec_R}(S, C_R)$ for any C_R under (\mathbb{A}, ρ) , $S \in (\mathbb{A}, \rho)$ where C_R is a derivative of C^* . As of [3], the derivative of C^* is defined as follows.
 - C^* is a derivative of itself.
 - If \mathcal{A} has issued a re-encryption key query on $(S, (\mathbb{A}', \rho')) \in \mathcal{C}$ to get $rk_{S \rightarrow (\mathbb{A}', \rho')}$ and obtained $C_R = \text{ReEnc}(param, rk_{S \rightarrow (\mathbb{A}', \rho')}, C^*)$ such that $\text{Dec}_R(param, sk_{S'}(C_R)) \in \{M_0, M_1\}$, then C_R is a derivative of C^* , where $S' \in (\mathbb{A}^*, \rho^*)$ and $S' \in (\mathbb{A}', \rho')$.
 - If \mathcal{A} has issued a re-encryption query on $(S, (\mathbb{A}', \rho'), C^*)$ and obtained the re-encrypted ciphertext C_R , then C_R is a derivative of C^* , where $S \in (\mathbb{A}^*, \rho^*)$.
- **Guess.** \mathcal{A} outputs a guess bit $b' \in \{0, 1\}$. If $b' = b$, \mathcal{A} wins.

The advantage of \mathcal{A} is defined as $Adv_{CP-ABPRE, \mathcal{A}}^{IND-CCA-Or}(1^k, \mathcal{U}) = |\Pr[b' = b] - \frac{1}{2}|$.

6.4. Comparison and observations

The first KP-ABE scheme that allows any monotone access structures was proposed by Goyal et al. [171]. In their cryptosystem, ciphertexts are labeled with sets of attributes, and private keys are associated with access structures that control which ciphertexts a user is able to decrypt. They demonstrated the applicability of our construction to sharing of audit-log information and broadcast encryption. Their construction supports delegation of private keys which subsumes hierarchical identity-based encryption (HIBE).

The CP-ABE scheme was presented by Waters et al. [177]. In that paper they presented a system for realizing complex access control on encrypted data. By using their techniques, encrypted data can be kept confidential even if the storage server is untrusted; moreover, their methods are secure against collusion attacks. Previous

attribute-based encryption systems used attributes to describe the encrypted data and built policies into user's keys, while in their system, attributes are used to describe a user's credentials, and a party encrypting data determines a policy for who can decrypt. Thus, their methods are conceptually closer to traditional access control methods such as role-based access control (RBAC).

Subsequently, there are some other versions of ABE that have been proposed such as the outsourced ABE. In outsourcing the decryption of ABE ciphertexts, they proposed a new paradigm for ABE that largely eliminates the size of the ciphertext, and the time required to decrypt it grows with the complexity of the access formula for users. Suppose that ABE ciphertexts are stored in the cloud. They showed how a user can provide the cloud with a single transformation key that allows the cloud to translate any ABE ciphertext satisfied by that user's attributes into a (constant-size) ElGamal-style ciphertext, without the cloud being able to read any part of the user's messages. In outsourcing encryption of attribute-based encryption with MapReduce, they formulated a novel paradigm of outsourcing encryption of ABE to cloud service provider to relieve local computation burden. They proposed an optimized construction with MapReduce cloud which is secure under the assumption that the master node as well as at least one of the slave nodes is honest. After outsourcing, the computational cost at the user's side during encryption is reduced to approximate four exponentiations, which is constant. Another advantage of the proposed construction is that the user is able to delegate encryption for any policy.

In 2014, Liang et al. [49] proposed a new the ciphertext-policy attribute-based proxy re-encryption (CP-ABPRE) to tackle the problem by integrating the dual-system encryption technology with selective proof technique. Although the new scheme supporting any monotonic access structures is built in the composite-order bilinear group, CCA security in the standard model without jeopardizing the expressiveness of access policy is proven adaptively. They further make an improvement for the scheme to achieve more efficiency in the re-encryption key generation and re-encryption phases.

In 2017, Fugkeaw et al. [192] extended the capability of their access control scheme: collaborative-ciphertext policy-attribute role-based encryption (C-CP-ARBE) to be capable of supporting secure and flexible policy updates in the big data outsourcing environment. They developed a secure policy updating algorithm and proposed a very lightweight proxy re-encryption (VL-PRE) technique to enable the policy updating to be done in the cloud in an efficient and computationally cost-effective manner.

Observations on the ABE with delegated capacity in terms of efficiency and verifiability are described as follows. Let SK denote secret key, and let CT denote ciphertexts. In addition, let ENC denote encryption and DEC denote decryption.

From **Tables 6.2** and **6.3**, we compare the six schemes with respect to the user key, the ciphertext length, the encryption, and the decryption.

	ABE	CP-ABE	CP-ABPRE
SK	$ A_u + LG_1$	$(2 A_u + 1)LG_1$	$3G_1$
CT	$ A_{ct} LG_1 + LG_2$	$(2 A_{CT} + 1)LG_1 + LG_2$	$(2L + 3)G_1 + G_2$
ENC	$ A_{ct} G_1 + G_2$	$(2 A_{CT} + 1)G_1 + 2G_2$	$(2L + 3)G_1 + G_2$
DEC	$dC_e + dG_2$	$2 A_u C_e + (2 m + 2)G_2$	$(2L + 2)G_2$

Table 6.2.
 Comparison 1.

	OD CP-ABE	OD KP-ABE	OE ABE
SK	$Z_p + 5G_1 + G_2$	$Z_p + (3 + 2l)G_1$	$4G_1$
C	$(2L + 1)G_1 + G_2$	$2G_1 + G_2$	$3G_1 + G_2$
ENC	$(2L + 1)G_1 + G_2$	$2G_1 + G_2$	$3G_1 + G_2$
DEC	G_2	G_2	G_2

Table 6.3.
Comparison 2.

The previous sections discuss the research process of ABE which has received substantial achievements. However, there still exist many problems worth further study. According to application requirements and the shortcoming of the existing algorithms, some possible future works remain open.

Proxy Re-Encryption for Secure Access Delegation

7.1. Introduction

The concept of proxy re-encryption (PRE) dates back to the work of Blaze et al. [36]. The goal of such systems is to securely enable the re-encryption of ciphertext from one key to another, without relying on trusted parties. PRE provides a secure and flexible method for a sender to store and share data. A user may encrypt his file with his own public key and then store the ciphertext in an honest-but-curious server. When the receiver is decided, the sender can delegate a re-encryption key associated with the receiver to the server as a proxy. Then the proxy re-encrypts the initial ciphertext to the intended receiver. Finally, the receiver can decrypt the resulting ciphertext with her private key. The security of PRE usually assures that (1) neither the server/proxy nor non-intended receivers can learn any useful information about the (re-)encrypted file, and (2) before receiving the re-encryption key, the proxy cannot re-encrypt the initial ciphertext in a meaningful way.

7.1.1 The motivation of PRE

The PRE has a significant application in the cloud computing environment. In the cloud computing environment, users can rent the storage space of cloud service providers to store data in order to save the cost of equipment purchase and maintenance. For security reasons, users usually encrypt the data before storing it in the cloud storage space. In many applications, users need to share certain data with the specified user, and they want to avoid other people, including the cloud server provider to get the data. One of the most simple ways is that data downloaded to the local users put those lines of cipher decryption, then use to specify the user's public key to encrypt, and then send to the specified user; the latter can decrypt the data. This method, then, requires a lot of communication overhead and computing costs and requires users to increase the local storage space, which is not in line with the original intention of the user to save the cost of equipment through cloud computing. Using the PRE, one can solve the problem more effectively: the user need only the agent for the key to the cloud computing service providers, which can convert encrypted data to specify users against ciphertext; then the specified user that uses its own private key can access the shared data. This can be done: even if the cloud service provider owns the agent key, he cannot know the content of the shared data. Users do not need to download the data to local, so they do not need to purchase additional local storage devices. **Figure 7.1** is a scenario that describes data sharing using the PRE in a cloud computing environment to achieve security.

7.1.2 History of PRE

Proxy re-encryption (PRE) was first formulated by Blaze et al. [36]. They proposed the first PRE scheme based on ElGamal cryptosystem, in which the proxy does not know the underlying messages in the re-encrypted ciphertext. Ateniese et al. [37] remarked that Blaze et al.'s scheme is inherent bidirectional, that is, the re-encryption key allows translating ciphertexts not only from Alice to Bob but also

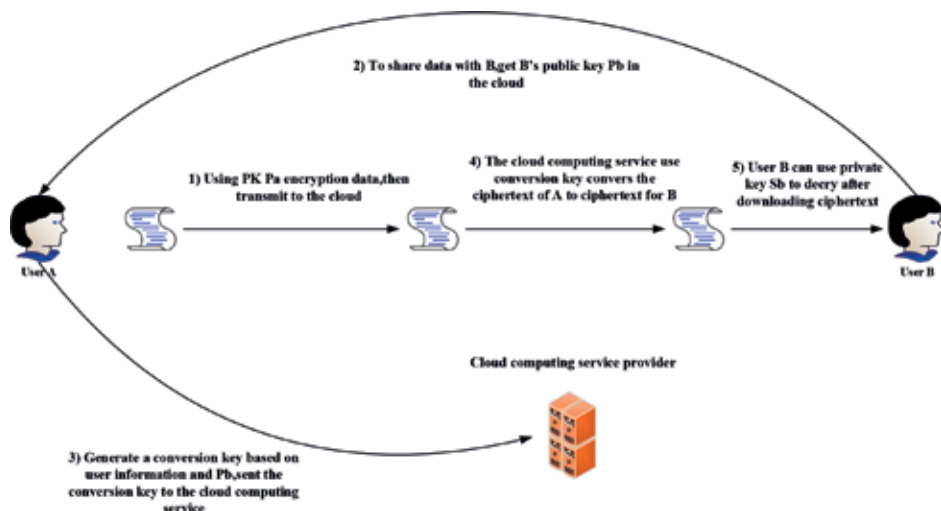


Figure 7.1. Using PRE to achieve secure data sharing in a cloud computing environment.

from Bob to Alice, which would damage the security of Bob's data. Jakobsson [193] proposed a quorum-based protocol which divides the proxy into many isolated components. The delegators' keys will be safe if the number of honest proxies arrives the set quorum. The notion of distributed proxies was also considered in [194].

Many works have been done to achieve unidirectional schemes, where the collusion between delegators and the proxy cannot compromise the delegatee. Ivan and Dodis [195] constructed a unidirectional proxy encryption by splitting a decryption key into two parts, one of which is for the proxy and the other is for Bob. This scheme protects delegatee's security, although the delegatee Bob is required to store additional secrets. The unidirectional PRE scheme presented by Ateniese et al. [37] eliminates the storage need of extra secrets before re-encryption. They also enumerated several properties that are often used to evaluate a PRE system. These desirable properties include unidirectionality, permitting Alice to delegate to Bob without making Alice able to decrypt Bob's ciphertext, noninteractivity, allowing Alice generating re-encryption key offline and multiuse capability and permitting a proxy to perform multiple re-encryptions on a ciphertext; key optimization, not expanding Bob's secret storage, regardless of how many delegations he can accept; and original accessibility, allowing Alice to decrypt the re-encrypted ciphertext that were originally encrypted by her.

It is an open problem left by the work of [37] to construct a PRE scheme secure against chosen-ciphertext attack (CCA). Canetti and Hohenberger [38] proposed a CCA-secure PRE scheme in bilinear groups, although this scheme is bidirectional. Libert and Vergnaud [39] presented the first replayable CCA-secure unidirectional PRE scheme in the standard model. Deng et al. [43] presented a PRE scheme without resorting to bilinear pairings and proved its CCA security in the random oracle model. By using signature of knowledge and Fujisaki Okamoto conversion, Shao and Cao [42] constructed a unidirectional PRE scheme that is proved CCA-secure in the random oracle. Subsequently, Matsuda et al. [41] proposed a CCA-secure PRE scheme in the standard model by applying the reapplicable lossy trap-door function, while Weng et al. [44] specified some security drawbacks of Matsuda et al.'s scheme.

There are many extensions of PRE from different aspects. Green and Ateniese [45] proposed the notion of identity-based proxy re-encryption (IBPRE) by mixing PRE with identity-based cryptography. They constructed an IBPRE scheme that is proved to be CCA-secure in the random oracle. Han et al. [48] proposed an IBPRE storage scheme where both queries from the intra-domain and inter-domain are considered and collusion attacks can be resisted. Moreover, the owner can determine the access permission independently. Chu and Tzeng [46] proposed an IBPRE scheme and proved its CCA security in the standard model. Matsuo [196] proposed a hybrid proxy re-encryption system that can re-encrypt a ciphertext in traditional public-key systems to a ciphertext of identity-based encryption. Due to the limitation of the public-key and identity-based encryption, this scheme does not support the broadcasting mechanism. To reduce security risks on the proxy, Libert and Vergnaud [39] provided a method to trace the malicious proxies in proxy re-encryption systems. Shao et al. [197] proposed an anonymous IBPRE (AIBPRE) to achieve the anonymity of recipients, and Kawai and Takashima [198] proposed a fully anonymous functional PRE scheme. Recently, Deng et al. [199] proposed an asymmetric cross-cryptosystem re-encryption which is similar with our IBPRE2, while their scheme does not achieve the CCA security. Some attribute-based proxy re-encryption schemes [200, 201] have been proposed to enforce more flexible access control on outsourced data. Liang et al. [202] presented a ciphertext-policy attribute-based re-encryption scheme which employs the PRE technology in the attribute-based encryption cryptographic setting and proves its adaptively CCA security in the standard model. Usually, attribute-based cryptosystem requires complicated computations for encryption and decryption, which makes them unaffordable to resource-limited users.

The existing PRE schemes do not provide a re-encryption from a complicated cryptosystem to a simple cryptosystem with higher security (i.e., CCA security), which limits their applications in cloud computing environments. Due to high demand on data mining and analytics activities, some new and developing systems are highly demanded like the Internet of things (IoT) [203] combined with the secret and reliable local storage system as well as the scalable IoT storage systems. The proposed IBPRE2 can transform ciphertext of an IBBE system into ciphertext of an IBE system, without requiring any interaction between delegator and delegatee, nor extra secret storage of the delegatee. Hence, users can directly forward their data decryptable by a set of users to a specific user, without performing decryption nor requiring the specific one to take any extra efforts. Zhou et al. [204] proposed a framework of identity-based proxy re-encryption (IBPRE2) system for cloud computing. Xu et al. [205] built the encrypted cloud email system based on our CIBPRE scheme.

7.1.3 Roadmap

The remainder of this chapter is organized as follows. Section 2 presents the formal definitions and security models of proxy re-encryption (PRE). Design philosophy is given in Section 3, together with concrete construction and security proof of several representative proxy re-encryption schemes. Comparison and observations are discussed in Section 4.

7.2. Framework and security model of PRE

7.2.1 The formal definition of PRE

There are eight algorithms in a PRE model:

- **Setup**(λ). In this algorithm, we can give a security parameter λ and get a parameter (par).
- **Keygen**(par). User i sets his public key and the private key (pk_i, sk_i).
- **ReKeygen**(pk_i, sk_j). The algorithm returns a re-encryption key $rk_{i \rightarrow j}$ from pk_i and sk_j .
- **Enc1**(m, pk_j, par). The algorithm encrypts a message m under the public key pk_j at the first level and returns the ciphertext CT_j .
- **Enc2**(m, pk_i, par). The algorithm encrypts a message m under the public key pk_i at the second level and returns the ciphertext CT_i .
- **ReEnc**($rk_{i \rightarrow j}, CT_i$). The algorithm re-encrypts the second level ciphertext CT_i under the re-encryption key and returns the first-level ciphertext CT_j .
- **Dec1**(CT_j, sk). The algorithm decrypts the first-level ciphertext CT_j under the private-key sk and returns the message m or \perp (when CT_j is an illegal ciphertext).
- **Dec2**(CT_i, sk). The algorithm decrypts the second-level ciphertext CT_i under the private-key sk and returns the message m or \perp (when CT_i is an illegal ciphertext).

7.2.2 The security properties of PRE

In this part, we will discuss the most useful properties of proxy re-encryption protocols:

- **Unidirectional**: Delegation from Alice to Bob does not allow re-encryption from Bob to Alice.

Noninteractive: Re-encryption keys can be generated by Alice using Bob's public key. No trusted third party or interaction is required.

- **Proxy invisibility**: The proxy is transparent in the sense that neither the sender of an encrypted message nor any of the delegates have to be aware of the existence of the proxy.
- **Original access**: Alice can decrypt re-encrypted ciphertext that were originally sent to her. In some applications, it may be desirable to maintain access to her re-encrypted ciphertext.
- **Temporary**: Bob is only able to decrypt messages intended for Alice that were authored during some specific time period i .
- **Non-transitive**: The proxy, alone, cannot re-delegate decryption rights. For example, from $rk_{a \rightarrow b}$ and $rk_{b \rightarrow c}$, he cannot produce $rk_{a \rightarrow c}$.
- **Nontransferable**: The proxy and a set of colluding delegates cannot re-delegate decryption rights. For example, from $rk_{a \rightarrow b}$, sk_b , and pk_c , they cannot produce $rk_{a \rightarrow c}$.

7.2.3 Security model of PRE

Setup. Challenger C runs Setup (λ) and gives the parameters to attacker A.

Phase 1. Then attacker A can search for the information below:

- **Public key:** Attacker A gives an index i to challenger C and gets the public key pk_i from C.
- **Private key:** Attacker A gives an index j to challenger C and gets the private key sk_j from C.
- **Re-encryption key:** Attacker A gives two public keys to challenger C. Then, challenger C runs $\text{ReKeygen}(pk_i, sk_j)$ and gives the re-encryption key to attacker.
- **Re-encrypt:** Attacker A gives a ciphertext to challenger C. Challenger C runs $\text{ReEnc}(rk_{i \rightarrow j}, CT_i)$ and returns the answer to A.
- **Decrypt:** Challenger C gets the ciphertext from attacker A and returns the message to A.

Challenge. When phase 1 is finished, attacker gives a public key pk and two messages m_0, m_1 to challenger. Challenger chooses a message m_δ randomly and runs $\text{Enc1}(m_\delta, pk, par)$. At last, give the ciphertext to attacker A.

Phase 2. Attacker A can do the same things like phase 1 without encrypt.

Guess. Finally, attacker A guesses which message challenger has chosen.

In this game, A's advantage is

$$adv = |\Pr[\delta' = \delta] - 1/2|.$$

If adv can be ignored in polynomial time, we can call it an ACC2-secure scheme.

7.3. State of the art

7.3.1 The bidirectional PRE

A basic goal of public-key encryption is to allow only the key or keys selected at the time of encryption to decrypt the ciphertext. To change the ciphertext to a different key requires re-encryption of the message with the new key, which implies access to the original cleartext and to a reliable copy of the new encryption key. Intuitively, this seems a fundamental, and quite desirable, property of good cryptography; it should not be possible for an untrusted party to change the key with which a message can be decrypted.

Blaze et al. [36] consider atomic proxy schemes for encryption, identification, and signatures. An encryption proxy key allows B to decrypt messages encrypted for A, and an identification or signature proxy key $\pi_{A \rightarrow B}$ allows A to identify herself as B or to sign for B (i.e., transforms A's signature into B's signature). Generating encryption proxy key $\pi_{A \rightarrow B}$ obviously requires knowledge of at least the secret component of A (otherwise the underlying system is not secure), and similarly generating identification or signature proxy key $\pi_{A \rightarrow B}$ requires B's secret, but the proxy key itself, once generated, can be published safely.

7.3.1.1 Concrete construction

7.3.1.1.1 Proxy encryption

7.3.1.1.1.1 Cryptosystem \mathcal{X} (encryption)

Let p be a prime of the form $2q + 1$ for a prime q and let g be a generator in \mathbb{Z}_p^* ; p and g are global parameters shared by all users. A's secret key a , $0 < a < p - 1$, is selected at random and must be in \mathbb{Z}_{2p}^* , i.e., relatively prime to $p-1$ (A also calculates the inverse $a^{-1} \pmod{2q}$). A publishes the public key $g^a \pmod p$. Message encryption requires a unique randomly selected secret parameter $k \in \mathbb{Z}_{2p}^*$. To encrypt m with A's key, the sender computes and sends two ciphertext values (c_1, c_2) :

$$c_1 = mg^k \pmod p$$

$$c_2 = (g^a)^k \pmod p.$$

Decryption reverses the process; since

$$c_2^{(a^{-1})} = g^k \pmod p,$$

it is easy for A (who knows a^{-1}) to calculate g^k and recover m :

$$m = c_1((c_2^{(a^{-1})})^{-1}) \pmod p.$$

The efficiency of this scheme is comparable to standard ELGamal encryption. The whole game is described in **Figure 7.2**.

7.3.1.1.1.2 Symmetric proxy function for \mathcal{X}

Observe that the c_1 ciphertext component produced by cryptosystem \mathcal{X} is independent of the recipient's public key. Recipient A's key is embedded only in the c_2 exponent; it is sufficient for a proxy function to convert ciphertext for A into ciphertext for B to remove A's key a from c_2 and replace it with B's key b . Part of what a proxy function must do, then, is similar to the first step of the decryption function, raising c_2 to a^{-1} to remove a . The proxy function must also contribute a factor of b to the exponent. Clearly, simply raising c_2 to a^{-1} and then to b would accomplish this, but obviously such a scheme would not qualify as a secure proxy function; anyone who examines the proxy key learns the secret keys for both A and B.

This problem is avoided, of course, by combining the two steps into one. Hence, the proxy key $\pi_{A \rightarrow B}$ is $a^{-1}b$ and the proxy function is simply $c_2^{\pi_{A \rightarrow B}}$. Note that this is a symmetric proxy function; A and B must trust one another bilaterally. B can learn A's secret (by multiplying the proxy key by $b^{?1}$), and A can similarly discover B's

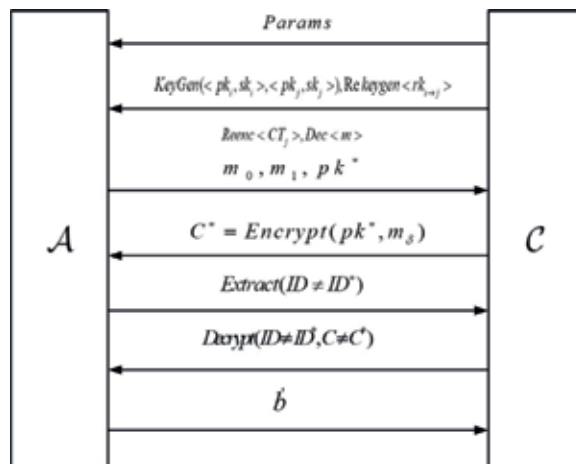


Figure 7.2.
Game in PRE scheme.

key. Observe that applying the proxy function is more efficient than decryption and re-encryption, in that only one exponentiation is required.

7.3.1.1.2 Proxy identification

In this section we describe a discrete-log-based identification scheme. With p , g , and a as before, Alice wishes to convince Charlotte that she controls a ; Charlotte will verify using public key g^a . As before, the proxy key $\pi_{A \rightarrow B}$ will be a/b —it will be safe to publish a/b , and Alice and Charlotte can easily use a/b to transform the protocol, so Charlotte is convinced that Alice controls b .

Note that in the case of a secure identification proxy key that transforms identification by A into identification by B , it is B whose secret is required to construct the proxy key because identification as B should not be possible without B 's cooperation.

7.3.1.1.2.1 Cryptosystem \mathcal{Y} (identification)

Let p and g be a prime and a generator in \mathbb{Z}_p^* , respectively. Alice picks random $a \in \mathbb{Z}_p^*$ to be her secret key and publishes g^a as her public key. Each round of the identification protocol is as follows:

- Alice picks a random $k \in \mathbb{Z}_p^*$ and sends Charlotte $s_1 = g^k$.
- Charlotte picks a random bit and sends it to Alice.
- Depending on the bit received, Alice sends Charlotte either $s_2 = k$ or $s'_2 = k/a$.
- Depending on the bit, Charlotte checks that $(g^a)^{s'_2} = s_1$ or that $g^{s_2} = g^k$.

Via the above steps, Charlotte can identify the message sent by Alice.

7.3.1.1.2.2 Symmetric proxy function for \mathcal{Y}

A symmetric proxy key is a/b . Suppose Charlotte wants to run the protocol with g^b instead of g^a , either Alice or Charlotte or any intermediary can use the proxy key to convert Alice's responses k/a to k/b .

7.3.1.1.3 Proxy signature

Now we will see how to use the proxy identification scheme to construct a proxy signature scheme. We suppose there exists a hash function h whose exact security requirements will be discussed below. The parameters p , g , a , and b are as before.

7.3.1.1.3.1 Cryptosystem \mathcal{Z} (signature)

To sign a message m , Alice picks k_1, k_2, \dots, k_ℓ at random and computes $g^{k_1}, \dots, g^{k_\ell}$. Next Alice computes $h(g^{k_1}, \dots, g^{k_\ell})$ and extracts ℓ pseudorandom bits $\beta_1, \dots, \beta_\ell$. For each i , depending on the i th pseudorandom bit, Alice (who knows a) computes $s_{2,i} = (k_i - m\beta_i)/a$, that is, $s_{2,i} = (k_i - m)/a$ or $s_{2,i} = k_i/a$. The signature consists of two components:

$$\begin{aligned} s_1 &= (g^{k_1}, \dots, g^{k_\ell}) \\ s_2 &= ((k_1 - m\beta_1)/a, \dots, (k_\ell - m\beta_\ell)/a). \end{aligned}$$

To verify the signature, first, β_i s are recovered using the hash function. The signature is then verified one "round" at a time, where the i th round is $(g^{k_1}, (k_i - m\beta_i)/a)$. To verify $(g^k, (k - m\beta)/a)$ using public key g^a , the recipient Charlotte raises (g^a) to the power $(k - m\beta)/a$ and checks that it matches $g^k/g^{m\beta}$.

7.3.1.1.3.2. Symmetric proxy function for \mathcal{Z}

A symmetric proxy key $\pi_{A \rightarrow B}$ for this signature scheme is a/b . The proxy function Π leaves s_1 alone and maps each component $s_{2,i}$ to $s_{2,i}\pi_{A \rightarrow B}$.

7.3.1.2 Security proof

7.3.1.2.1 Secure of \mathcal{X}

First, we will prove the cryptosystem \mathcal{X} is secure by confirming that cleartext and secret keys cannot be recovered from ciphertext and public keys. Beyond that, we also show that publishing the proxy key compromises neither messages nor secret keys. Since recovering a secret key enables an adversary to recover a message and since cryptanalysis is easier with more information (i.e., a proxy key), it is sufficient to show that no cleartext is recoverable from ciphertext, public keys, and proxy keys. Specifically, we will show that the problem of recovering m from

$$(g^a, g^b, g^c, \dots, mg^k, g^{ak}, a^{-1}b, a^{-1}c, \dots),$$

is at least as hard as Diffie-Hellman.

The intuition of security proof in the scheme is illustrated in **Figure 7.3**.

7.3.1.2.2 Secure of \mathcal{Y}

Protocol \mathcal{Y} , with or without proxy keys published, is a zero knowledge protocol that convinces the verifier that the prover knows the secret key.

Proof. The proof is found in [36].

7.3.1.2.3 Secure of \mathcal{Z}

This scheme relies on the existence of a hash function h . Specifically (hash assumption), we assume there exists a function h such that:

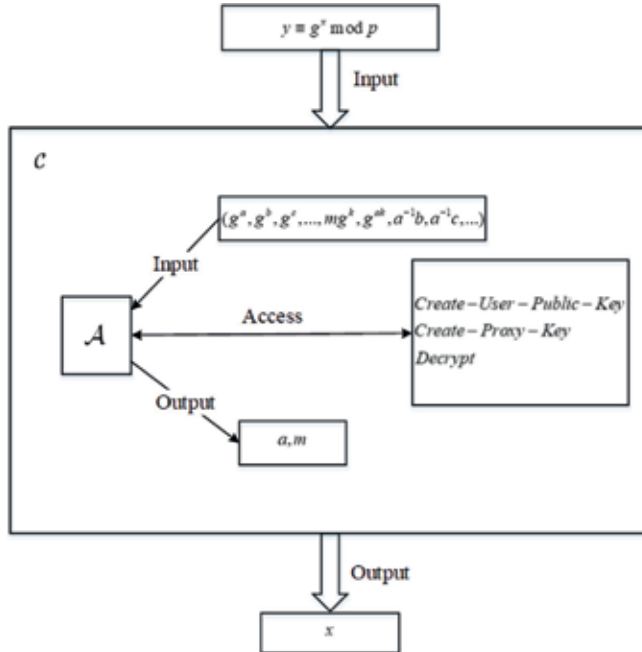


Figure 7.3. Intuition of security proof in PRE scheme.

- On random input (g^a, m) , it is difficult to generate r_i and β_i such that $h(g^{(ar_1+m\beta_1)^{-}}, \dots, g^{(ar_\ell+m\beta_\ell)^{-}}) = \beta_1 \dots \beta_\ell \cdot z$.

- More generally, it is difficult to generate such r_i and β_i on input g^a, m , and samples of signatures on random messages signed with a .

It is not our intention to conjecture about the existence of such functions h . In particular, we do not know the relationship between the hash assumption and assumptions about collision freedom or hardness to invert. We note that this generic transformation of a protocol to a signature scheme has appeared in the literature written by [206].

We now analyze the hash assumption. Note that in order to produce a legitimate signature on m that verifies with g^a , a signer needs to produce $\langle gk_i \rangle$ and $\langle (k_i - m\beta_i)/a \rangle$. Thus, putting $\langle \beta_i \rangle = h(\langle gk_i \rangle)$ and then $\langle r_i \rangle = \langle (k_i - m\beta_i)/a \rangle$, it is straightforward to see that the signer could actually produce r_i 's and β_i 's of the stated type in the course of producing the signature.

While we do not address the security of h , we can state that issuing proxy keys does not weaken the system.

7.3.2 The unidirectional PRE

As we discussed before, a bidirectional PRE scheme can only be used in a limited situation. As a result, a new concrete unidirectional scheme is proposed by [39] to resolve the problem. A unidirectional re-encryption scheme means the re-encryption key is used to transform the ciphertext from for Alice's key into for Bob's, while it cannot transform it from for Bob to for Alice. The authors introduce a concrete unidirectional mechanism involving some significant security properties such as collusion resistance (one of the most essential goals in the design of unidirectional schemes) without abandoning the property of chosen-ciphertext attack (CCA) security in the standard model. Although like other unidirectional schemes, the new introduced scheme is single hop while it is still efficient, and a noninteractive complexity assumption in bilinear groups is required when proving the security of the scheme.

7.3.2.1 Concrete construction

Like the construction proposed by [38], the presented scheme uses a strongly unforgeable one-time signature to tie several ciphertext components altogether and offers a safeguard against chosen-ciphertext attacks in the fashion of [186]. For simplicity, the description below assumes that verification keys of the one-time signature are encoded as elements from \mathbb{Z}_p^* . In practice, such verification keys are typically much longer than $|p|$, and a collision-resistant hash function should be applied to map them onto \mathbb{Z}_p^* :

- **Setup** (λ) . The algorithm is given a security parameter λ and chooses several bilinear map groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order $p2^\lambda$. Then it generates $g, u, v \in \mathbb{G}$ and chooses a strongly unforgeable one-time signature scheme $\text{Sig} = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ (Sig means one-time signatures and you can learn more details in [39]'s work). The global parameters are

$$par := \{\mathbb{G}, \mathbb{G}_T, g, u, v, \text{Sig}\}.$$

- **Keygen** (λ) . User i sets his public key as $X_i = g^{x_i}$ for a random $x_i \in \mathbb{Z}_p^*$.

- **ReKeygen**(x_i, X_j). The algorithm is given user i 's private key x_i and user j 's public key X_j and generates the unidirectional re-encryption key

$$R_{ij} = X_j^{1/x_i} = g^{x_j/x_i}.$$
- **Enc1**(m, X_i, par). The encryptor encrypts a message $m \in \mathbb{G}_T$ under the public key X_i at the first level and proceeds as follows.

1. Select a one-time signature key pair $(ssk, svk) \xleftarrow{R} \mathcal{G}(\lambda)$ and set $C_1 = svk$.

2. Pick $r, t \xrightarrow{R} \mathbb{Z}_p^*$ and compute

$$C_2' = X_i^t \quad C_2'' = g^{1/t} \quad C_2''' = X_i^{rt} \quad C_3 = e(g, g)^r \cdot m \quad C_4 = (u^{svk} \cdot \phi)^r.$$

3. Generate a one-time signature $\sigma = \mathcal{S}(ssk, (C_3, C_4))$ on (C_3, C_4) .

The ciphertext is $C_i = (C_1, C_2', C_2'', C_2''', C_3, C_4, \sigma)$.

- **Enc2**(m, X_i, par). The encryptor encrypts a message $m \in \mathbb{G}_T$ under the public key X_i at level 2 and conducts the following steps:

1. Select a one-time signature key pair $(ssk, svk) \xleftarrow{R} \mathcal{G}(\lambda)$ and set $C_1 = svk$.

2. Choose $r \xrightarrow{R} \mathbb{Z}_p^*$ and compute

$$C_2 = X_i^r \quad C_3 = e(g, g)^r \cdot m \quad C_4 = (u^{svk} \cdot \phi)^r.$$

3. Generate a one-time signature $\sigma = \mathcal{S}(ssk, (C_3, C_4))$ on the pair (C_3, C_4) .

The ciphertext is $C_i = (C_1, C_2, C_3, C_4, \sigma)$.

- **ReEnc**(R_{ij}, C_i). The algorithm is given the re-encryption key $R_{ij} = g^{x_j/x_i}$ and a ciphertext $C_i = (C_1, C_2, C_3, C_4, \sigma)$, and check the validity of the latter by testing the following conditions:

$$e(C_2, u^{C_1} \cdot \phi) = e(X_i, C_4) \quad (7.1)$$

$$\mathcal{V}(C_1, \sigma, (C_3, C_4)) = \perp \quad (7.2)$$

If all the steps are well formed, C_i is re-encrypted by choosing $t \xrightarrow{R} \mathbb{Z}_p^*$ and computing

$$C_2' = X_i^t \quad C_2'' = R_{ij}^{1/t} = g^{(x_j/x_i)t-1} \quad C_2''' = C_2^t = X_i^{rt}.$$

The re-encrypted ciphertext is

$$C_j = (C_1, C_2', C_2'', C_2''', C_3, C_4, \sigma).$$

If they are ill formed, C_i is declared “invalid.”

- **Dec1**(C_j, sk_j). The validity of a level 1 ciphertext C_j is checked by testing if

$$e(C'_2, C''_2) = e(X_j, g) \quad (7.3)$$

$$e(C'''_2, u^{C_1} \cdot v) = e(C'_2, C_4) \quad (7.4)$$

$$\mathcal{V}(C_1, \sigma, (C_3, C_4)) = 1. \quad (7.5)$$

If relations (7.3)–(7.5) hold, the plaintext $m = C_3/e(C''_2, C_2)^{1/x_j}$ will be output. Otherwise, the algorithm exports “invalid.”

- **Dec2**(C_i, sk_i). If the level 2 ciphertext $C_i = (C_1, C_2, C_3, C_4, \sigma)$ satisfies relations (7.1)–(7.2), receiver i can obtain $m = C_3/e(C_2, g)^{1/x_i}$. The algorithm outputs “invalid” otherwise.

Outputs of the re-encryption algorithm are perfectly indistinguishable from level 1 ciphertext produced by the sender. Indeed, if $\tilde{t} = tx_i/x_j$, we can write

$$C'_2 = X_i^t = X_j^{\tilde{t}} \quad C''_2 = g^{(x_j/x_i)t-1} = g^{\tilde{t}-1} \quad C'''_2 = X_i^{rt} = X_j^{r\tilde{t}}$$

As in the original scheme described in [37], second-level ciphertext, which is stored in remote untrusted servers, can be changed into first-level ciphertext encrypted for the same receiver if the identity element of \mathbb{G} is used as a re-encryption key.

In the first-level decryption algorithm, relations (7.3)–(7.5) guarantee that re-encrypted ciphertext have the correct shape. Indeed, since the relation $C_4 = (u^{C_1} \cdot v)^r$ holds for some unknown exponent $r \in \mathbb{Z}_p$, equality (7.4) implies that $C'''_2 = C_2^r$. From (7.3), it comes that $e(C''_2, C'''_2) = e(X_j, g)^r$.

We finally note that first-level ciphertext can be publicly re-randomized by converting (C'_2, C''_2, C'_3) into $(C'^s_2, C''^{1/s}_2, C'^s_3)$ for a random $s \in \mathbb{Z}_p^*$. However, the pairing value $e(C''_2, C'''_2)$ remains constant and re-randomizations of a given first-level ciphertext are publicly detectable.

7.3.2.2 Security proof

For convenience, we will prove security under an equivalent formulation of the 3-wDBDHI assumption.

Lemma 7.1. *The 3-wDBDHI problem is equivalent to decide whether T equals $e(g, g)^{b/a^2}$ or a random value given $(g, g^{1/a}, g^a, g^{(a^2)}, g^b)$ as input.*

Proof 7.1. With the given elements $(g, g^{1/a}, g^a, g^{(a^2)}, g^b, T)$, we can construct a 3-wDBDHI instance by setting $(y = g^{1/a}, y^A = g, y^{(A^2)} = g^a, y^{(A^3)} = g^{(a^2)}, y^B = g^b)$, which implicitly defines $A = a$ and $B = ab$. Then, we compute $e(y, y)^{B/A} = e(g, g)^{b/a^2}$. The converse implication is easily established and demonstrates the equivalence between both problems.

The intuition of security proof in the scheme is illustrated in **Figure 7.4**.

Theorem 7.1. *Assuming the strong unforgeability of the one-time signature, the scheme is RCCA-secure at level 2 under the 3-wDBDHI assumption.*

Proof 7.2. Let $(g, A_{-1} = g^{1/a}, A_1 = g^a, A_2 = g^{(a^2)}, B = g^b, T)$ be a modified 3-wDBDHI instance. We build an algorithm \mathcal{B} deciding if $T = e(g, g)^{b/a^2}$.

Before presenting \mathcal{B} , we assume an event called F_{OTS} and observe the probability it may occurs. Let $C^* = (svk^*, C_2^*, C_3^*, C_4^*, \sigma^*)$ denote the challenge ciphertext given to \mathcal{A} in the game. Let F_{OTS} be the event that, at some point, \mathcal{A} issues a decryption

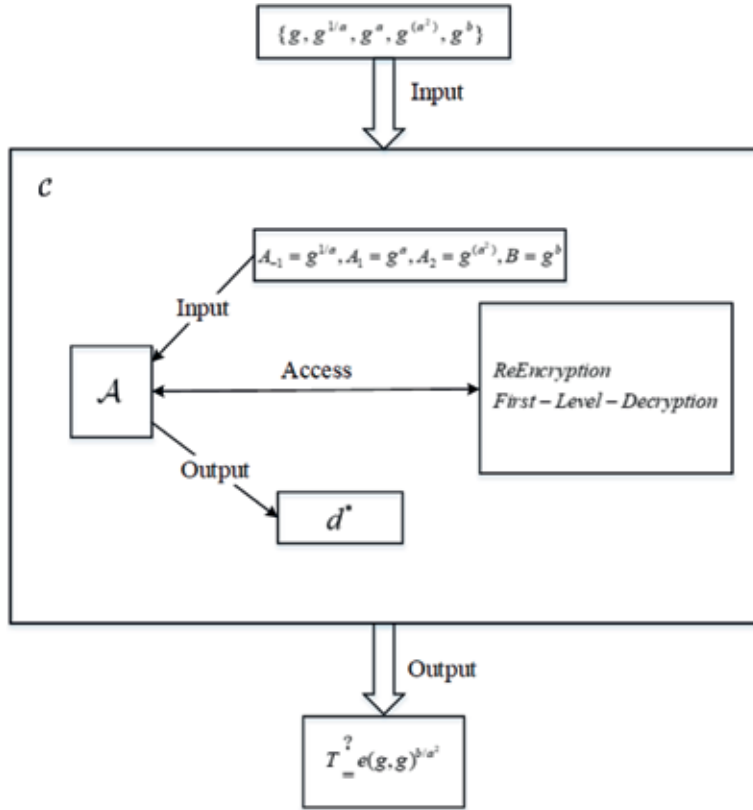


Figure 7.4.
Intuition of security proof in IUCCS-PRE scheme.

query for a first-level ciphertext $C = (svk^*, C_2^-, C_2', C_2'', C_3, C_4, \sigma)$ or a re-encryption query $C = (svk^*, C_2, C_3, C_4, \sigma)$ where $(C_3, C_4, \sigma) \neq (C_3^*, C_4^*, \sigma^*)$ but $\mathcal{V}(\sigma, svk, (C_3, C_4)) = 1$. In the “find” stage, \mathcal{A} cannot learn any information about svk^* . Hence, the probability of a pre-challenge occurrence of F_{OTS} does not exceed $q_O \cdot \delta$ if q_O is the overall number of oracle queries and δ denotes the maximal probability (which does not exceed $1/p$ according to the assumption) that any one-time verification key svk is output by \mathcal{G} . In the “guess” stage, F_{OTS} clearly gives rise to an algorithm breaking the strong unforgeability of the one-time signature. Therefore, the probability $\Pr[F_{OTS}] \leq q_O/p + Adv^{OTS}$, where the second term accounts for the probability of definition 5, must be negligible by assumption.

We now proceed with the description of \mathcal{B} that simply halts and outputs a random bit if F_{OTS} occurs. In a preparation phase, \mathcal{B} generates a one-time signature key pair $(ssk^*, svk^*) \leftarrow \mathcal{G}(\lambda)$ and provides \mathcal{A} with public parameters including $u = A_1^{\alpha_1}$ and $v = A_1^{-\alpha_1 svk^*} \cdot A_2^{\alpha_2}$ for random $\alpha_1, \alpha_2 \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$. Observe that u and v define a “hash function” $F(svk) = u^{svk} \cdot v = A_1^{\alpha_1 (svk - svk^*)} \cdot A_2^{\alpha_2}$. In the following, we call HU as the set of honest parties, including user i^* that is assigned the target public key pk^{**} , and CU as the set of corrupt parties. Throughout the game, \mathcal{A} ’s environment is simulated as follows:

- **Key generation.** Public keys of honest users $i \in HU \setminus \{i^*\}$ are defined as $X_i = A_1^{x_i} = g^{ax_i}$ for a randomly chosen $x_i \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$. The target user’s public key is set as $X_{i^*} = A_2^{x_{i^*}} = g^{(x_{i^*} a^2)}$ with $x_{i^*} \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$. The key pair of a corrupt user $i \in CU$ is set

as $(X_i = g^{x_i}, x_i)$, for a random $x_i \in \mathbb{Z}_p^*$, so that pairs (X_i, x_i) can be given to \mathcal{A} . To generate re-encryption keys R_{ij} from player i to player j , \mathcal{B} has to distinguish several situations:

- If $i \in \text{HU} \setminus \{i^*\}$ and $j = i^*$, \mathcal{B} returns $R_{ii^*} = A_1^{x_i/x_i} = g^{x_i a^2 / (ax_i)}$ which is a valid re-encryption key.
- If $i = i^*$ and $j \in \text{HU} \setminus \{i^*\}$, \mathcal{B} responds with $R_{i^*j} = A_{-1}^{x_i/x_i^*} = g^{(ax_i/(x_i a^2))}$ that has also the correct distribution.
- If $i, j \in \text{HU} \setminus \{i^*\}$, \mathcal{B} returns $R_{ij} = g^{x_j/x_i} = g^{(ax_j)/(ax_i)}$.
- If $i \in \text{HU} \setminus \{i^*\}$ and $j \in \text{CU}$, \mathcal{B} outputs $R_{ij} = A_{-1}^{x_j/x_i} = g^{x_j/(ax_i)}$ which is also computable.
- **Re-encryption queries.** When receiving a re-encryption query from user i to user j for a second-level ciphertext $C_i = (C_1, C_2, C_3, C_4, \sigma)$, \mathcal{B} returns “invalid” if relations (7.1)–(7.2) are not satisfied.
 - If $i \neq i^*$ or if $i = i^*$ and $j \in \text{HU} \setminus \{i^*\}$, \mathcal{B} simply re-encrypts the corresponding ciphertext with the re-encryption key R_{ij} which is available in either case.
 - If $i = i^*$ and $j \in \text{CU}$.
 - If $C_1 \neq \text{svk}^*$, \mathcal{B} can observe the occurrence of F_{OTS} and abort from the game. Indeed, re-encryptions of the challenge ciphertext toward corrupt users are disallowed in the “guess” stage. Therefore, $(C_3, C_4, \sigma) \neq (C_3^*, C_4^*, \sigma^*)$ since we would have $C_2 \neq G_2^*$ and $i \neq i^*$ if $(C_3, C_4, \sigma) = (C_3^*, C_4^*, \sigma^*)$.
- We now discuss the left case $C_1 \neq \text{svk}^*$, $i = i^*$, and $j \in \text{CU}$. With the given ciphertext $C_2^{1/x_i^*} = A_2^r$ and $C_4 = E(\text{svk})^r = (A_1^{\alpha_1(\text{svk}-\text{svk}^*)} \cdot A_2^{\alpha_2})^r$, \mathcal{B} can compute

$$A_1^r = (g^a)^r = \frac{C_4}{C_2^{\alpha_2/x_i^*}} \left(\frac{1}{\alpha_1(\text{svk}-\text{svk}^*)} \right) \quad (7.6)$$

Knowing g^{ar} and user j 's private key x_j , \mathcal{B} picks $t \in \mathbb{Z}_p^*$ to compute

$$C_2^{t'} = A_1^t = g^{at} \quad C_2^{t''} = A_{-1}^{x_j/t} = (g^{1/a})^{x_j/t} \quad C_2^{t'''} = (A_1^r)^t = (g^{ar})^t$$

and returns $C_j = (C_1, C_2^{t'}, C_2^{t''}, C_2^{t'''}, C_3, C_4, \sigma)$ which has the proper distribution. Indeed, if we set $\tilde{t} = at/x_j$, we have

$$C_2^{t'} = X_{j\tilde{t}} \quad C_2^{t''} = g^{1/\tilde{t}} \quad C_2^{t'''} = X_j^{\tilde{t}}$$

- **First-level decryption queries.** At any time, \mathcal{A} may ask for the decryption of a first-level ciphertext $C_j = (C_1, C_2^{t'}, C_2^{t''}, C_2^{t'''}, C_3, C_4, \sigma)$ under a public key X_j . For

such a request, \mathcal{B} returns “invalid” if relations (7.3)–(7.5) do not hold. We assume that $j \in \text{HU}$ since \mathcal{B} can decrypt using the known private key otherwise. Let us first assume that $C_1 = \mathcal{E}_{sk^*}^*(m)$. If $(C_3, C_4, \sigma) \neq (C_3^*, C_4^*, \sigma^*)$, \mathcal{B} is presented with an occurrence of FOTS and halts. If $(C_3, C_4, \sigma) = (C_3^*, C_4^*, \sigma^*)$, \mathcal{B} outputs \perp which deems C_j as a derivative of the challenge pair (C^*, X_{i^*}) . Indeed, it must be the case that $e(C_2', C_2'') = e(g, X_j)^r$ for the same underlying exponent r as in the challenge phase. We now assume $C_1 \neq \mathcal{E}_{sk^*}^*(m)$.

- If $j \in \text{HU} \setminus \{i^*\}$, $X_j = g^{ax_j}$ for a known $x_j \in \mathbb{Z}_p^*$. The validity of the ciphertext ensures that $e(C_2', C_2'') = e(X_j, g)^r = e(g, g)^{arx_j}$ and $C_4 = \mathcal{E}(svk)^r = g^{\alpha_1 ar (svk - svk^*)} \cdot g^{a^2 r \alpha_2}$ for some $r \in \mathbb{Z}_p$. Therefore,

$$e(C_4, A_{-1}) = e(C_4, g^{1/a}) = e(g, g)^{\alpha_1 r (svk - svk^*)} \cdot e(g, g)^{ar \alpha_2}, \quad (7.7)$$

and

$$e(g, g)^r = \frac{e(C_4, A_{-1})}{e(C_2', C_2'')^{\alpha_2/x_j}}, \quad (7.8)$$

reveals the plaintext m since $svk \neq svk^*$.

- If $j = i^*$, we have $X_j = g^{(x_i^* a^2)}$ for a known exponent $x_i^* \in \mathbb{Z}_p^*$. Since we know that

$$\begin{aligned} e(C_2', C_2'') &= e(X_{i^*}, g)^r = e(g, g)^{a^2 r x_i^*}, \\ e(C_4, g) &= e(g, g)^{\alpha_1 ar (svk - svk^*)} \cdot e(g, g)^{a^2 r \alpha_2}, \end{aligned}$$

\mathcal{B} can first obtain

$$\gamma = e(g, g)^{ar} = \frac{e(C_4, g)}{e(C_2', C_2'')^{\alpha_2/x_i^*}}.$$

Together with relation (7.7), γ in turn uncovers

$$e(g, g)^r = \frac{e(C_4, A_{-1})}{\gamma^{\alpha_2/x_i^*}}.$$

and gets the plaintext $m = \mathcal{D}_{sk^*}(e(g, g)^r)$.

In the “guess” stage, \mathcal{B} must check that m differs from messages m_0, m_1 involved in the challenge query. If $m \in \{m_0, m_1\}$, \mathcal{B} returns \perp according to the RCCA security rules.

- **Challenge.** When she decides that the first phase is over, \mathcal{A} chooses messages (m_0, m_1) . At this stage, \mathcal{B} flips a coin $d \in \{0, 1\}$ and generates the challenge ciphertext C^* as

$$C_1^* = \mathcal{E}_{sk^*}^*(m) = \mathcal{E}_{X_{i^*}}^*(C_3^*) = m \cdot \mathcal{E}_{d^*}^*(C_4^*) = \mathcal{E}^*(m, d^*),$$

and $\sigma = \mathcal{S}(sk^*, (C_3^*, C_4^*))$.

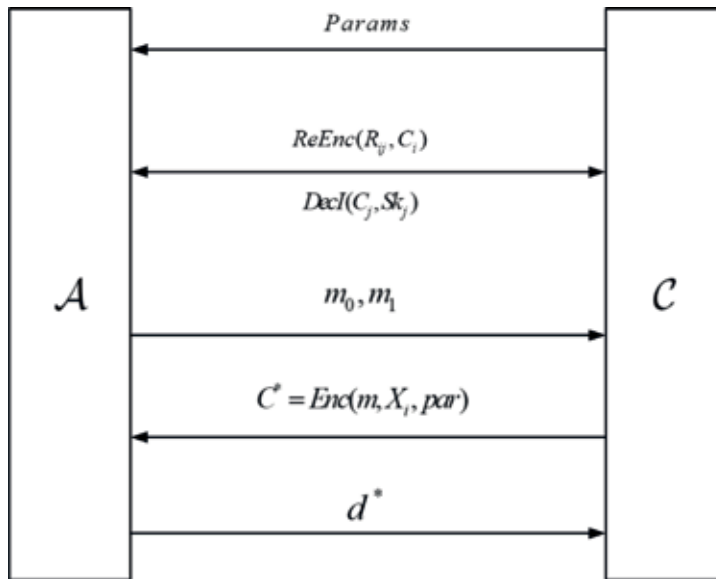


Figure 7.5.
 RCCA game in IUCCS-PRE scheme.

The whole game is described in **Figure 7.5**.

Since $X_{i^*} \leftarrow \mathcal{A}_2^{x_{i^*}} = g^{x_{i^*} a^2}$ and $B = g^b$, $C^{* \leftarrow}$ is a valid encryption of m_{d^*} with the random exponent $r = b/a^2$ if $T = e(g, g)^{b/a^2}$. In contrast, if T is a random element in \mathbb{G}_T , $C^{* \leftarrow}$ perfectly hides m_{d^*} , and \mathcal{A} cannot guess d^* with better probability than $1/2$. When \mathcal{A} eventually outputs her result $d' \in \{0, 1\}$, \mathcal{B} decides that $T = e(g, g)^{b/a^2}$ if $d' = d^*$ or T is random otherwise.

Theorem 7.2. *Assuming the strong unforgeability of the one-time signature, the scheme is RCCA-secure at level 1 under the 3-wDBDHI assumption.*

Proof 7.3. The proof is very similar to the one of theorem 1. We construct an algorithm; \mathcal{B} is given a 3-wDBDHI instance $(g, A_{-1} = g^{1/a}, A_1 = g^a, A_2 = g^{(a^2) \leftarrow}, B = g^b, T)$ and uses the adversary \mathcal{A} to decide if $T = e(g, g)^{b/a^2}$.

Before describing \mathcal{B} , we consider the same event F_{OTS} as the one in the proof of theorem 1 except that it can only arise during a decryption query (since there is no re-encryption oracle). Assuming the strong unforgeability of the one-time signature, such an event occurs with negligible probability as detailed in the proof of Theorem 1. We can now describe our simulator \mathcal{B} that simply aborts and outputs a random bit if F_{OTS} ever occurs. Let also $C^* = (C_1^*, C_2^*, C_2'^*, C_2''^*, C_3^*, C_4^*, \sigma^*)$ denote the challenge ciphertext at the first level.

Algorithm \mathcal{B} generates a one-time key pair $(ssk^*, svk^*) \leftarrow \mathcal{G}(\lambda)$ and the same public parameters as in theorem 1. Namely, it sets $u = \mathcal{A}_1^{\alpha_1}$ and $v = \mathcal{A}_1^{-\alpha_1 svk^*} \cdot \mathcal{A}_2^{\alpha_2}$ with $\alpha_1, \alpha_2 \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ so that $F(svk) = u^{svk} \cdot v = \mathcal{A}_1^{\alpha_1 (svk - svk^*)} \cdot \mathcal{A}_2^{\alpha_2}$. As in the proof of the theorem 1, i^* identifies the target receiver. The attack environment is simulated as follows:

- **Key generation.** For corrupt users $i \in \text{CU}$ and honest ones, $i \in \text{HU} \setminus \{i^*\}$, \mathcal{B} sets $X_i = g^{x_i}$ for a random $x_i \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$. The target user's public key is defined as $X_{i^*} = \mathcal{A}_1$. For corrupt users $i \in \text{CU}$, X_i and x_i are both revealed. All

re-encryption keys are computable and given to \mathcal{A} . Namely, $R_{ij} = g^{x_j/x_i}$ if $i, j \neq i^*$; $R_{i^*j} = A_{-1}^{x_j}$ and $R_{ji^*} = A_1^{1/x_j}$ for $j \neq i^*$.

- **First-level decryption queries.** When a ciphertext $C_j = (C_1, C_2, C_2', C_2'', C_3, C_4, \sigma)$ is queried for decryption with a public key X_j , \mathcal{B} returns “invalid” if relations (7.3)–(7.5) do not hold. We assume that $j = i^*$ since \mathcal{B} can decrypt using the known private key x_j otherwise. We have $C_2 = A_1^t$, $C_2' = g^{1/t}$, $C_2'' = A_1^{rt}$ for unknown exponents $r, t \in \mathbb{Z}_p^*$. Since $e(C_2, C_2') = e(g, g)^{ar}$ and

$$e(C_4, A_{-1}) = e(g, g)^{\alpha_1 r (svk - svk^*)} \cdot e(g, g)^{ar\alpha_2},$$

\mathcal{B} can obtain

$$e(g, g)^r = \frac{e(C_4, A_{-1})}{e(C_2, C_2')} \left(\frac{1}{\alpha_1 (svk - svk^*)} \right),$$

which reveals the plaintext $m = C_3 / e(g, g)^r$ as long as $svk \neq svk^*$. In the event that $C_1 = svk^*$ holds in a post-challenge query:

- If $e(C_2, C_2') = e(C_2'', C_2'')$, \mathcal{B} returns \perp which means that C_j is merely a re-randomization (and thus a derivative) of the challenge ciphertext.
- Otherwise, we necessarily have $(C_3, C_4, \sigma) \neq (C_3, C_4, \sigma)$, which is an occurrence of F_{OTS} and implies \mathcal{B} 's termination.

In the “guess” stage, \mathcal{B} must ensure that m differs from messages m_0, m_1 of the challenge phase before answering the query.

- **Challenge.** When the first phase is over, \mathcal{A} outputs messages (m_0, m_1) and \mathcal{B} flips a bit $d \in \{0, 1\}$. Then, it chooses $\mu \in \mathbb{Z}_p^*$ and sets

$$\begin{aligned} C_2 &= A_2^\mu, \quad C_2' = A_{-1}^{1/\mu}, \quad C_2'' = B^\mu \\ C_1 &= svk^*, \quad C_3 = m_d, \quad C_4 = B^{\alpha_2}, \end{aligned}$$

and $\sigma = S(ssk^*, (C_3, C_4))$.

Since $X_{i^*} = A_1$ and $B = g^b$, C^* is a valid encryption of m_{d^*} with the random exponents $r = b/a^2$ and $t = a\mu$ whenever $T = e(g, g)^{b/a^2}$. When T is a random message, C^* perfectly hides m_{d^*} and \mathcal{A} cannot guess d^* with better probability than 1/2. Eventually, \mathcal{B} bets that $T = e(g, g)^{b/a^2}$ if \mathcal{A} correctly guesses d^* and that T is random otherwise.

7.3.3 Identity-based PRE

In an identity-based proxy re-encryption (IBPRE) scheme, Alice can give proxy a re-encryption key. Proxy can convert an encryption computed under Alice's public key into an encryption intended for Bob. Proxy isn't trusted, so it cannot get Alice's and Bob's private key and it also cannot get any information from

ciphertexts. Green and Ateniese [45] firstly use PRE to deal with this problem. Now, we will talk about a kind of IBPRE scheme with CCA security.

7.3.3.1 Concrete construction

- **Setup(k).** Let n be a polynomial in the security parameter k . Let $e: G_1 \times G_1 \rightarrow G_T$ be a bilinear map, where G_1, G_T have order q and $G_1 = \langle g \rangle$. Selecting a parameter $s \in \mathbb{Z}_p^{*}$ and output $\text{par} = (H_1, H_2, H_3, H_4, H_5, H_6, g, g^s)$, $\text{msk} = s$. H_{1-6} is defined as

$$H_1 : \{0, 1\}^* \rightarrow G_1, H_2 : \{0, 1\}^* \rightarrow G_1, H_3 : \{0, 1\}^* \rightarrow G_1 \\ H_4 : G_T \times \{0, 1\}^n \rightarrow \mathbb{Z}_q^*, H_5 : G_T \rightarrow \{0, 1\}^n$$

- **Keygen(par, msk, id).** To add identity information into the key, return $sk_{id} = H_1(id)^s, id \in \{0, 1\}^*$.
- **Enc(par, id, $\{0, 1\}^n$).** To encrypt m under $id \in \{0, 1\}^*$, the steps are in the following words:
 - Select $\delta \in G_T$, and set $r = H_4(\delta, m)$.
 - Compute $c' = (g^r, \delta \cdot e(g^s, H_1(id))^r), m \oplus H_5(\delta)$.
 - Compute $S = H_3(id || c')^r$.
 - Output the ciphertext $c = \langle S, c' \rangle$.
- **ReKeygen(par, sk_{id_1}, id_1, id_2).** To compute the re-encryption key from id_1 to id_2 :
 - Select $N \in \{1, 0\}^n$, and compute $K = e(sk_{id_1}, H_1(id_2))$.
 - Output $rk_{id_1 \rightarrow id_2} = \langle N, H_2(K || id_1 || id_2 || N) \cdot sk_{id_1} \rangle$
- **ReEnc(par, $rk_{id_1 \rightarrow id_2}, c_{id_1}$).** To re-encrypt a first-level ciphertext, first parse c_{id_1} as $\langle S, A, B, C \rangle$, and parse $rk_{id_1 \rightarrow id_2}$ as $\langle N, R \rangle$. Next:
 - Let $h = H_3(id_1 || \langle A, B, C \rangle)$.
 - Check if $e(g, S) = e(h, A)$. If not, return \perp .
 - Otherwise, select $t \in \mathbb{Z}_q^{*}$ and compute $B' = B / (e(A, R \cdot h^t) / e(g^t, S))$.
 - Output the re-encrypted ciphertext $c_{id_2} = \langle A, B', C, id_1, N \rangle$.
- **Dec1(par, sk_{id}, c_{id}).** To decrypt a first-level ciphertext, first parse c_{id} as $\langle S, A, B, C \rangle$. Next:
 - Let $h = H_3(id, \langle A, B, C \rangle)$.
 - Select $t \in \mathbb{Z}_q^{*}$, and compute $\delta' = B / e(A, sk_{id} \cdot h^t) / e(g^t, S)$.

- Compute $m' = C \oplus H_5(\delta')$, and $r' = H_4(\delta', m')$.
- Verify that $S = h^{r'}$ and $A = g^{r'}$. If either check fails, return \perp ; otherwise output m' .
- **Dec2(par, sk_{id}, c_{id})**. To decrypt a second-level ciphertext, first parse c_{id} as (A, B, C, id_{src}, N) . Next:
 - Compute $K = e(H_1(id_{src}, sk_{id}))$.
 - Compute $\delta' = B \bullet e(A, H_2(K || id_{src} || id || N))$.
 - Compute $m' = C \oplus H_5(\delta', m')$.
 - Verify that $A = g^{r'}$. If this check fails, return \perp ; otherwise output m' .

Now, we talk about the correctness of this scheme. Firstly, we show correctness for first-level ciphertext. Let $c_{id_1} = \langle S, A, B, C \rangle = \langle h^r, g^r, \delta \bullet e(g^s, H_1(id)^r), m \oplus H_5(\delta) \rangle$ be the first-level encryption of m under id_1 , with $h = H_3(id_1 || \langle A, B, C \rangle)$. Let $sk_1 = H_1(id_1)^s$ be the decryption key. For a random $t \in \mathbb{Z}_q^*$, the decryption process proceeds as follows:

$$(\delta \bullet e(g^s, H_1(id_1)^r)) / (e(g^r, H_1(id_1)^s \bullet h^t) / e(g^t, h^r)) = \delta,$$

$$H_5(\delta) \oplus (m \oplus H_5(\delta)) = m$$

$$g^{H_4(\delta, m)} \stackrel{?}{=} g^r$$

$$h^{H_4(\delta, m)} \stackrel{?}{=} h^r.$$

The correctness under re-encryption is shown as follows. Given a first-level ciphertext and a correctly formed re-encryption key $rk_{id_1 \rightarrow id_2} = \langle N, R \rangle$, then we check the ciphertext as

$$e(g, h^r) \stackrel{?}{=} e(h, g^r).$$

Recall that $R = sk_{id_1} \bullet W$ where $W = H_2(e(H_1(id_1)^s, H_1(id_2)^s) || id_1 || id_2 || N)$. To generate the second-level ciphertext $c_{id_2} = (A, B', C, id_1, N)$, for some $t \in \mathbb{Z}_q^*$, we obtain

$$B' = (\delta \bullet e(g^s, H_1(id_1)^r)) / (e(g^r, R \bullet h^t) / e(g^t, h^r)) = \delta / e(g^r, W).$$

Given $sk_{id_2} = H_1(id_2)^s$ we decrypt $c_{id_2} = (A, B', C, id_1, N) = (g^r, \delta / e(g^r, W), m \oplus H_5(\delta), N)$ as follows:

$$H_2(e(H_1(id_1), H_1(id_2)^s) || id_1 || id_2 || N) = W$$

$$(\delta / e(g^r, W)) \bullet e(g^r, W) = \delta$$

$$H_5(\delta) \oplus (m \oplus H_5(\delta)) = m$$

$$g^{H_4(\delta, m)} \stackrel{?}{=} g^r.$$

7.3.3.2 Security proof

We will prove the security of this scheme. The intuition of security proof in the scheme is illustrated in **Figure 7.6**.

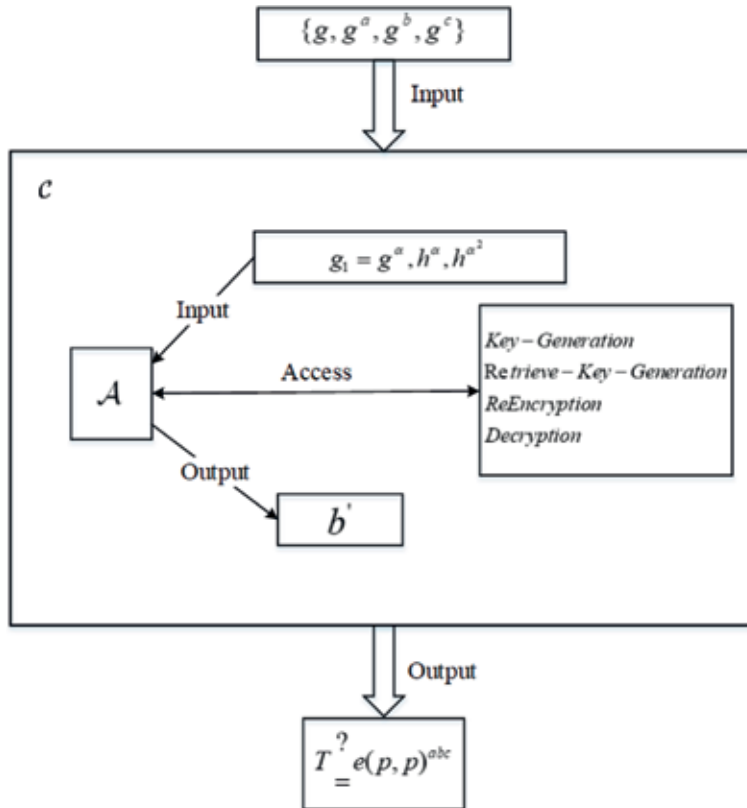


Figure 7.6.
 Intuition of security proof in IUCCS-PRE scheme.

Let A be a p.p.t. algorithm that has a non-negligible advantage ϵ in attacking the scheme. We use A in order to construct a second algorithm B which has a non-negligible advantage at solving the DBDH problem in G_1, G_T . Algorithm B accepts as input a properly distributed tuple $\langle G_1 = \langle g \rangle, g^a, g^b, g^c, T \rangle \in G_1^4 \times G_T$ and outputs 1 if $T = e(g, g)^{abc}$. We now describe algorithm B , which interacts with algorithm A via the IND-Pr-ID-CPA interface.

Oracle Queries. B simulates the random oracles H_1, H_2, H_3, H_4, H_5 as follows:

- $H_1: \{0, 1\}^* \rightarrow G_1$. On receipt of a new query for id , select $y, z \stackrel{\$}{\leftarrow} Z_q^{*c}$ and randomly set $\alpha \in \{0, 1\}$ such that $Pr[\alpha = 0] = \gamma$. If $\alpha = 0$ compute $h \stackrel{\$}{\leftarrow} (g^c)^\alpha$; if $\alpha = 1$ compute $h \stackrel{\$}{\leftarrow} g^z$. Record the tuple (id, h, y, z, α) and return h . Notice that h is correctly distributed.
- $H_2: \{0, 1\}^* \rightarrow G_1$. On a new query $X \in \{0, 1\}^*$, return $x \stackrel{\$}{\leftarrow} G_1$ and record (X, x) .
- $H_3: \{0, 1\}^* \rightarrow G_1$. On receipt of a new query $s \in \{0, 1\}^{*c}$ of the form $(id || \langle A, B, C \rangle)$, select $z \stackrel{\$}{\leftarrow} Z_q^{*c}$ and randomly set $\alpha \in \{0, 1\}$ such that $Pr[\alpha = 0] = \gamma$. Compute $H_1(id)$ to obtain the tuple $(id, h_1, y_1, z_1, \alpha_1)$. If $\alpha = \alpha_1$; then set $h = g^z$; if $\alpha = 0$ and $\alpha_1 = 1$, then set $h = (g^c)^z$; if $\alpha = 1$ and $\alpha_1 = 0$, then set $h = g^z / (g^c)^{y_1^{-1}z_1}$. Record the tuple (s, h, z, α) and return h . Notice that h is correctly distributed.

- $H_4 : G_T \times \{0, 1\}^n \rightarrow \mathcal{Z}_q^*$. On a receipt of a new query (δ, m) , return $r \in \mathcal{Z}_q^*$ and record (δ, m, r, g^r) .
- $H_5 : G_T \rightarrow \{0, 1\}^n$. On receipt of a new query δ , return $p \in \{0, 1\}^n$ and record (δ, p) .

The game between attacker A and challenger B is running as follows:

- **Select.** Choose $i \in \{0, 1\}$.
- **Setup.** B generates the scheme's master parameters $par = (G_1, H_1, H_2, H_3, H_4, H_5, g, g^a)$ and gives par to A.
- **Find.** In this stage, A can issue any things without restriction.
- **Challenge.** After finishing the find stage, A submits (id^*, m_0, m_1) . B forms the challenge ciphertext as follows:
 - Choose $\delta \in G_T, p \in \{0, 1\}^n$, and insert (δ, p) into the H_5 table. Insert (δ, m_b, g^b) into the H_4 table. Evaluate $H_1(id^*)$ and obtain the recorded value z .
 - Compute $(A^*, B^*, C^*) = \langle g^b, \delta \cdot T^z, p \oplus m_i \rangle$.
 - Select $z_1 \in \mathcal{Z}_p^*$ and insert $(id^* || \langle A^*, B^*, C^* \rangle, z_1, g^{z_1}, \alpha_1 = 1)$ into the H_3 table, and compute $S^* = (g^b)^{z_1}$.

B outputs the challenge ciphertext $c^* = (S^*, A^*, B^*, C^*) = (g^{bz}, g^b, \delta \cdot T^z, p \oplus m_i)$ to A and begins the guess stage.

Guess. In the guess stage, A is restricted from issuing the following queries:

- $(extract, id^*)$. id^* is the challenge identity.
- $(decrypt, id^*, c^*)$. c^* is the challenge ciphertext.
- Any pair of queries $\langle (rkextract, id^*, id_i), (extract, id_i) \rangle$.
- Any pair of queries $\langle (reencrypt, id^*, id_i, c^*), (extract, id_i) \rangle$.
- Any pair of queries $\langle (rkextract, id^*, id_i), (decrypt, id_i, c_i) \rangle$. $c_i = ReEnc(rk_{id^* \rightarrow id_i}, c^*)$.

Now, we let id^* be the target identity, and parse the challenge ciphertext c^* as (S^*, A^*, B^*, C^*) . In both phases, B responds to A's queries as follows:

- On $(decrypt, id, c)$, where stage $id \neq id^*$, B evaluates $H_1(id)$ to obtain (z, α) . B outputs $sk_{id} = (g^a)^z$ to A.
- On $(rkextract, id_1, id_2)$, B selects $N \in \{0, 1\}^n$ and returns the value: $rk_{id_1 \rightarrow id_2} = \langle (g^a)^{z_1} \cdot H_2(e(g^{az_1}, H_1(id_2))) || id_1 || id_2 || N \rangle, N$.
- On $(decrypt, id, c)$, if $(id, c) \neq (id^*, c^*)$, check whether c is a level-1 or level-2 ciphertext.

For a level-1 ciphertext, B parses c as (S, A, B, C) and:

- Looks up the value A in the H_4 table, to obtain the tuple (δ, m, r) . If A is not in the table, or $A = A^*$, then B returns \perp to A .
- Checks that $S = H_3(id || \langle A, B, C \rangle)^r$. If not, B returns \perp to A .
- Checks that $\delta = B/e(g^a, H_1(id)^r)$. If not, B returns \perp to A .
- Checks that $C = H_5(\delta) \oplus m$. If not, B returns \perp to A .
- Otherwise, B returns m to A .

For a level-2 ciphertext, B parses c as (A, B, C, id_{src}, N) and:

- Computes $H_1(id_{src}), H_1(id)$ to obtain $(h_1, z_1, \alpha_1), (h_2, z_2, \alpha_2)$. If both $\alpha_1 = \alpha_2 = 0$, then stop the game. If $id = id^{*\leftarrow}$, let $K = e(H_1(id), (g^a)^{z_1})$; Otherwise, let $K = e(H_1(id_{src}), (g^a)^{z_2})$.
 - Let $\delta' = B/e(A, H_2(K || id_{src} || id || N))$. Let $m' = C \oplus H_5(\delta')$, and let $r' = H_4(\delta', m')$. If $g^{r'} = A$, return m' to A .
 - If not, look up A in the H_4 table to obtain (r, δ, m) . Check that $C = H_5(\delta) \oplus m$ and $B = e(g^a, H_1(id_{src})^r) \cdot m/e(A, rk_{id_{src} \rightarrow id})$. If the answer is yes, return m . Otherwise, return \perp .
- On $(reencrypt, id_1, id_2)$, B parses c as (S, A, B, C) and:

- Computes $H = H_3(id_1 || \langle A, B, C \rangle)$, and performs the ciphertext validity check $e(g, S) = e(H, A)$. If this check does not succeed, B returns \perp to A .
- If $(id_1, A) = (id^*, A^*)$ and $c \neq c^{*\leftarrow}$, then B runs a forgery event that will be described in the next section.
- B selects $N \stackrel{\$}{\leftarrow} \{0, 1\}^n$, and:

If $id_1 = id^{*\leftarrow}$, B computes $H_1(id_1)$ to obtain $(h_1, y_1, z_1, \alpha_1)$ and evaluates $H_3(id_1 || \langle A, B, C \rangle)$ to obtain (H, z_2, α_2) . If $\alpha_2 = 0$, stop the game.

Otherwise, B selects $t \stackrel{\$}{\leftarrow} \{0, 1\}_q$ and computes $\delta' = B/(e(A, H^{y_1 t})/e(g^{y_1 t}, S))$.

Finally, B outputs

$$c_{id_2} = (A, \delta' \cdot e(A, H_2(e(H_1(id_1), (g^a)^{z_2}) || id_1 || id_2 || N)), C, id_1, N).$$

If $id_1 \neq id^{*\leftarrow}$, B outputs $c_{id_2} = ReEnc(par, rk_{id_1 \rightarrow id_2}, c)$.

At the end, A outputs its guess bit i^{\leftarrow} .

Forgery events. In this section, B operates as follows. On values $id_1, id^*, c = (S, A, B, C)$, and $c^* = (S^*, A^*, B^*, C^*)$. Let $H = H_3(id_1 || \langle A, B, C \rangle)$ and extract the H_3 table value y .

By the definition of H_3 , it holds that $Pr[H = g^{cz}] \geq \gamma$. In this case we substitute into the validity check to obtain $e(g, S) = e(g^{cz}, g^b)$ which finds $S = g^{bcz}$. B computes $S' = S^{z^{-1}} = g^{bc}$, and solves DBDH as $i_{forge} \leftarrow (e(S', g^a) \stackrel{?}{=} T)$. If all conditions are satisfied, B returns i_{forge} as the result. Lemma. If there exists a p.p.t. algorithm A^{\leftarrow} with non-negligible advantage ϵ^{\leftarrow} at distinguishing the simulation above from a “correct” simulation (in which all values are correctly formed), then we can construct an algorithm B^{\leftarrow} that solves the DBDH problem in (G_1, G_T) with non-negligible advantage.

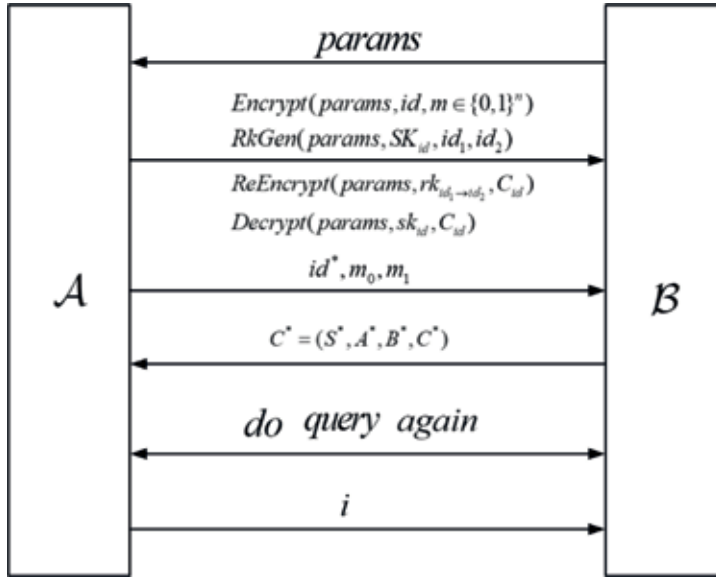


Figure 7.7.
CCA game in IBPRE scheme.

Proof. Let $P_1 = \Pr[i = i']$ when A' is given correctly distributed values as in the real attack, and $P_2 = \Pr[i = i']$ when A' is given one or more invalid re-encryption key(s) of the form described above. Let A' be an algorithm such that $|P_1 - P_2| > \nu(k)$: i.e., A' has this advantage at distinguishing the simulations. We use distinguishing A' to construct B' that solves DBDH with probability $> \nu(k)$.

We do not specify the full details of B' 's simulation but instead reuse much of the simulation above. Given a tuple (g, g^a, g^b, g^c, T) , let (g, g^a) be the public key given to A' . We contrive the H_1 oracle to (probabilistically) output g^{bz_i} (for some known z_i) on input $H_1(id_i)$ and g^{cz_1} (for known z_1) on input $H_1(id^*)$. When A' extracts a single re-encryption key, $rk_{id^* \rightarrow id_i}$, B' computes the key as $g^{az_1} \cdot W$ where

$W = H_2(T^{z_1 z_i} || id^* || id_1 || N)$. that (z_1, z_i) are randomly distributed and outside of the view of A' . Note also that when $T = e(g, g)^{abc}$, then $T^{z_1 z_i}$ is correctly distributed; otherwise $T^{z_1 z_i}$ is a random element.

When T is random, then A' has a negligible probability of issuing an oracle call on $H_2(T^{z_1 z_i} || id^* || id_1 || N)$. Conversely, when $T = e(g, g)^{abc}$, it must be the case that A' issues a call on $H_2(T^{z_1 z_i} || id^* || id_1 || N)$ with probability $> \nu(k)$. B' examines the trace to find such a call, and if it is present, outputs 1, and otherwise outputs 0. Using standard arguments, we may extend this claim to address a case where A' extracts multiple incorrectly formed re-encryption keys.

The whole game is described in **Figure 7.7**.

7.3.4 The identity-based PRE with mobile access

Zhou et al. [204] introduce an impressive proxy re-encryption scheme, referred to as identity-based proxy re-encryption version 2 (IBPRE2). The IBPRE2 successfully creates a hybrid identity-based re-encryption scheme adopting two separating mechanisms, IBBE and IBE, which require different parameters and algorithms. In the scheme, a user could authorize a proxy to

transform an encrypted data computed by an IBBE algorithm into an encrypted data which can be recognized by IBE scheme; meanwhile the proxy cannot learn any information about the plaintext nor receive any assistance from the cloud service provider. The authors introduce the framework of the IBPRE2 and define its security. An IBPRE2 mechanism exploits the re-encryption method connecting an IBBE system and an IBE system. At the system setup stage, the system initializes the parameters required by the two subsystems (IBBE and IBE) and performs the two systems in the standard way. And an IBBE user can allow a specified IBE user to decrypt a ciphertext by directly generating a re-encryption key and sending it to the proxy, which could convert an IBBE-encrypted data into an IBE-encrypted data.

Considering the IBBE and IBE's CCA security, the authors also define the same selective-identity and adaptively chosen ciphertext security of the IBPRE2 (selective CCA security) as the two schemes. Considering the re-encryption mechanism is proposed, additionally queries involving re-encryption algorithm and re-encryption key generation algorithm, as well as the decryption algorithm for re-encrypted ciphertext, are provided to the attackers in the challenge game, while the attackers still cannot distinguish the re-encrypted messages.

7.3.4.1 Concrete construction

We now describe the **IBPRE2** scheme which converts a ciphertext of the IBBE scheme introduced by [207] into a ciphertext of the tamed IBE scheme. The construction is built from prime-order bilinear groups. The **PKG** runs $(p, \mathbb{G}, \mathbb{G}_T, e) \leftarrow$

$\mathcal{G}(1^\lambda)$ to generate two cyclic groups \mathbb{G} and \mathbb{G}_T of prime order p and a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ with an input security parameter λ . IBPRE2 scheme proposed by [204] is described as follows:

- **Setup** (1^λ) . Let $l \leq |p| - 2$. Choose two coding functions $F_1 : \{0, 1\}^{l+1} \rightarrow \mathbb{G}_T$ and $F_2 : \{0, 1\}^l \rightarrow \mathbb{G}_T$. Choose a generator $g \in \mathbb{G}$ and two random elements $\alpha \in \mathbb{Z}_p$ and $h \in \mathbb{G}$. Set $(m - 1)$ as the maximal size of the broadcasting receiver set, and compute $g_1 = g^\alpha, h^\alpha, h^{\alpha^2}, \dots, h^{\alpha^m}$. For the IBBE system, define the public key

$$PK_{IBBE} = (g_1, e(g, h), h, h^\alpha, \dots, h^{\alpha^m}, H(\cdot), F_1(\cdot), F_2(\cdot))$$

and the master secret key $MSK_{IBBE} = (g, \alpha)$, where the hash function $H : \{0, 1\}^* \rightarrow \mathbb{Z}_p^*$. Then, define the public key and master secret key for the IBE system as

$$PK_{IBE} = (g_1, e(g, h), h, h^\alpha, H(\cdot), F_1(\cdot))$$

$$MSK_{IBE} = (g, \alpha).$$

Note that PK_{IBE} is a part of PK_{IBBE} and $MSK_{IBE} = MSK_{IBBE}$. We set $MSK = MSK_{IBBE} = MSK_{IBE}$.

- **KeyGen** $_{IBBE}(PK_{IBBE}, MSK, ID) = \mathbf{KeyGen}_{IBE}(PK_{IBE}, MSK, ID)$. Given an identity ID and $MSK = (g, \alpha)$, the algorithm outputs the secret key

$$SK_{ID} = SK_{IBE, ID} = SK_{IBBE, ID} = g^{\frac{1}{\alpha + H(ID)}}$$

- **Encrypt_{IBBE}** (PK_{IBBE}, S, Msg). To encrypt a message Msg with the set of identities, $S = \{ID_i\}_{i=1}^n$, where $n < m$, the algorithm computes $M = \mathcal{F}_1(\text{Msg}||0)$. Choose a random $s \in \mathbb{Z}_p^{*}$ and compute

$$C_0 = \mathcal{M} \cdot e(g, h)^s, \quad C_1 = g_1^{-s},$$

and a hash value $V = \mathcal{H}(C_0, C_1)$. Form

$$C_2 = \mathcal{H}^s \prod_{i=1}^n (\alpha + H(ID_i))^{(\alpha+V)}$$

Output $CT_S = (C_0, C_1, C_2)$.

The IBE encryption **Encrypt_{IBE}** (PK_{IBE}, ID, Msg) is defined the same as the proposed IBE scheme except that the message to be encrypted is $M = \mathcal{F}_1(\text{Msg}||0)$.

- **RKGen** ($PK_{IBBE}, SK_{ID_j}, S, ID$). This step would convert CT_S into a ciphertext for ID . Given the public key PK_{IBBE} , a set of identities $S = \{ID_i\}_{i=1}^n$ with $n < m$, and a secret key SK_{ID_j} for $ID_j \in S$, the re-encryption key generation algorithm chooses a random $k \in \{0, 1\}^l$, computes $F_2(k)$, and obtains $R = \text{Encrypt}_{\text{IBE}}^{\leftarrow}(PK_{IBE}, ID, k)$, where **Encrypt_{IBE}** is identical to **Encrypt_{IBE}** except that the message to be encrypted is $M = \mathcal{F}_1(k||1)$. Finally, output the re-encryption key as

$$RK_{S \rightarrow ID} = (SK_{ID_j} \cdot \mathcal{F}_2(k), R).$$

ReEnc ($PK_{IBE}, RK_{S \rightarrow ID}, CT_S$). Parse $RK_{S \rightarrow ID} = (d, R)$ that is given by the user $ID_j \in S$, to re-encrypt $CT_S = (C_0, C_1, C_2)$. And then compute $V = \mathcal{H}(C_0, C_1)$ and check if

$$e(g_1, C_2) = e\left(C_1^{-1}, h \prod_{i=1}^n (\alpha + H(ID_i))^{(\alpha+V)}\right)$$

holds. If not, output a false symbol \perp . Otherwise, select a random $t \in \mathbb{Z}_p^{*}$ and compute

$$d_1 = \mathcal{A} \cdot g_1^t, \quad d_2 = \mathcal{H} \prod_{i=1}^n (\alpha + H(ID_i))^{(\alpha+V)}$$

$$d_3 = (d_2)^t = \mathcal{H}^t \prod_{i=1}^n (\alpha + H(ID_i))^{(\alpha+V)}$$

Also compute

$$K = \mathcal{H}^{\frac{1}{t}} \left(\prod_{i=1, i \neq j}^n (\alpha + H(ID_i))^{(\alpha+V)} \prod_{i=1, i=j}^n H(ID_i)V \right).$$

Then, output $CT_{ID} = (CT_S, R, d_1, d_2, d_3, K)$.

- **Decrypt** (PK, CT, SK_{ID}). Given the public key $PK = PK_{IBBE} \supset PK_{IBE}$, a ciphertext CT , and a secret key SK_{ID} for identity ID , the algorithm runs as follows.
 - If CT is a ciphertext of the IBE system, the algorithm performs the same executions of the decryption of the IBE scheme to output $M = \mathcal{F}_1(\text{Msg}||0)$. Then recover Msg by removing the last bit from $F_1^{-1}(M)$.
 - If CT is a ciphertext of the IBBE system, the algorithm performs the same executions of the decryption of the CCA-secure IBBE scheme described by [207] to output $M = \mathcal{F}_1(\text{Msg}||0)$. Then recover Msg by removing the last bit from $F_1^{-1}(M)$.

- If CT is a re-encrypted ciphertext, let $CT = CT_{ID} = (C_0, C_1, C_2, R, d_1, d_2, d_3, K)$ for identity ID . Suppose that CT_{ID} is re-encrypted by the re-encryption key that is generated by the user $ID_j \in S$. Check if the following equations holds:
 - $e(g_1, C_2) = e\left(C_1^{-1}, h^{\prod_{i=1}^n (\alpha + H(ID_i))(\alpha + V)}\right)$ and
 - $e(d_1 F_2^{-1}(k), d_2) = e(g_1, d_3) \cdot e(g_1, K) \cdot e(g, h)^{\prod_{i=1, i \neq j}^n H(ID_i)V}$, where $V = H(C_0, C_1)$ and $k = \text{Decrypt}(PK_{IBE}, SK_{ID}, R)$.

We define Decrypt^* the same as Decrypt except that it outputs k if the decrypted message ends with bit “1” and outputs \perp if it ends with bit “0.” If any one of the equations does not hold, output \perp ; otherwise, compute

$$M^* = \left(e(C_1, K) \cdot e(d_1 F_2^{-1}(k), C_2) \cdot e(C_1, d_3) \right)^{\frac{1}{\prod_{i=1, i \neq j}^n H(ID_i)V}}$$

and $M = C_0 / M^*$. Compute $F_1^{-1}(M) = \text{Msg} \| B$. If $B = 0$, output message Msg ; otherwise, output \perp .

Remark 7.1. In the encryption algorithm, we append an additional bit (0 or 1) onto the end of the message in order to distinguish the encryption of a message and the encryption of a random value used in the re-encryption key generation, so that in the security proof, the adversary is unable to take the challenge ciphertext as R in the re-encrypted ciphertext to query the decryption oracle and obtain some useful information.

Remark 7.2. In the re-encryption algorithm, the proxy computes the component K in order to reduce the burden of decryption for re-encrypted ciphertext on the delegatee’s side. Since in the original IBBE scheme this component can be publicly computed, it will not damage the security of the message encrypted in the ciphertext.

Correctness 1. If CT is a ciphertext of IBBE or IBE, the correctness follows the correctness of the IBBE proposed by [208] and the tamed IBE, respectively.

If CT is a properly re-encrypted ciphertext, the correctness follows from the following validations:

$$\begin{aligned} e(d_1 F_1^{-1}(k), d_2) &= e\left(g^{\frac{1}{\alpha + H(ID_j)}} F_2(k) \cdot g_1^t \cdot F_2^{-1}(k), h^{\prod_{i=1}^n (\alpha + H(ID_i))(\alpha + V)}\right) \\ &= e\left(g_1^t, h^{\prod_{i=1}^n (\alpha + H(ID_i))(\alpha + V)}\right) \cdot e(g, h^{\Delta_j(\alpha)}) \\ &= e\left(g_1, h^{\prod_{i=1}^n (\alpha + H(ID_i))(\alpha + V)}\right) \cdot e\left(g^\alpha, h^{\frac{1}{\alpha} \left(\Delta_j(\alpha) - \prod_{i=1, i \neq j}^n H(ID_i)V\right)}\right) \\ &\quad \cdot e\left(g, h^{\prod_{i=1, i \neq j}^n H(ID_i)V}\right) \\ &= e(g_1, d_3) \cdot e(g_1, K) \cdot e(g, h)^{\prod_{i=1, i \neq j}^n H(ID_i)V} \end{aligned}$$

where $\Delta_j(\alpha) = \prod_{i=1, i \neq j}^n (\alpha + H(ID_i))(\alpha + V)$. If the checks pass, observe that

$$\begin{aligned}
 M' &= e(C_1, K) \cdot e(d_1 F_2^{-1}(k), C_2) \cdot e(C_1, d_3) \\
 &= e \left(g^{-\alpha}, h^{\frac{1}{\alpha} \left(\Delta_j(\alpha) - \prod_{i=1, i \neq j}^n H(ID_i) V \right)} \right) \\
 &= e \left(g^{\frac{1}{\alpha + H(ID_j)}} \cdot g_1^t, h^s \prod_{i=1}^n (\alpha + h(ID_i)) (\alpha + V) \right), \\
 &= e \left(g, h^{-s \left(\Delta_j(\alpha) - \prod_{i=1, i \neq j}^n H(ID_i) V \right)} \right) \cdot e(g, h^{s \Delta_j(\alpha)}) \\
 &= e(g, h)^{s \prod_{i=1, i \neq j}^n H(ID_i) V}
 \end{aligned}$$

and $M' = (M'')^{\frac{1}{\prod_{i=1, i \neq j}^n H(ID_i) V}} = e(g, h)^s$. It then follows that $M = C_0 / M'$. The game between \mathcal{A} and challenger is introduced in **Figure 7.8**.

7.3.4.2 Security model

We now show the selective CCA security of the IBPRE2 scheme. The CCA security of the IBBE scheme introduced by [207] can be easily achieved by following the [209] approach. Similar with the tamed CCA-secure IBE, an additional “identity” that is only used for ciphertext validity test is also introduced into the

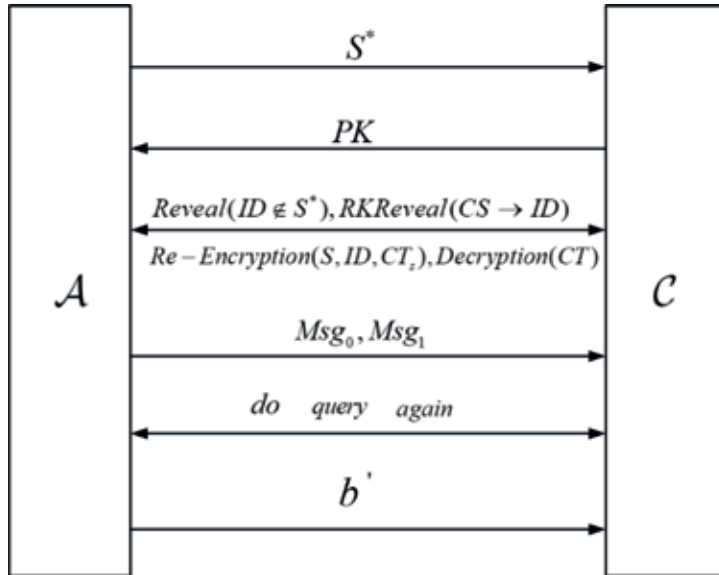


Figure 7.8. Security game in IBPRE2 scheme.

secret keys and ciphertexts of the IBBE scheme. We can use the same technique in the proof of Theorem 1 to prove the CCA security of the IBBE scheme.

The following theorem shows that the proposed IBPRE2 scheme is selectively secure against chosen-ciphertext attack if the proposed IBE scheme and the IBBE scheme introduced by [207] are selectively CCA-secure.

Theorem 7.3. *If there exists an adversary \mathcal{A} which has an advantage $Adv_{\mathcal{A}}^{IBPRE2}$ in breaking the CCA security of the IBPRE2 system, then we can construct an algorithm \mathcal{B} that breaks the CCA security of the IBBE scheme proposed by [208] and the proposed IBE scheme with advantage ϵ_1 and ϵ_2 , respectively, satisfying*

$$\epsilon_1 + \epsilon_2 \geq Adv_{\mathcal{A}}^{IBPRE2} / e(1 + q_K),$$

where e is the base of natural logarithm and q_K is the maximum number of \mathcal{A} 's secret key queries.

Proof 7.4. Given an adversary \mathcal{A} breaking the IBPRE2 scheme, we build an algorithm \mathcal{B} which interacts with \mathcal{A} to break the selective CCA security of the IBBE scheme introduced by [207] or the proposed IBE scheme. The adversary is allowed to query the re-encryption for identity sets of its choice. To respond to \mathcal{A} 's **RKReveal**($S \rightarrow ID$) queries, algorithm \mathcal{B} has to call the key generation oracle of the IBBE scheme and then applies the output keys to run the **RKGen** algorithm to obtain the requested re-encryption keys. In an unwanted case, \mathcal{B} is queried by the adversary to generate re-encryption key for the challenge identity set S^* . Unfortunately, algorithm \mathcal{B} cannot respond by giving the queried key. This is because that the security definition of IBBE forbids key queries for the challenge identity set and \mathcal{B} cannot obtain the correct IBBE secret key. We let \mathcal{B} generate real keys with a determined probability and output random keys with the complementary probability. To this end, we create a table $Z = (\beta \in \{0, 1\}, S, ID, RK)$ to record queried keys, and let \mathcal{B} output a real key when $\beta = 1$ and output a random key when $\beta = 0$. The real re-encryption keys and random ones need to be proved indistinguishable in \mathcal{A} 's view.

The algorithm \mathcal{B} first runs the setup algorithm of the IBBE scheme introduced by [207] with input m to get public key PK' . It also picks two encoding functions F_1 and F_2 to form public key $PK = (PK', F_1, F_2)$. We no longer discriminate PK_{IBBE} and PK_{IBE} since the former includes the latter. The algorithm \mathcal{B} also maintains a table $Z = (\beta, S, ID, RK)$ initialized to be empty. Algorithm \mathcal{B} interacts with adversary \mathcal{A} as follows:

- **Init.** The adversary \mathcal{A} outputs an identity S^* to be attacked.
- **Setup.** Algorithm \mathcal{B} gives the public key PK to \mathcal{A} .
- **Phase 1.** The adversary \mathcal{A} makes the following queries:
 - **Reveal**($ID \notin S^*$). Algorithm \mathcal{B} flips a random $\beta \in \{0, 1\}$, so that $Pr[\beta = 1]$ with a probability θ which will be determined later. If $\beta = 0$, or $(0, *, ID, *)$ exists on the table, \mathcal{B} aborts the simulation. Otherwise, it runs the key generation algorithm of the IBBE scheme to create a secret key for ID . \mathcal{B} returns this key to \mathcal{A} and records $(\beta, *, ID, \perp)$ on table Z .
 - **RKReveal**($S \rightarrow ID$). Algorithm \mathcal{B} flips a random $\beta \in \{0, 1\}$. If $\beta = 1$ and $S = S^*$, or $(1, S, *, \perp)$ exists on the table, algorithm \mathcal{B} first calls the key generation algorithm of the IBBE scheme to output a secret key for an

identity in S and then uses this key to generate a re-encryption key RK with S to ID . \mathcal{B} sends RK to \mathcal{A} . Otherwise, \mathcal{B} returns a random re-encryption key $RK = (D, \text{Encrypt}'_{\text{IBBE}}(PK, ID, z))$, where D is a random element of \mathbb{G} and z is a random string. Finally, \mathcal{B} writes (β, S, ID, RK) on table Z .

- **Re-Encryption**(S, ID, CT_S). If $(*, S, ID, *)$ does not exist on the table, which means that \mathcal{A} has not queried the re-encryption key from S to ID , algorithm \mathcal{B} runs the **RKGen** algorithm to obtain a re-encryption key $RK_{S \rightarrow ID}$ and writes $(*, S, ID, RK_{S \rightarrow ID})$ on the table. Then, \mathcal{B} runs the **ReEnc** algorithm using $RK_{S \rightarrow ID}$ to re-encrypt CT_S and returns the re-encrypted ciphertext to \mathcal{A} .
- **Decryption**(CT). If CT is a regular ciphertext for S , algorithm \mathcal{B} forwards it to the decryption oracle of the IBBE scheme and returns the plaintext if it ends with bit "0." If CT is a re-encrypted ciphertext for an identity ID , let $CT_{ID} = (CT_S, R, d_1, d_2, d_3, K)$. If there is a record $(*, S, ID, RK')$ on the table, \mathcal{B} examines whether CT_{ID} is the real ciphertext re-encrypted by $RK'_{S \rightarrow ID}$. To do so, parsing $RK'_{ID \rightarrow S} = (d', R')$, \mathcal{B} checks if the following equations holds

$$e(d_1, d_2) = e(d', h\left(\prod_{i=1}^n (\alpha + H(ID_i))\right)^{(\alpha + V)}) \cdot e(g_1, d_3),$$

and $R' = R$. If so, algorithm \mathcal{B} calls the decryption oracle of the IBBE scheme to decrypt CT_S and returns the plaintext if it ends with a bit "0." Otherwise, algorithm \mathcal{B} calls the decryption oracle of the IBE scheme to decrypt R and obtains k if $k \neq \perp$ is the output of the decryption oracle. Then, it computes M as in the decryption algorithm of the proposed scheme. Set $Msg \parallel B = \mathcal{E}_1^{-1}(M)$. If $B = \emptyset$, output Msg to \mathcal{A} ; otherwise, output \perp .

- **Challenge**. The adversary \mathcal{A} outputs two messages Msg_0 and Msg_1 . If there is a record $(1, S^*, ID', RK')$ for any ID' and RK' on the table, \mathcal{B} aborts. Otherwise, \mathcal{B} gives the challenge $(S^*, Msg_0 \parallel 0, Msg_1 \parallel 0)$ to the encryption oracle of the IBBE scheme and returns the challenge ciphertext to \mathcal{A} .
- **Phase 2**. Phase 1 is repeated except the not allowed queries described in Definition 1.
- **Guess**. The adversary \mathcal{A} outputs its guess $b' \in \{0, 1\}$ and \mathcal{B} outputs the same bit.

During the whole game above, \mathcal{B} perfectly simulates the real scheme if it does not abort and always outputs real keys. Let q_K denote the number of \mathcal{A} secret key queries. Then \mathcal{B} does not abort in phase 1 or 2 with probability θ^{q_K} and does not abort in the challenge phase with probability $1 - \theta$. Therefore, \mathcal{B} does not abort in the whole game with the probability $\theta^{q_K} (1 - \theta)$. This value is maximized at $1 - \frac{1}{q_K + 1}$, from which we can obtain the probability that \mathcal{B} does not abort is at least $1/e(1 + q_K)$.

It remains to show that no adversary can distinguish the real keys from random keys. The random re-encryption key is $(D, \text{Encrypt}'_{\text{IBBE}}(PK, ID, z))$, where $D \stackrel{R}{\in} \mathbb{G}_T$ and $z \in \{0, 1\}^l$; the real re-encryption key is $(SK_{ID}, F_2(k), \text{Encrypt}'_{\text{IBBE}}(PK, ID, k))$,

where SK_{ID_i} is randomized by $F_2(k)$. The distinguishability of these two keys is equivalent to distinguishing the IBE ciphertext, which breaks the security of the IBE scheme. Therefore, \mathcal{A} successfully distinguishes the real keys from random keys with a negligible probability.

To sum up, algorithm \mathcal{B} can use the advantage $Adv_{\mathcal{A}}^{IBPRE2}$ of \mathcal{A} attacking IBPRE2 system to break the security of the IBBE and the IBE systems with advantage ϵ_1 and ϵ_2 , respectively, satisfying $\epsilon_1 + \epsilon_2 \geq Adv_{\mathcal{A}}^{IBPRE2}/e(1 + q_K)$.

The intuition of security proof is described in **Figure 7.9**.

7.3.5 Conditional identity-based PRE with mobile access

Xu et al. [205] introduce a promising and more flexible mechanism of conditional identity-based broadcast PRE (CIBPRE), involving the advantages of IPRE, CPRE, and BPRE. In the system, the environment of the CIBPRE will be prepared by an honest key generation center (KGC), and the system parameters required by CIBPRE and private keys for users will be generated either. A user can broadcast an encrypted message securely and merely those receivers who possess a certain identity and file-sharing conditions could extract the plaintext. Moreover, if the sender would like to authorize other users to access the message with the same condition, the sender could grant a re-encryption key associated with the condition, which would be used to transform the ciphertext, to the proxy. And the proxy will choose some parameters independent of prior receivers when generating the key. And the original encrypted data matching the condition could be converted to a new

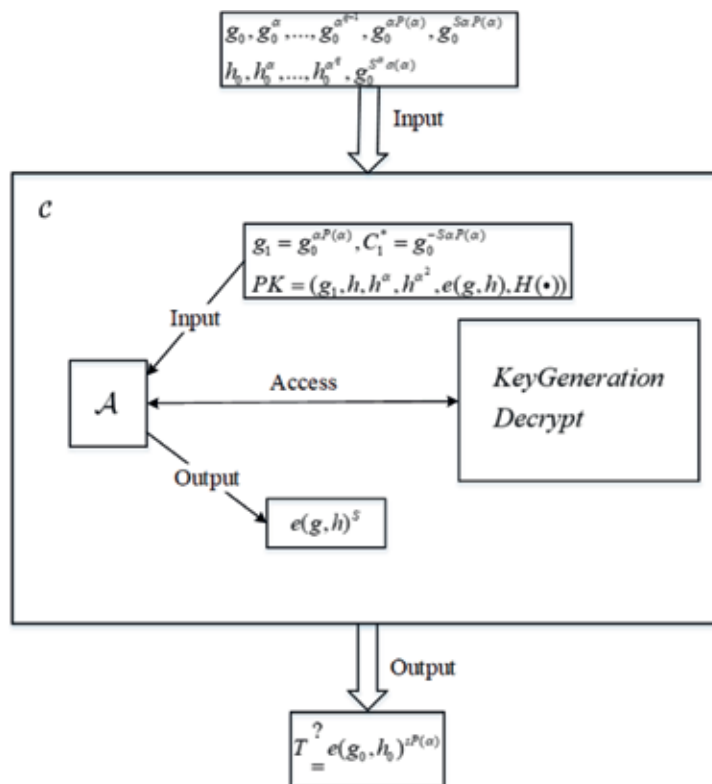


Figure 7.9.
 Intuition of security proof in IBPRE2 scheme.

ciphertext for the new set of users by the proxy. Except for the original authorized receivers, the new authorized user could transform the new computed ciphertext into plaintext with their private keys with CIBPRE. It is worthy to note that the initial ciphertext are securely stored in the cloud service provider and the sender can only grant a small-size key to the proxy to authorize new set of user without downloading the ciphertext from the proxy again. So we can observe from the above example that the CIBPRE with those features could impressively guarantee the security of files especially there are multiple users who could access those files.

7.3.5.1 Concrete construction

Referring to the concept of CIBPRE, both the initial CIBPRE ciphertext and the re-encrypted CIBPRE ciphertext are the IBBE ciphertext. But it is different with an IBBE scheme that CIBPRE would provide several algorithms to transform an IBBE ciphertext (corresponding to an initial CIBPRE ciphertext) into another IBBE ciphertext (corresponding to a re-encrypted CIBPRE ciphertext). Moreover, the transformation is correct if it satisfies the consistencies defined by CIBPRE. Therefore, in order to construct a CIBPRE scheme, we refer to the D07 scheme which was reviewed in Section 2. Compared with the D07 scheme introduced by [208], the proposed CIBPRE scheme associates a D07 IBBE ciphertext with a new part to generate an initial CIBPRE ciphertext. This new part will be used to realize the capability “conditional” of CIBPRE. In addition, it provides some new algorithms, which are, respectively, to generate a re-encryption key, re-encrypt an initial CIBPRE ciphertext, and decrypt a re-encrypted CIBPRE ciphertext. The decryption of an initial CIBPRE ciphertext is the same with the D07 scheme. The proposed CIBPRE scheme is as follows:

- **Setup_{PRE}**(λ, N). Given a security parameter $\lambda \in \mathbb{N}$ and value N (the maximum number of receivers in each encryption), this algorithm probabilistically constructs a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, where \mathbb{G} and \mathbb{G}_T are two multiplicative groups with prime order p ($|\mathbb{G}| = \lambda$), randomly chooses $(g, h, u, t) \in \mathbb{G}^4$ and $\gamma \in \mathbb{Z}_p^*$, chooses two cryptographic hash functions $H : \{0, 1\}^* \rightarrow \mathbb{Z}_p^*$ and $H' : \mathbb{G}_T \rightarrow \mathbb{G}$, finally outputs a master public key $\mathbf{PK}_{PRE} = (p, \mathbb{G}, \mathbb{G}_T, e, w, v, h, h^\gamma, \dots, h^{\gamma^N}, u, u^\gamma, \dots, u^{\gamma^N}, t, t^\gamma, \dots, t^{\gamma^N}, H, H')$ and a master secret key $\mathbf{MK}_{PRE} = (g, \gamma)$, where $w = g^\gamma$ and $v = e(g, h)$.
- **Extract_{PRE}**(\mathbf{MK}_{PRE}, ID). Given \mathbf{MK}_{PRE} and an identity ID , this algorithm outputs the private key $SK_{PRE}^{ID} = g^{\frac{1}{\gamma + H(ID)}}$.
- **Enc_{PRE}**($\mathbf{PK}_{PRE}, S, m, \alpha$). Given \mathbf{PK}_{PRE} , a set S of some identities (where $|S| \leq N$), a plaintext $m \in \mathbb{G}_T$, and a condition $\alpha \in \mathbb{Z}_p^*$, this algorithm randomly picks $k \in \mathbb{Z}_p^*$ and outputs an initial CIBPRE ciphertext $C = (c_1, c_2, c_3, c_4)$, where

$$c_1 = w^{-k}, c_2 = h^{k \cdot \prod_{ID_i \in S} (\gamma + H(ID_i))},$$

$$c_3 = v^k \cdot m, c_4 = (u \cdot t^\alpha)^k \cdot \prod_{ID_i \in S} \left(\frac{\gamma + H(ID_i)}{H(ID_i)} \right).$$
- **RKExtract_{PRE}**($\mathbf{PK}_{PRE}, ID, SK_{PRE}^{ID}, S', \alpha$). Given \mathbf{PK}_{PRE} , an identity ID and its private key SK_{PRE}^{ID} , a set S' of some identities (where $|S'| \leq N$), and a condition

$\alpha \in \mathbb{Z}_p^*$, this algorithm randomly picks $(k', s) \in \mathbb{Z}_p^{*2}$ and outputs a re-encryption key $d_{ID \rightarrow S'|\alpha} = (d_1, d_2, d_3, d_4)$, where

$$d_1 = w^{-k'}, d_2 = h^{k' \cdot \prod_{ID_i \in S'} (\gamma + H(ID_i))},$$

$$d_3 = H'(v^{k'}) \cdot h^s, d_4 = SK_{PRE}^{ID} \cdot (u \cdot t^\alpha)^{\frac{s}{H(ID)}}$$

- **ReEnc_{PRE}**($\mathbf{PK}_{PRE}, d_{ID \rightarrow S'|\alpha}, C, S$). Given \mathbf{PK}_{PRE} , a re-encryption key $d_{ID \rightarrow S'|\alpha} = (d_1, d_2, d_3, d_4)$, an initial CIBPRE ciphertext $C = (c_1, c_2, c_3, c_4)$, and a set S of some identities (where $|S| \in N$), this algorithm outputs a re-encrypted CIBPRE ciphertext $\tilde{C} = (\tilde{c}_1, \tilde{c}_2, \tilde{c}_3, \tilde{c}_4, \tilde{c}_5)$, where $\tilde{c}_1 = d_1, \tilde{c}_2 = d_2, \tilde{c}_3 = d_3, \tilde{c}_4 = c_4$ and

$$\tilde{c}_5 = c_3 \cdot (e(c_1, h^{A_\gamma(ID, S)})) \cdot e(d_4, c_2)^{\frac{-1}{\prod_{ID_i \in S \wedge ID_i = ID} H(ID_i)}},$$

with

$$A_\gamma(ID, S) = \gamma^{-1} \cdot \left(\prod_{ID_i \in S \wedge ID_i = ID} (\gamma + H(ID_i)) - \prod_{ID_i \in S \wedge ID_i = ID} H(ID_i) \right).$$

- **Dec-1_{PRE}**($\mathbf{PK}_{PRE}, ID, SK_{PRE}^{ID}, C, S$). Given \mathbf{PK}_{PRE} , an identity ID and its private key SK_{PRE}^{ID} , an initial CIBPRE ciphertext $C = (c_1, c_2, c_3, c_4)$, and a set S of some identities (where $|S| \leq N$), this algorithm computes

$$K = (e(c_1, h^{A_\gamma(ID, S)}) \cdot e(SK_{PRE}^{ID}, c_2))^{\frac{1}{\prod_{ID_i \in S \wedge ID_i = ID} H(ID_i)}},$$

with

$$A_\gamma(ID, S) = \gamma^{-1} \cdot \left(\prod_{ID_i \in S \wedge ID_i = ID} (\gamma + H(ID_i)) - \prod_{ID_i \in S \wedge ID_i = ID} H(ID_i) \right).$$

- **Dec-2_{PRE}**($\mathbf{PK}_{PRE}, ID', SK_{PRE}^{ID'}, \bar{C}, S'$). Given \mathbf{PK}_{PRE} , an identity ID' and its private key $SK_{PRE}^{ID'}$, an initial CIBPRE ciphertext $\bar{C} = (\bar{c}_1, \bar{c}_2, \bar{c}_3, \bar{c}_4, \bar{c}_5)$, and a set S' of some identities (where $|S'| \leq N$), this algorithm computes

$$K = (e(\bar{c}_1, h^{A_\gamma(ID', S')}) \cdot e(SK_{PRE}^{ID'}, \bar{c}_2))^{\frac{1}{\prod_{ID_i \in S' \wedge ID_i = ID'} H(ID_i)}},$$

with

$$A_\gamma(ID', S') = \gamma^{-1} \cdot \left(\prod_{ID_i \in S' \wedge ID_i = ID'} (\gamma + H(ID_i)) - \prod_{ID_i \in S' \wedge ID_i = ID'} H(ID_i) \right),$$

computes $K' = \frac{\bar{c}_5}{H(K)}$, and finally outputs plaintext $m = \bar{c}_5 \cdot e(K', \bar{c}_4)$.

The consistency of the proposed CIBPRE scheme is guaranteed by the Theorems 4 and 5.

Theorem 7.4. For any initial CIBPRE ciphertext $C = \text{Enc}_{\text{PRE}}(\text{PK}_{\text{PRE}}, S, m, \alpha) \leftarrow$ and any private key $SK_{\text{PRE}}^{\text{ID}} = \text{Extract}_{\text{PRE}}(\text{MK}_{\text{PRE}}, \text{ID})$, if $\text{ID} \in S$, then algorithm $\text{Dec} - 1_{\text{PRE}}(\text{PK}_{\text{PRE}}, \text{ID}, SK_{\text{PRE}}^{\text{ID}}, C, S)$ always outputs the plaintext m .

Proof 7.5. For any initial CIBPRE ciphertext $C = \text{Enc}_{\text{PRE}}(\text{PK}_{\text{PRE}}, S, m, \alpha) \leftarrow$ and any private key $SK_{\text{PRE}}^{\text{ID}} = \text{Extract}_{\text{PRE}}(\text{MK}_{\text{PRE}}, \text{ID})$, we have a $C = (c_1, c_2, c_3, c_4)$ and $SK_{\text{PRE}}^{\text{ID}} = g^{\frac{1}{\gamma + H(\text{ID})}}$ without loss of generality, where $c_1 = w^{-k}$, $c_2 = h^k \cdot \prod_{\text{ID}_i \in S} (\gamma + H(\text{ID}_i)) \leftarrow$, $c_3 = v^k \cdot m$ and $c_4 = u^k \cdot \prod_{\text{ID}_i \in S} (\frac{\gamma + H(\text{ID}_i)}{H(\text{ID}_i)})$. We next prove $\text{Dec} - 1_{\text{PRE}}(\text{PK}_{\text{PRE}}, \text{ID}, SK_{\text{PRE}}^{\text{ID}}, C, S) = m$ if $\text{ID} \in S$. When $\text{ID} \in S$, we have

$$\begin{aligned} & (e(c_1, h^{\Delta_\gamma(\text{ID}, S)}) \cdot e(SK_{\text{PRE}}^{\text{ID}}, c_2))^{\frac{1}{\prod_{\text{ID}_i \in S \wedge \text{ID}_i \neq \text{ID}} H(\text{ID}_i)}} \leftarrow \\ &= (e(g^{-k \cdot \gamma}, h^{\Delta_\gamma(\text{ID}, S)}) \cdot e(g^{\frac{1}{\gamma + H(\text{ID})}}, h^k \cdot \prod_{\text{ID}_i \in S} (\gamma + H(\text{ID}_i)))^{\frac{1}{\prod_{\text{ID}_i \in S \wedge \text{ID}_i \neq \text{ID}} H(\text{ID}_i)}} \leftarrow \\ &= (e(g^{-k \cdot \gamma}, h^{\Delta_\gamma(\text{ID}, S)}) \cdot e(g, h)^k \cdot \prod_{\text{ID}_i \in S \wedge \text{ID}_i \neq \text{ID}} (\gamma + H(\text{ID}_i)))^{\frac{1}{\prod_{\text{ID}_i \in S \wedge \text{ID}_i \neq \text{ID}} H(\text{ID}_i)}} \leftarrow \\ &= (e(g, h)^{\leftarrow \prod_{\text{ID}_i \in S \wedge \text{ID}_i \neq \text{ID}} (H(\text{ID}_i))})^{\frac{1}{\prod_{\text{ID}_i \in S \wedge \text{ID}_i \neq \text{ID}} H(\text{ID}_i)}} = v^k, \end{aligned}$$

with

$$\Delta_\gamma(\text{ID}, S) = \gamma^{-1} \cdot \left(\prod_{\text{ID}_i \in S \wedge \text{ID}_i \neq \text{ID}} (\gamma + H(\text{ID}_i)) \leftarrow \prod_{\text{ID}_i \in S \wedge \text{ID}_i \neq \text{ID}} H(\text{ID}_i) \right).$$

Since $m = c_3/v^k$, we proved that $\text{Dec} - 1_{\text{PRE}}(\text{PK}_{\text{PRE}}, \text{ID}, SK_{\text{PRE}}^{\text{ID}}, C, S) = m$ if $\text{ID} \in S$.

Theorem 7.5. For any re-encrypted CIBPRE ciphertext $\tilde{C} = \text{ReEnc}_{\text{PRE}}(\text{PK}_{\text{PRE}}, d_{\text{ID} \rightarrow S' | \alpha'}, C, S)$ and any private key $SK_{\text{PRE}}^{\text{ID}' \leftarrow} = \text{Extract}_{\text{PRE}}(\text{MK}_{\text{PRE}}, \text{ID}' \leftarrow)$, where $d_{\text{ID} \rightarrow S' | \alpha'} = \text{RKExtract}_{\text{PRE}}(\text{PK}_{\text{PRE}}, \text{ID}, SK_{\text{PRE}}^{\text{ID}}, S', \alpha')$, $C = \text{Enc}_{\text{PRE}}(\text{PK}_{\text{PRE}}, S, m, \alpha)$, and $SK_{\text{PRE}}^{\text{ID}' \leftarrow} = \text{Extract}_{\text{PRE}}(\text{MK}_{\text{PRE}}, \text{ID})$, if $\text{ID} \in S \wedge \alpha = \alpha' \wedge \text{ID}' \in S'$, then algorithm $\text{Dec} - 2_{\text{PRE}}(\text{PK}_{\text{PRE}}, \text{ID}', SK_{\text{PRE}}^{\text{ID}' \leftarrow}, C, S)$ always outputs the plaintext m .

Theorem 5 is proved by [205]. The whole game is described in **Figure 7.10**.

7.3.5.2 Security proof

In addition to the DBDH assumption, the security of our proposed CIBPRE scheme also relies on the IND-sID-CPA security of the D07 scheme. In other words, the IND-sID-CPA security of the proposed CIBPRE scheme will be reduced to the DBDH assumption and the IND-sID-CPA security of the D07 scheme. Let \perp denote to abort an algorithm. The proof is as follows.

Theorem 7.6. Suppose that the DBDH assumption holds. Since the D07 scheme is IND-sID-CPA-secure in the RO model, the proposed CIBPRE scheme is IND-sID-CPA-secure in the RO model.

Proof 7.6. Suppose that there is an adversary \mathcal{A} having advantage $\text{Adv}_{\text{CIBPRE}, \mathcal{A}}^{\text{IND-sID-CPA}}$ to break the IND-sID-CPA security of the proposed CIBPRE scheme. To prove this theorem, we will construct an adversary \mathcal{B} , who leverages adversary \mathcal{A} to break the IND-sID-CPA security of the D07 scheme with advantage $\text{Adv}_{\text{D07}, \mathcal{B}}^{\text{IND-sID-CPA}}$. Moreover, we will prove that

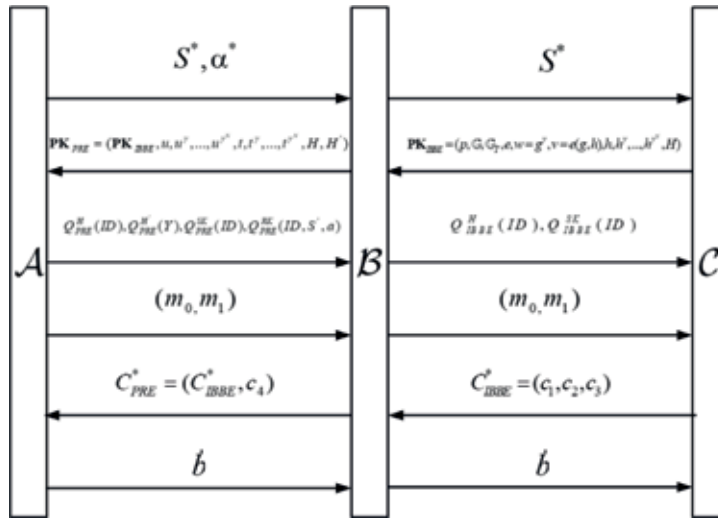


Figure 7.10.
 IND-sID-CPA game in CIBPRE scheme.

$Adv_{D07,B}^{IND-sID-CPA} = Adv_{CIBPRE,A}^{IND-sID-CPA} (1 - T \cdot Adv_{DBDH,A}^{IND} - T \cdot Adv_{D07,A}^{IND-sID-CPA})$ holds, where value T will be defined latterly.

Let \mathcal{C} be the challenger defined in D07's IND-sID-CPA security. Next, we will describe a game among adversary \mathcal{A} , adversary \mathcal{B} , and challenger \mathcal{C} . In this game, challenger \mathcal{C} tests adversary \mathcal{B} 's ability in breaking D07's IND-sID-CPA security; adversary \mathcal{B} also serves as a challenger to test adversary \mathcal{A} 's ability in breaking the IND-sID-CPA security of the proposed CIBPRE scheme. So in this game, adversary \mathcal{B} aims to utilize adversary \mathcal{A} 's ability to break the IND-sID-CPA security of D07. The details of this game are as follows:

- **InitializationPhase:** Adversary \mathcal{A} chooses a set S^* of challenge identities (where $|S^*| \leq N$) and a challenge condition α^* that he wants to attack and sends S^* and α^* to adversary \mathcal{B} ; then adversary \mathcal{B} forwards set S^* to challenger \mathcal{C} . In addition, adversary \mathcal{B} prepares the following two tables:
 - \mathcal{L}_{SK} to store the identities which private keys have been queried by adversary \mathcal{A}
 - \mathcal{L}_{RK} with columns (*Identity, ReceiverSet, Re-EncryptionKey, Condition*) to store the related information of the re-encryption keys queried by adversary \mathcal{A}
- **SetupPhase:** This phase consists of two parts: Firstly challenger \mathcal{C} generates D07's master public and secret keys for adversary \mathcal{B} ; secondly adversary \mathcal{B} generates the proposed CIBPRE scheme's master public key for adversary \mathcal{A} . In these two parts, three hash functions are modeled as random oracles, respectively, by challenger \mathcal{C} and adversary \mathcal{B} . So above master public keys do not contain any hash function in this proof. The details are as follows:
 - Challenger \mathcal{C} runs D07's algorithm $\text{Setup}_{IBBE}(\lambda, N)$ to generate a master public key PK_{IBBE} and a master secret key MK_{IBBE} , where $\text{PK}_{IBBE} = (p, \mathbb{G}, \mathbb{G}_T, e, w = g^\gamma, v = e(g, h), h, h', \dots, h^N, H)$ and $\text{MK}_{IBBE} = (g, \gamma)$. Since

the hash function H is modeled as a random oracle in this proof, challenger \mathcal{C} sends the reduced master public key $\mathbf{PK}_{IBBE} = (p, \mathbb{G}, \mathbb{G}_T, e, w = g^r, v = e(g, h), h, h^r, \dots, h^{r^N})$ to adversary \mathcal{B} . And challenger \mathcal{C} provides hash query service $Q_{IBBE}^H(ID)$ for adversary \mathcal{B} and models the hash function H as a random oracle. He prepares a table \mathcal{L}_H for $Q_{IBBE}^H(ID)$ with columns (*Identity*, *HashValue*) to store the queried identities and their responses.

- Adversary \mathcal{B} randomly picks $(x, y) \in \mathbb{Z}_p^{*2}$ and generates the proposed CIBPRE scheme's master public key $\mathbf{PK}_{PRE} = (\mathbf{PK}_{IBBE}, u, u^r, \dots, u^{r^N}, t, t^r, \dots, t^{r^N}, H, H')$, where $u^{r^i} = h^{r^i \cdot x}$ and $t^{r^i} = h^{r^i \cdot y}$ for $i \in [0, N]$. Since the hash functions H and H' also are modeled as two random oracles in this proof, adversary \mathcal{B} sends the reduced master public key $\mathbf{PK}_{PRE} = (\mathbf{PK}_{IBBE}, u, u^r, \dots, u^{r^N}, t, t^r, \dots, t^{r^N})$ to adversary \mathcal{A} . And adversary \mathcal{B} provides the hash query services $Q_{PRE}^H(ID)$ and $Q_{PRE}^{H'}(Y)$ (where $Y \in \mathbb{G}_T$) for adversary \mathcal{A} and models the hash functions H and H' as two random oracles. Indeed, $Q_{PRE}^H(ID)$ is a copy of $Q_{IBBE}^H(ID)$. Any query of adversary \mathcal{A} to $Q_{PRE}^H(ID)$ will be forwarded by adversary \mathcal{B} to challenger \mathcal{C} , and any response from challenger \mathcal{C} also will be forwarded by adversary \mathcal{B} to adversary \mathcal{A} . In addition, adversary \mathcal{B} prepares a table $\mathcal{L}_{H'}$ for $Q_{PRE}^{H'}$ with columns (*GroupElement*, *HashValue*) to store the queried group elements and their responses.
- **Query Phase 1:** Adversary \mathcal{B} can issue hash query $Q_{IBBE}^H(ID)$ and private-key query $Q_{IBBE}^{SK}(ID)$ to challenger \mathcal{C} . Adversary \mathcal{A} can issue hash queries $Q_{PRE}^H(ID)$ and $Q_{PRE}^{H'}(Y)$, private-key query $Q_{PRE}^{SK}(ID)$, and re-encryption key query $Q_{PRE}^{RK}(ID, S', a)$ to adversary \mathcal{B} . These queries work as follows:
 - Hash query $Q_{IBBE}^H(ID)$: To answer the query “ ID ,” if there is a tuple $(ID, X \in \mathbb{Z}_p^*) \in \mathcal{L}_H$, challenger \mathcal{C} responds X ; otherwise challenger \mathcal{C} randomly chooses a value $X \in \mathbb{Z}_p^*$, adds tuple (ID, X) into table \mathcal{L}_H , and responds X .
 - Private-key query $Q_{IBBE}^{SK}(ID)$: To answer the query “ ID ,” if $ID \in S^*$, challenger \mathcal{C} responds \perp ; otherwise, challenger \mathcal{C} responds $SK_{IBBE}^{ID} = g^{\frac{1}{r + Q_{IBBE}^H(ID)}}$.
 - Hash query $Q_{PRE}^H(ID)$: To answer the query “ ID ,” adversary \mathcal{B} forwards the query to $Q_{IBBE}^H(ID)$ and returns the response of $Q_{IBBE}^H(ID)$.
 - Hash query $Q_{PRE}^{H'}(Y \in \mathbb{G}_T)$: To answer the query “ Y ,” if there is a tuple $(Y, Z \in \mathbb{G}) \in \mathcal{L}_{H'}$, adversary \mathcal{B} responds Z ; otherwise adversary \mathcal{B} randomly chooses a value $Z \in \mathbb{G}$, adds tuple (Y, Z) into table $\mathcal{L}_{H'}$, and responds Z .
 - Private-key query $Q_{PRE}^{SK}(ID)$: To answer the query “ ID ,” if $ID \in S^*$, adversary \mathcal{B} responds \perp ; if there is a tuple $(ID', S', *, \alpha^*)$ in table \mathcal{L}_{RK}

and it has $ID \in S^{*\leftarrow}$ and $ID \in S'$, adversary \mathcal{B} responds \perp ; otherwise, adversary \mathcal{B} forwards the query to $Q_{IBBE}^{SK}(ID)$ and obtains SK_{IBBE}^{ID} , returns $SK_{PRE}^{ID} = SK_{IBBE}^{ID}$, and adds ID into table \mathcal{L}_{SK} .

- Re-encryption key query $Q_{PRE}^{RK}(ID, S', \alpha)$: Adversary \mathcal{A} queries the re-encryption key from the identity ID to the identity set S' with the condition α . To answer this query, if there is a tuple $(ID, S', d_{ID \rightarrow S'|\alpha}, \alpha)$ in table \mathcal{L}_{RK} , adversary \mathcal{B} responds $d_{ID \rightarrow S'|\alpha}$ to adversary \mathcal{A} ; otherwise, we have the following cases:

- $ID \notin S^*$: Adversary \mathcal{B} queries $Q_{IBBE}^{SK}(ID)$ to get the private key SK_{IBBE}^{ID} , randomly chooses $(k, s) \in \mathbb{Z}_p^{*2}$, computes the re-encryption key

$$d_{ID \rightarrow S'|\alpha} = (w^{-k}, h^{k \prod_{ID_i \in S'} (\gamma + Q_{PRE}^H(ID_i))}, Q_{PRE}^H(v^{k'}) \cdot h^s, SK_{IBBE}^{ID} \cdot \left(u \cdot t^\alpha \right)^{\frac{s}{Q_{PRE}^H(ID)}}$$

responds $d_{ID \rightarrow S'|\alpha}$ to adversary \mathcal{A} , and adds tuple $(ID, S', d_{ID \rightarrow S'|\alpha}, \alpha)$ into table \mathcal{L}_{RK} .

- $ID \in S^{*\leftarrow}, \alpha = \alpha^{*\leftarrow}$ and $S' \cap \mathcal{L}_{SK} \neq \emptyset$: Adversary \mathcal{B} responds \perp .

- In the following three cases—(a) $ID \in S^{*\leftarrow}, \alpha \neq \alpha^*$, and $S' \cap \mathcal{L}_{SK} \neq \emptyset$; (b) $ID \in S^{*\leftarrow}, \alpha \neq \alpha^*$, and $S' \cap \mathcal{L}_{SK} = \emptyset$; and (c) $ID \in S^{*\leftarrow}, \alpha = \alpha^*$, and $S' \cap \mathcal{L}_{SK} = \emptyset$ —adversary \mathcal{B} randomly chooses $d_4 \in \mathbb{G}$ and $(k, s) \in \mathbb{Z}_p^{*2}$, responds the re-encryption key $d_{ID \rightarrow S'|\alpha} = (w^{k'}, h^{k' \prod_{ID_i \in S'} (\gamma + Q_{PRE}^H(ID_i))}, Q_{PRE}^H(v^{k'}) \cdot h^s, d_4)$ to adversary \mathcal{A} , and adds tuple $(ID, S', d_{ID \rightarrow S'|\alpha}, \alpha)$ into table \mathcal{L}_{RK} .

- **Challenge phase:** When adversary \mathcal{A} decides that query Phase 1 is over, he sends two challenge messages (m_0, m_1) to adversary \mathcal{B} . Adversary \mathcal{B} forwards (m_0, m_1) to challenger \mathcal{C} . Challenger \mathcal{C} generates the challenge ciphertext $C_{IBBE}^{* \leftarrow} = \text{Enc}_{IBBE}(\text{PK}_{IBBE}, S^*, m_b)$ (where b is randomly chosen in $\{0, 1\}$) and sends $C_{IBBE}^{* \leftarrow}$ to adversary \mathcal{B} . According to the structure of the ciphertext $C_{IBBE}^{* \leftarrow}$, let $C_{IBBE}^{* \leftarrow} = (c_1, c_2, c_3)$. And without loss of generality, let $c_1 = w^{-k}, c_2 = h^{k \prod_{ID_i \in S^*} (\gamma + Q_{IBBE}^H(ID_i))}$ and $c_3 = v^k \cdot m_b$. Recall that in setup phase, adversary \mathcal{B} randomly chooses $(x, y) \in \mathbb{Z}_p^{*2}$ and sets $u = h^x$ and $t = h^y$. Adversary \mathcal{B} can extend the ciphertext $C_{IBBE}^{* \leftarrow}$ to a valid initial CIBPRE ciphertext by computing $c_4 = (c_2^x \cdot c_2^{y-\alpha})^{\frac{1}{\prod_{ID_i \in S^*} Q_{PRE}^H(ID_i)}}$. And let $C_{PRE}^{* \leftarrow} = (C_{IBBE}^{* \leftarrow}, c_4)$. The ciphertext $C_{PRE}^{* \leftarrow}$ is a valid challenge ciphertext which was defined in the IND-sID-CPA security of CIBPRE, since we have $Q_{IBBE}^H = Q_{PRE}^H$ and $c_4 = (c_2^x \cdot c_2^{y-\alpha})^{\frac{1}{\prod_{ID_i \in S^*} Q_{PRE}^H(ID_i)}}$ = $(u \cdot t^\alpha)^{\frac{k \prod_{ID_i \in S^*} (\gamma + Q_{IBBE}^H(ID_i))}{Q_{PRE}^H(ID_i)}}$. Finally adversary \mathcal{B} sends the challenge ciphertext $C_{PRE}^{* \leftarrow}$ to adversary \mathcal{A} .

- **Query Phase 2:** This phase is the same with query phase 1.

- **Guess phase:** When adversary \mathcal{A} outputs a guess $b' \in \{0, 1\}$, adversary \mathcal{B} forwards b' to challenger \mathcal{C} .

We can find that adversary \mathcal{B} successfully and efficiently simulates \mathcal{A} 's view in attacking the IND-sID-CPA security of the proposed CIBPRE scheme, except the re-encryption key query $Q_{PRE}^{RK}(ID, S', \alpha)$ in the following three cases:

1. Case 1 : $ID \in S^*$, $\alpha \neq \alpha^*$ and $S' \cap \mathcal{L}_{SK} \neq \emptyset$;
2. Case 2 : $ID \in S^*$, $\alpha \neq \alpha^*$ and $S' \cap \mathcal{L}_{SK} = \emptyset$;
3. Case 3 : $ID \in S^*$, $\alpha = \alpha^*$ and $S' \cap \mathcal{L}_{SK} = \emptyset$.

In other words, if we suppose the exception in above three cases is not found by adversary \mathcal{A} . Then adversary \mathcal{B} successfully breaks the IND-sID-CPA security of D07 scheme, if adversary \mathcal{A} successfully breaks the IND-sID-CPA security of the proposed CIBPRE scheme. Next, we compute the probability of adversary \mathcal{A} to find the exception.

In the exception, adversary \mathcal{B} forges a re-encryption key:

$$d_{ID \rightarrow S' | \alpha} = (w^{??-k'}, h^{k'} \cdot \prod_{ID_i \in S'} (\gamma + Q_{PRE}^H(ID_i)), Q_{PRE}^{H'}(v^{k'}) \cdot h^s, d_4),$$

by randomly choosing $d_4 \in \mathbb{G}$ and $(k', s) \in \mathbb{Z}_p^{*2}$. Suppose adversary \mathcal{A} totally queries $Q_{PRE}^{RK} T$ times. We next prove that the probability of adversary \mathcal{A} to find the exception is no more than $T \cdot (Adv_{DBDH, \mathcal{A}}^{IND} + Adv_{D07, \mathcal{A}}^{IND-CPA})$ by following the following two lemmas. Let $\mathcal{A}_{Dis}(A, B)$ denote the event that adversary \mathcal{A} can distinguish event A and event B .

Lemma 7.2. *In Case 1, the probability of adversary \mathcal{A} to find the exception of $d_{ID \in S' | \alpha}$ is no more than $Adv_{DBDH, \mathcal{A}}^{IND}$.*

Proof 7.7. Suppose adversary \mathcal{A} can find the exception of $d_{ID \in S' | \alpha}$ in this case. In Case 1, since $S' \cap \mathcal{L}_{SK} \neq \emptyset$, adversary \mathcal{A} can decrypt out h^s ; we have that $\mathcal{A}_{Dis}(d_4, SK_{IBBE}^{ID} \cdot (u \cdot t^\alpha)^{\frac{s}{Q_{PRE}^H(ID)}})$ holds. Since adversary \mathcal{A} can compute $h^{(\gamma + Q_{PRE}^H(ID))}$, we have $\mathcal{A}_{Dis}(d_4, SK_{IBBE}^{ID} \cdot (u \cdot t^\alpha)^{\frac{s}{Q_{PRE}^H(ID)}})$ holds and $\Rightarrow \mathcal{A}_{Dis}(e(d_4, h^{(\gamma + Q_{PRE}^H(ID))}), e(SK_{IBBE}^{ID} \cdot (u \cdot t^\alpha)^{\frac{1}{Q_{PRE}^H(ID)}}), h^{(\gamma + Q_{PRE}^H(ID))}))$ holds. Since $e(g, h)$ is a public value, we have $\mathcal{A}_{Dis}(e(d_4, h^{(\gamma + Q_{PRE}^H(ID))}), e(SK_{IBBE}^{ID} \cdot (u \cdot t^\alpha)^{\frac{1}{Q_{PRE}^H(ID)}}), h^{(\gamma + Q_{PRE}^H(ID))}))$ holds and $\Rightarrow \mathcal{A}_{Dis}(e(d_4, h^{(\gamma + Q_{PRE}^H(ID))}) \cdot e(g, h)^{-1}, e((u \cdot t^\alpha)^{\frac{s}{Q_{PRE}^H(ID)}}), h^{(\gamma + Q_{PRE}^H(ID))}))$ holds. Since d_4 was randomly chosen in \mathbb{G} , there is a value $r \in \mathbb{Z}_p^*$ having $e(d_4, h^{(\gamma + Q_{PRE}^H(ID))}) \cdot e(g, h)^{-1} = e(h, h)^r$. And since $u = h^x$ and $t = h^y$, we have $\mathcal{A}_{Dis}(e(d_4, h^{(\gamma + Q_{PRE}^H(ID))}) \cdot e(g, h)^{-1}, e((u \cdot t^\alpha)^{\frac{s}{Q_{PRE}^H(ID)}}), h^{(\gamma + Q_{PRE}^H(ID))})) \Rightarrow \mathcal{A}_{Dis}(e(h, h)^r, e(h, h)^{(x+y\alpha) \cdot S \cdot (\gamma + Q_{PRE}^H(ID)) \cdot \frac{1}{Q_{PRE}^H(ID)}})$ holds. In addition, it is clear that adversary \mathcal{A} knows $h^{x+y\alpha}$, h^s and $h^{(\gamma + Q_{PRE}^H(ID)) \cdot \frac{1}{Q_{PRE}^H(ID)}}$. Now we compute the probability of the event that $\mathcal{A}_{Dis}(e(h, h)^r, e(h, h)^{(x+y\alpha) \cdot S \cdot (\gamma + Q_{PRE}^H(ID)) \cdot \frac{1}{Q_{PRE}^H(ID)}})$ holds. When $\mathcal{A}_{Dis}(e(h, h)^r, e(h, h)^{(x+y\alpha) \cdot S \cdot (\gamma + Q_{PRE}^H(ID)) \cdot \frac{1}{Q_{PRE}^H(ID)}})$ holds, it means that taking $h^{x+y\alpha}$, h^s and $h^{(\gamma + Q_{PRE}^H(ID)) \cdot \frac{1}{Q_{PRE}^H(ID)}}$ as input, adversary \mathcal{A} can solve an instance of the DBDH problem by distinguishing $e(h, h)^r$ and $e(h, h)^{(x+y\alpha) \cdot S \cdot (\gamma + Q_{PRE}^H(ID)) \cdot \frac{1}{Q_{PRE}^H(ID)}}$. According to the definition of the DBDH problem, we have that $\mathcal{A}_{Dis}(e(h, h)^r,$

$e(h, h)^{(x+\gamma\alpha) \cdot S \cdot (\gamma + Q_{PRE}^H(ID)) \cdot \frac{1}{Q_{PRE}^H(ID)}}$ holds with the probability no more than $Adv_{DBDH, \mathcal{A}}^{IND}$. Therefore, adversary \mathcal{A} has the probability no more than $Adv_{DBDH, \mathcal{A}}^{IND}$ to find the exception of $d_{ID \rightarrow S'}|_{\alpha}$ in Case 1.

Lemma 7.3. *In Cases 2 and 3, the probability of adversary \mathcal{A} to find the exception of $d_{ID \rightarrow S'}|_{\alpha}$ is no more than $Adv_{D07, \mathcal{A}}^{IND-sID-CPA}$.*

Proof 7.8. Since d_4 is randomly chosen in \mathbb{G} , it is clear that there is a value $s' \in \mathbb{Z}_p^*$ having $d_4 = SK_{IBBE}^{ID} \cdot (u \cdot t^\alpha)^{\frac{1}{Q_{PRE}^H(ID)}}$. If adversary \mathcal{A} can find the exception in Cases 2 and 3, it means that adversary \mathcal{A} can find that $s' = s$. Recall that in $d_{ID \rightarrow S'}|_{\alpha}$ the part $(w^{??-k'}, h^{k'} \cdot \prod_{ID_i \in S'} (\gamma + Q_{PRE}^H(ID_i)), Q_{PRE}^H(v^{k'}) \cdot h^s)$ indeed is an encryption of h^s . If adversary \mathcal{A} can find that $s' = s$, we have that $A_{Dis}(A, B)$ holds, where $A = (w^{??-k'}, h^{k'} \cdot \prod_{ID_i \in S'} (\gamma + Q_{PRE}^H(ID_i)), Q_{PRE}^H(v^{k'}) \cdot h^s)$, $B = (w^{??-k''}, h^{k''} \cdot \prod_{ID_i \in S'} (\gamma + Q_{PRE}^H(ID_i)), Q_{PRE}^H(v^{k''}) \cdot h^s)$ and k'' is randomly chosen in \mathbb{Z}_p^* . Next, we prove that $A_{Dis}(A, B)$ holds only when adversary \mathcal{A} can break the IND-sID-CPA security of the D07 scheme.

Since Q_{PRE}^H is a random oracle, there is a value $R \in \mathbb{G}_T$ having $Q_{PRE}^H(v^{k''}) \cdot h^s = Q_{PRE}^H(v^{k''} \cdot R) \cdot h^s$. So when $A_{Dis}(A, B)$ holds, we have that $A_{Dis}(A', B')$ holds, where $A' = (w^{??-k'}, h^{k'} \cdot \prod_{ID_i \in S'} (\gamma + Q_{PRE}^H(ID_i)), v^{k'})$ and $B' = (w^{??-k''}, h^{k''} \cdot \prod_{ID_i \in S'} (\gamma + Q_{PRE}^H(ID_i)), v^{k''} \cdot R)$. It is clear that A' and B' are two ciphertext generated by the D07 scheme. Furthermore, since we have $S' \cap \mathcal{L}_{SK} = \emptyset$ both in Cases 2 and 3, it means that when adversary \mathcal{A} takes S' as the set of challenge identities and takes $1 \in \mathbb{G}_T$ and R as two challenge messages, he can break the IND-sID-CPA security of the D07 scheme without knowing the private key SK_{IBBE}^{ID} of any $ID \in S'$. According to the definition of D07's IND-sID-CPA security, the probability of adversary \mathcal{A} to find the exception is no more than $Adv_{D07, \mathcal{A}}^{IND-sID-CPA}$ in Cases 2 and 3.

According to above two lemmas, we clearly have the following result.

Lemma 7.4. *When adversary \mathcal{A} totally queries $Q_{PRE}^{RK} T$ times, adversary \mathcal{B} successfully simulates adversary \mathcal{A} 's view in attacking the IND-sID-CPA security of our proposed CIBPRE scheme, except the probability no more than $T \cdot (Adv_{DBDH, \mathcal{A}}^{IND} + Adv_{D07, \mathcal{A}}^{IND-sID-CPA})$.*

Since we suppose that adversary \mathcal{A} has the advantage $Adv_{CIBPRE, \mathcal{A}}^{IND-sID-CPA}$ to break our scheme, we have that adversary \mathcal{B} has the advantage $Adv_{D07, \mathcal{B}}^{IND-sID-CPA} = Adv_{CIBPRE, \mathcal{A}}^{IND-sID-CPA} (1 - ??T \cdot Adv_{DBDH, \mathcal{A}}^{IND} - ??T \cdot Adv_{D07, \mathcal{A}}^{IND-sID-CPA})$ to break the IND-sID-CPA security of the D07 scheme.

Since the D07 scheme is IND-sID-CPA secure in the RO model, both $Adv_{D07, \mathcal{B}}^{IND-sID-CPA}$ and $Adv_{D07, \mathcal{A}}^{IND-sID-CPA}$ are negligible. In addition, since the DBDH assumption holds, $Adv_{DBDH, \mathcal{A}}^{IND}$ also is negligible. So it demonstrates that $Adv_{CIBPRE, \mathcal{A}}^{IND-sID-CPA}$ must be negligible. In other words, the proposed CIBPRE scheme is IND-sID-CPA secure in the RO model.

The intuition of security proof is described in **Figure 7.11**.

7.4. Comparison and observations

Blaze et al. [36] proposed the notion of “atomic proxy cryptography,” in which a user could flexibly authorize a set of users to access files stored in

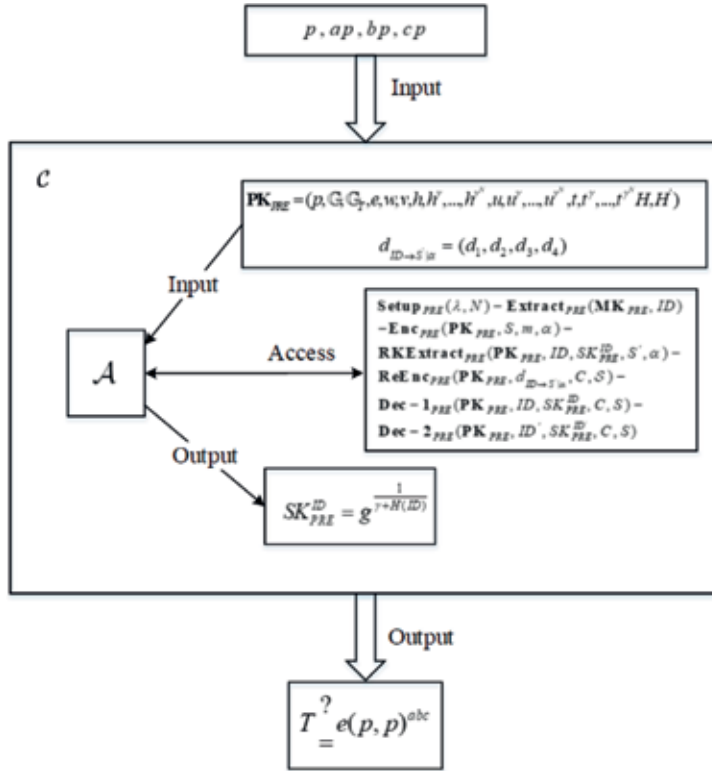


Figure 7.11. Intuition of security proof in CIBPRE scheme.

remote server and the original authorized users change nothing. However, the scheme involves a non-ignorable limitation noted by the authors. The scheme is bidirectional so that the apply scenarios will be limited to the situation that the message sender and receiver should trust each other. This problem could be resolved by generating a pair of delegation key, but that would increase some computation overload of the proxy. Moreover, the scheme remains several sensitive problems. There exists a problem that a medium proxy could create a non-existent delegation between two users without knowing each other. Another drawback of this scheme is not resistant to the collusion attack between the proxy and the sender.

Ateniese et al. [37] propose a notion of improved proxy re-encryption from both theoretical and practical perspectives. Theoretically, the authors introduce the construction of the scheme and give the security proof to describe the CPA security and demonstrate the usefulness of proxy re-encryption as a method of adding access control to a secure file system. Practically, they also implement a concrete cryptographic file system and measure the performance measurements of the experimental file system guarantee the efficiency of the scheme.

The first unidirectional PRE realizations with chosen-ciphertext security in the standard model is explored by [39]. By permitting attacker's query to any delegates' public key, they can provide the security guarantee. To the best of our knowledge, these are the first security results in the so-called chosen key model for the proxy re-encryption primitive. While the scheme still remains a mount of open questions. For example, the system cannot guarantee the security if some users and proxy collude with each other. That would be an interesting remaining problem to resolve by proposing a new scheme resisting the collusion.

Green and Ateniese [45] introduce the notion of identity-based encryption (IBE) to proxy re-encryption, in which senders use the an identity information as the public key to compute the encrypted data. For instance, Charles could encrypt a message for Alice by just using her email address. And the constructions are compatible with existing IBE schemes and can be implemented by running on the existing cryptographic system. And the authors give a concrete security proof shown that the identity-based proxy re-encryption scheme is IND-PrID-CCA-secure. One challenging open problem is to explore an efficient constructions for multiuse CCA-secure IBE-PRE schemes.

Zhou et al. [204] creatively explore a promising proxy re-encryption mechanism, identity-based proxy re-encryption version 2 (IBPRE2), which allows the proxy to convert from a ciphertext in one cryptosystem (IBBE) into another cryptosystem (IBE). With IBPRE2, a sender can utilize the IBBE to securely broadcast a message to a group of receivers. Furthermore, if another user requests to access the encrypted data, it is simple to authorize the privilege to those users by delegating a re-encryption key to proxy to convert the ciphertext, without decrypting the IBBE ciphertext nor leaking any sensitive information. Based on formalizing the security requirements in IBPRE2, the introduced mechanism is probably chosen-ciphertext attack secure.

By taking the advantage of conditional PRE (CPRE), identity-based PRE (IPRE), and broadcast PRE (BPRE), Xu et al. [205] describe a versatile primitive called conditional identity-based broadcast PRE (CIBPRE) and formalize its semantic

Schemes	Unidirectional	CPA security	CCA2 security	Collusion-safe
BBS-PRE	NO	YES	NO	NO
ASDS-IPRE	YES	YES	NO	YES
IUCCS-PRE	YES	YES	YES	YES
IBPRE	YES	YES	YES	YES
IBPRE2	YES	YES	YES	NO
CIBB-PRE	YES	YES	NO	NO

Table 7.1.
Property comparison.

Steps	BBS-PRE	ASDS-IPRE	IUCCS-PRE	IBPRE	IBPRE2	CIBB-PRE
Enc1	2Exp1	3Exp1	5Exp1+1P+1Exp2	4Exp1+1P	2Exp1+1P+Exp2	5Exp1
Enc2	X	2Exp1	3Exp1+1P+1Exp2	X	X	X
Re-Encryption	1Exp1	1P	4Exp1	2Exp1+2P	5Exp1+1P	1Exp1+2P+1Exp2
Dec1	1Exp1	2Exp1+1P+1Exp2	1Exp1+1P+1Exp2	2Exp1+2P	1Exp1+9P+2Exp2	1Exp1+2P+1Exp2
Dec2	X	2Exp1+1P+1Exp2	1Exp1+1P+1Exp2	X	X	1Exp1+2P+1Exp2

^aExp1 denotes a modular exponentiation on Group G_1 ; Exp2 denotes a modular exponentiation on Group G_2 .
^bP denotes a paring computation.

Table 7.2.
Efficiency comparison.

security. It allows a sender to broadcast his outsourced ciphertext to some other receivers in a fine-grained manner. Compared with the previous techniques such as PGP and IBE, the CIBPRE-based system is much more efficient in the aspect of communication and more practical in user experience.

In **Table 7.1**, we make a comparison of properties of the schemes. And we give the efficiency analysis in **Table 7.2**.

Symbol description


PKI	Public Key Infrastructure
CA	Certificate Authority
ID-PKC	Identity-based Public Key Cryptography
CL-PKC	Certificateless Public Key Cryptography
PKG	Private Key Generator
ICA	Identity Certifying Authority
KGC	Key Generation Center
CLE	Certificateless Encryption
CLS	Certificateless Signature
CL-AKA	Certificateless Authenticated Key Agreement
CB-PKC	Certificate-based Public Key Cryptography
SGC-PKC	Self-generated-certificate Public Key Cryptography

Author details

Zhen Qin, Erqiang Zhou, Yi Ding, Yang Zhao, Fuhu Deng and Hu Xiong*
University of Electronic Science and Technology of China, Chengdu, China

*Address all correspondence to: xionghu.uestc@gmail.com

IntechOpen

© 2018 The Author(s). Licensee IntechOpen. Distributed under the terms of the Creative Commons Attribution - NonCommercial 4.0 License (<https://creativecommons.org/licenses/by-nc/4.0/>), which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited. 

References

- [1] Liang X, Lu R, Lin X, Shen X. Security and Privacy in Mobile Social Networks. Berlin: Springer; 2013. p. 1
- [2] Ne S, Verine. Social media in rural China/social media in industrial China. China Perspectives. 2017
- [3] Toh C-K. Wireless ATM and Ad-Hoc Networks. US: Springer; 1997
- [4] Toh CK. Ad Hoc Mobile Wireless Networks: Protocols and Systems. Prentice Hall PTR; 2002
- [5] Zanjireh MM, Larijani H. A survey on centralised and distributed clustering routing algorithms for WSNS. In: Vehicular Technology Conference; 2015. pp. 1-6
- [6] Yan H, Huo H, Xu Y, et al. Wireless sensor network based E health system - implementation and experimental results. IEEE Transactions on Consumer Electronics. 2013;56(4):2288-2295
- [7] Ehsan S, Bradford K, Brugger M, Hamdaoui B, Kovchegov Y, Johnson D, Louhaichi M. Design and analysis of delay-tolerant sensor networks for monitoring and tracking free-roaming animals. IEEE Transactions on Wireless Communications. 2012;11(3):1220-1227
- [8] White BA, Tsourdos A, Ashokaraj I, Subchan S, Zbikowski R. Contaminant cloud boundary monitoring using network of UAV sensors. IEEE Sensors Journal. 2008;8(10):1681-1692
- [9] Hollow M. Pre-1900 utopian visions of the 'cashless society'. MPRA Paper; 2012
- [10] Wang Y, Chen I, Wang D, et al. A survey of mobile cloud computing applications: Perspectives and challenges. Wireless Personal Communications. 2015;80(4):1607-1623
- [11] Mun M, Reddy S, Shilton K, Yau N, Burke J, Estrin D, Hansen M, Howard E, West R, Boda P. Peir, the personal environmental impact report, as a platform for participatory sensing systems research. In: Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services; 2009. pp. 55-68
- [12] Vallet D, Castells P, Motta E. Semantically enhanced information retrieval: An ontology-based approach. Web Semantics Science Services & Agents on the World Wide Web. 2011; 9(4):434-452
- [13] Deng N, Tian Y, Zhang C. Support Vector Machines: Optimization based theory, algorithms, and extensions. Chapman & Hall, London, England, UK: CRC Press; 2012
- [14] Bengio Y et al. Learning deep architectures for AI. Foundations and Trends in Machine Learning. 2009;2(1): 1-127
- [15] Geoffrey E. Hinton. Deep belief networks. Scholarpedia. 2009;4(6):5947
- [16] Lecun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015;521(7553): 436-444
- [17] Dahl GE, Yu D, Deng L, Acero A. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. IEEE Transactions on Audio, Speech & Language Processing. 2012;20(1):30-42
- [18] Ji S, Xu W, Yang M, Yu K. 3D convolutional neural networks for human action recognition. IEEE Transactions on Pattern Analysis & Machine Intelligence. 2012;35(1): 221-231
- [19] Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F,

- Schwenk H, Bengio Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *Computer Science*. 2014; **arXiv:1406.1078**
- [20] Zhao R, Ouyang W, Li H, Wang X. Saliency detection by multi-context deep learning. In: *IEEE Conference on Computer Vision and Pattern Recognition*; 2015. pp. 1265-1274
- [21] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. In: *International Conference on Neural Information Processing Systems*; 2012. pp. 1097-1105
- [22] Bengio Y, Vincent P, Delalleau O, Le Roux N, Ouimet M. Out-of-sample extensions for LLE, isomap, MDS, eigenmaps, and spectral clustering. In: *International Conference on Neural Information Processing Systems*; 2003. pp. 177-184
- [23] Socher R, Pennington J, Huang EH, Ng AY, Manning CD. Semi-supervised recursive autoencoders for predicting sentiment distributions. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*; 27–31 July 2011. Edinburgh, UK: John McIntyre Conference Centre, A Meeting of Sigdat, A Special Interest Group of the ACL; 2011. pp. 151-161
- [24] Mikolov T, Sutskever I, Chen K, Corrado G, Dean J. Distributed representations of words and phrases and their compositionality. In: *International Conference on Neural Information Processing Systems*; 2013. pp. 3111-3119
- [25] Prakash A, McDaniel P. Methods and limitations of security policy reconciliation. *ACM Transaction on Information and System Security*. 2006; **9(3):259-291**
- [26] Waters B, Sahai A. Fuzzy identity-based encryption. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*; 2005. pp. 457-473
- [27] Hohenberger S, Green M, Waters B. Outsourcing the decryption of aBE ciphertexts. In: *Proceedings of the 20th USENIX conference on Security*; 2011
- [28] Li J, Li J, Jia C, Chen X. Outsourcing encryption of attribute-based encryption with mapreduce. In: *Proceedings of the 14th International Conference on Information and Communications Security*; 2012. pp. 191-201
- [29] Li J, Jia C, Ma J, Li J, Chen X, Lou W. Fine-grained access control system based on outsourced attribute-based encryption. In: *Proceedings of the 18th European Symposium on Research in Computer Security*; 2013. pp. 592-609
- [30] Guan C, Lai J, Deng RH, Weng J. Attribute-based encryption with verifiable outsourced decryption. *IEEE Transactions on Information Forensics & Security*. 2013;**8**: 1343-1354
- [31] Li J, Chen X, Li J, Huang X, Xiang Y. Securely outsourcing attribute-based encryption with checkability. *IEEE Transactions on Parallel & Distributed Systems*. 2014;**25(8):2201-2210**
- [32] Ma H, Lin S, Zhang R, Wang M. Revisiting attribute-based encryption with verifiable outsourced decryption. *IEEE Transactions on Information Forensics & Security*. 2015;**10(10): 2119-2130**
- [33] Ma H, Zhang R, Wan Z, Lu Y, Lin S. Verifiable and exculpable outsourced attribute-based encryption for access control in cloud computing. *IEEE Transactions on Dependable & Secure Computing*. 2017;**PP(99):1**

- [34] Mao X, Lai J, Mei Q, Chen K, Weng J. Generic and efficient constructions of attribute-based encryption with verifiable outsourced decryption. *IEEE Transactions on Dependable & Secure Computing*. 2016;**13**(5):533-546
- [35] Xu J, Wen Q, Li W, Jin Z. Circuit ciphertext-policy attribute-based hybrid encryption with verifiable delegation in cloud computing. *IEEE Transactions on Parallel & Distributed Systems*. 2016; **27**(1):119-129
- [36] Blaze M, Bleumer G, Strauss M. Divertible protocols and atomic proxy cryptography. In: *Proceedings of International Conference on the Theory and Application of Cryptographic Techniques*; 1998. pp. 127-144
- [37] Ateniese G, Fu K, Green M, Hohenberger S. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security*. 2006; **9**(1):1s-30s
- [38] Canetti R, Hohenberger S. Chosen-ciphertext secure proxy re-encryption. *IACR Cryptology ePrint Archive*. 2007; **2007**: 171
- [39] Libert B, Vergnaud D. Unidirectional chosen-ciphertext secure proxy re-encryption. In: *Proceedings of the 11th International Workshop on Practice and Theory in Public-Key Cryptography*; 2008. pp. 360-379
- [40] Hanaoka G, Kawai Y, Kunihiro N, Matsuda T, Weng J, Zhang R, Zhao Y. Generic construction of chosen ciphertext secure proxy re-encryption. In: *Conference on Topics in Cryptology*; 2012. pp. 349-364
- [41] Matsuda T, Nishimaki R, Tanaka K. CCA proxy re-encryption without bilinear maps in the standard model. In: *Proceedings of the 13th International Conference on Practice and Theory in Public Key Cryptography*; 2010. pp. 261-278
- [42] Shao J, Cao Z. CCA-secure proxy re-encryption without pairings. *IACR Cryptology ePrint Archive*. 2009;**2009**: 164
- [43] Weng J, Deng RH, Liu S, Chen K, Lai J, Wang XA. Chosen-ciphertext secure proxy re-encryption without pairings. *IACR Cryptology ePrint Archive*. 2008;**2008**:509
- [44] Weng J, Zhao Y, Hanaoka G. On the security of a bidirectional proxy re-encryption scheme from PKC 2010. *Proceedings of the 14th International Conference on Practice and Theory in Public Key Cryptography*. 2011. pp. 284-295
- [45] Green M, Ateniese G. Identity-based proxy re-encryption. In: *Proceedings of the 5th International Conference on the Applied Cryptography and Network Security*; 2007. pp. 288-306
- [46] Chu C-K, Tzeng W-G. Identity-based proxy re-encryption without random oracles. In: *Proceedings of 10th International Conference on the Information Security*; 2007. pp. 189-202
- [47] Green M, Ateniese G. Identity-based proxy re-encryption. In: *International Conference on Applied Cryptography and Network Security*; 2007. pp. 288-306
- [48] Han J, Susilo W, Mu Y. Identity-based data storage in cloud computing. *Future Generation Computer Systems*. 2013;**29**(3):673-681
- [49] Liang K, Liu JK, Wong DS, Susilo W. An Efficient Cloud-Based Revocable Identity-Based Proxy Re-encryption Scheme for Public Clouds Data Sharing. Berlin, Germany: Springer International Publishing; 2014

- [50] Luo S, Shen Q, Chen Z. Fully Secure Unidirectional Identity-Based Proxy Re-encryption. Berlin/Heidelberg: Springer; 2011
- [51] Matsuo T. Proxy re-encryption systems for identity-based encryption. In: International Conference on Pairing-Based Cryptography; 2007. pp. 247-267
- [52] Shao J, Cao Z. Multi-use unidirectional identity-based proxy re-encryption from hierarchical identity-based encryption. Information Sciences. 2012;206(16):83-95
- [53] Wang L, Wang L, Mambo M, Okamoto E. Identity-based proxy cryptosystems with revocability and hierarchical confidentialities. IEICE Trans Fundamentals. 2012;95(1): 383-400
- [54] Tang Q. Type-based proxy re-encryption and its construction. In: International Conference on Cryptology in India: Progress in Cryptology; 2008. pp. 130-144
- [55] Qiu J, Hwang GH, Lee H. Efficient conditional proxy re-encryption with chosen-ciphertext security. In: International Conference on Information Security; 2009. pp. 151-166
- [56] Weng J, Deng RH, Ding X, Chu CK, Lai J. Conditional proxy re-encryption secure against chosen-ciphertext attack. In: ACM ASIACCS; 2009. pp. 322-332
- [57] Liang K, Chu CK, Tan X, Wong DS, Tang C, Zhou J. Chosen-ciphertext secure multi-hop identity-based conditional proxy re-encryption with constant-size ciphertexts. Theoretical Computer Science. 2014;539(9):87-105
- [58] Liang K, Liu Z, Tan X, Wong DS, Tang C. A CCA-secure identity-based conditional proxy re-encryption without random oracles. In: International Conference on Information Security and Cryptology; 2012. pp. 231-246
- [59] He K, Weng J, Deng RH, Liu JK. On the security of two identity-based conditional proxy re-encryption schemes. Theoretical Computer Science. 2016;652:18-27
- [60] He K, Weng J, Liu JK, Zhou W, Liu JN. Efficient Fine-Grained Access Control for Secure Personal Health Records in Cloud Computing. Berlin, Germany: Springer International Publishing; 2016
- [61] Liang X, Cao Z, Lin H, Shao J. Attribute based proxy re-encryption with delegating capabilities. In: International Symposium on Information, Computer, and Communications Security; 2009. pp. 276-286
- [62] Lin S, Zhang R, Wang M. Verifiable attribute-based proxy re-encryption for secure public cloud data sharing. Security & Communication Networks. 2016;9(12):1748-1758
- [63] Fang L, Susilo W, Ge C, Wang J. Interactive conditional proxy re-encryption with fine grain policy. Journal of Systems & Software. 2011; 84(12):2293-2302
- [64] Liang K, Man HA, Liu JK, Susilo W, Wong DS, Yang G, Yu Y, Yang A. A secure and efficient ciphertext-policy attribute-based proxy re-encryption for cloud data sharing. Future Generation Computer Systems. 2015;52(C):95-108
- [65] Liang K, Susilo W. Searchable attribute-based mechanism with efficient data sharing for secure cloud storage. IEEE Transactions on Information Forensics & Security. 2015; 10(9):1981-1992
- [66] Liang K, Susilo W, Liu JK. Privacy-preserving ciphertext multi-sharing control for big data storage. IEEE Transactions on Information Forensics & Security. 2015;10(8):1578-1589

- [67] Lee CC, Li CT, Chen CL, Chiu ST. A searchable hierarchical conditional proxy re-encryption scheme for cloud storage services. *Information Technology & Control*. 2016;**45**(3): 289-299
- [68] LeVeque L. *Fundamentals of Number Theory*. New ed. Dover Publications; 1996
- [69] Galbraith SD. *Mathematics of Public Key Cryptography*. Cambridge, England, UK: Cambridge University Press; 2012
- [70] Shoup V. *A Computational Introduction to Number Theory and Algebra*. Cambridge, England, UK: Cambridge University Press; 2008
- [71] Hankerson D, Menezes A, Vanstone S. *Guide to Elliptic Curve Cryptography*. Berlin, Germany: Springer; 2012
- [72] Barreto P, Kim H, Bynn B, Scott M. Efficient algorithms for pairing-based cryptosystems. In: *Advances in Cryptology-CRYPTO 2002*; Springer; 2002. pp. 354-369
- [73] Menezes AJ, Oorschot PC, Vanstone SA. *Handbook of Applied Cryptography*. Boca Raton, Florida, USA: CRC Press; 1996
- [74] Katz J. *Digital Signatures*. Berlin, Germany: Springer; 2010
- [75] Rivest RL, Shamir A, Adleman LM. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the Association for Computing Machinery*. 1978;**21**(2):120-126
- [76] El Gamal T. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*. 1985;**31**(4):469-472
- [77] Boneh D, Franklin M. Identity-based encryption from the weil pairing. In: *Advances in Cryptology-CRYPTO 2001*; Springer; 2001. pp. 213-229
- [78] Schnorr C-P. Efficient identification and signatures for smart cards. In: *Advances in Cryptology-Crypto '89*; Springer Press; 1984. pp. 239-252
- [79] NIST. *Digital Signature Standard (DSS)*. Federal Information Processing Standards (FIPS PUB 186-4); 1993
- [80] Cha JC, Cheon JH. An identity-based signature from gap diffie-hellman groups. In: *6th International Workshop on Theory and Practice in Public Key Cryptography (Public Key Cryptography-PKC 2003)*; Springer Press; 2003. pp. 18-30
- [81] Bellare M, Namprempre C, Neven G. Security proofs for identity-based identification and signature schemes. In: *Advances in Cryptology-EuroCrypt 2004*; Springer; 2004. pp. 268-286
- [82] Goldwasser S, Micali S. Probabilistic encryption. *Journal of Computer and System Sciences*. 1984; **28**(2):270-299
- [83] Katz J, Lindell Y. *Introduction to Modern Cryptography*. Boca Raton, Florida, USA: CRC Press; 2007
- [84] Stern J. Why provable security matters? In: *Advances in Cryptology-EUROCRYPT*. Berlin, Germany: Springer Press; 2003. pp. 449-461
- [85] Cramer R, Shoup V. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: *Advances in Cryptology-CRYPTO*; Springer; 1998. pp. 13-25
- [86] Naor M, Yung M. Public-key cryptosystems provably secure against chosen ciphertext attacks. In: *Proceedings of the 22nd Annual ACM*

Symposium on Theory of Computing (STOC 1990); 1990. pp. 427-437

[87] Goldwasser S, Micali S, Rivest RL. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*. 1988;17(2): 281-308

[88] Salton G, Lesk ME. Computer evaluation of indexing and text processing. *Journal of the ACM*. 1968;15(1):8-36

[89] Salton G. The SMART Retrieval System—Experiments in Automatic Document Processing. Englewood Cliffs, NJ: Prentice Hall Inc.; 1971

[90] Salton G, Buckley C. Term-weighting approaches in automatic retrieval. *Information Processing & Management*. 1988;24(5):513-523

[91] Robertson SE, Sparck Jones K. Relevance weighting of search terms. *Journal of the American Society for Information Sciences*. 1976;27(3): 129-146

[92] Fuhr N. Probabilistic models in information retrieval. *The Computer Journal*. 1992;35(3):243-255

[93] van Rijsbergen CJ. *Information Retrieval*. Butterworths; 1979

[94] Francis W, Kucera H. *Frequency Analysis of English Usage*. Houghton Mifflin Co.; 1982

[95] Tran TM, Le XMT, Vinh VT, Nguyen HT, Nguyen TM. A weighted local mean-based k-nearest neighbors classifier for time series. In: *International Conference on Machine Learning and Computing*; 2017. pp. 157-161

[96] Heaps J. *Information Retrieval—Computational and Theoretical Aspects*. Academic Press; 1978

[97] Nesbit J. The accuracy of approximate string matching algorithms. *Journal of Computer-Based Instruction*. 1986;13(3):80-83

[98] Fox C. Lexical analysis and stoplists. In: Frakes W, Baeza-Yates R, editors. *Information Retrieval: Data Structures and Algorithms*. Englewood Cliffs, NJ, USA: Prentice Hall; 1992. pp. 102-130

[99] Frakes WB, Baeza-Yates R. *Information Retrieval: Data Structures and Algorithms*. Englewood Cliffs, NJ, USA: Prentice Hall; 1992

[100] Townsend JT. Theoretical analysis of an alphabetic confusion matrix. *Attention, Perception, & Psychophysics*. 1971;9(1):40-50

[101] Sokolova M, Japkowicz N, Szpakowicz S. Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation. In: *Australian Conference on Artificial Intelligence*; 2006;4304:1015-1021

[102] Kohavi R. Scaling up the accuracy of Naive-Bayes classifiers: A decision-tree hybrid. *KDD*. 1996;96:202-207

[103] Schmid H. Probabilistic part-of-speech tagging using decision trees. *International Conference on New Methods in Language Processing*. 1994. p. 154

[104] Zhang Y, Wang S, Phillips P, Ji G. Binary PSO with mutation operator for feature selection using decision tree applied to spam detection. *Knowledge-Based Systems*. 2014;64(1):22-31

[105] Kent JT. Information gain and a general measure of correlation. *Biometrika*. 1983;70(1):163-173

[106] Liu, Yuxun, N. Xie. Improved ID3 algorithm. *IEEE International Conference on Computer Science and Information Technology*, 2010:465-468

- [107] Kaminski B, Jakubczyk M, Szufel P. A framework for sensitivity analysis of decision trees. *Central European Journal of Operations Research*. 2018;**26**(1):135-159
- [108] Surhone LM, Timpledon MT, Marseken SF. *Machine Learning, Decision Tree, Horizon Effect, Alpha Beta Pruning, and Artificial Neural Network. Pruning (Algorithm)*. Betascript Publishing; 2010
- [109] Santos I, Brezo F, Ugarte-Pedrero X, Bringas PG. Opcode sequences as representation of executables for data-mining-based unknown malware detection. *Information Sciences*. 2013; **231**(9):64-82
- [110] Wang S, Manning CD. Baselines and bigrams: Simple, good sentiment and topic classification. In: *Meeting of the Association for Computational Linguistics: Short Papers*; 2012. pp. 90-94
- [111] Rish I. An empirical study of the naive bayes classifier. *Journal of Universal Computer Science*. 2001;**1**(2):127
- [112] Geenen PL, Van Der Gaag LC, Loeffen WL, Elbers AR. Constructing naive bayesian classifiers for veterinary medicine: A case study in the clinical diagnosis of classical swine fever. *Research in Veterinary Science*. 2011; **91**(1):64-70
- [113] Adankon MM, Cheriet M. Support vector machine. In: *International Conference on Intelligent Networks and Intelligent Systems*; 2009. pp. 418-421
- [114] Utkin LV, Chekh AI, Zhuk YA. Binary classification SVM-based algorithms with interval-valued training data using triangular and epanechnikov kernels. *Neural Networks*. 2016;**80**: 53-66
- [115] Deylami HM, Singh YP. Adaboost and SVM based cybercrime detection and prevention model. *Artificial Intelligence Research*. 2012;**1**(2):117-130
- [116] Zhang J, Marszalek M, Lazebnik S, Schmid C, Zhang J, et al. Local features and kernels for classification of texture and object categories: A comprehensive study. *International Journal of Computer Vision (IJCV)*. 2007;**73**(2): 213-238
- [117] Choi YS, Xue F, Yeh SP, Talwar S. Transformed kernels for cancelling non-linear distortion. US patent 9698862. 2017
- [118] Lin HT, Lin CJ. A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods. *Neural Computation*. 2005;**27**(1):15-23
- [119] Paepke D. A slightly different view of heaven. *Esquire*. 2013;**160**(4):24
- [120] Liu X, Aggarwal C, Li YF, Kong X, Sun X, Sathe S. Kernelized matrix factorization for collaborative filtering. In: *SIAM International Conference on Data Mining*; 2016. pp. 378-386
- [121] Selesnick I, Farshchian M. Sparse Signal Approximation via Nonseparable Regularization. *IEEE Transactions on Signal Processing*. 2017;**65**(10):2561-2575
- [122] Surhone LM, Tennoe MT, Henssonow SF, Breiman L. Random forest. *Machine Learning*. 2010;**45**(1): 5-32
- [123] Pal M. Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*. 2005;**26**(1):217-222
- [124] Bosch A, Zisserman A, Munoz X. Image classification using random forests and ferns. In: *IEEE International Conference on Computer Vision*; 2007. pp. 1-8

- [125] Shen L, Liu C, Jiang L, Chunxue WU. An adaptive improved algorithm for k-nearest neighbors. *Electronic Science & Technology*. 2017;**30**(7): 29-32
- [126] Veerasha M, Sugumaran M. k-Nearest neighbor queries in wireless broadcast environments. In: *International Conference on Communication and Electronics Systems*; 2017. pp. 1-4
- [127] Li Y, Chen H, Lv M, Li Y. Event-based k-nearest neighbors query processing over distributed sensory data using fuzzy sets. *Soft Computing*. 2017; **3**:1-13
- [128] Chandel K, Kunwar V, Sabitha S, Choudhury T, Mukherjee S. A comparative study on thyroid disease detection using k-nearest neighbor and naive bayes classification techniques. *CSI Transactions on ICT*. 2016;**4**(2-4): 313-319
- [129] Keller JM, Gray MR, Givens JA. A fuzzy k-nearest neighbor algorithm. *IEEE Transactions on Systems Man & Cybernetics*. 2012; **15**(4):580-585
- [130] Fukunaga K, Hostetler L. k-nearest-neighbor Bayes-risk estimation. *IEEE Transactions on Information Theory*. 2003;**21**(3):285-293
- [131] Xu X, Frank E. Logistic regression and boosting for labeled bags of instances. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*; 2004. pp. 272-281
- [132] Von Luxburg U. A tutorial on spectral clustering. *Statistics & Computing*. 2007;**17**(4):395-416
- [133] Zhang S, Wong HS, Shen Y. Generalized adjusted rand indices for cluster ensembles. *Pattern Recognition*. 2012;**45**(6):2214-2226
- [134] Santos JM, Embrechts M. On the Use of the Adjusted Rand Index as a Metric for Evaluating Supervised Classification. Berlin/Heidelberg: Springer; 2009
- [135] Zhuang Y, Xu Z, Tang Y. A credit scoring model based on bayesian network and mutual information. In: *Web Information System and Application Conference*; 2016. pp. 281-286
- [136] Arthur D, Vassilvitskii S. k-means++: The advantages of careful seeding. In: *Soda 07. Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete algorithms, Society for Industrial and Applied Mathematics 2015*;**11**(6):1027-1035
- [137] Miroev I. k-Means Algorithm. 2017
- [138] Guo C, Zhu H. Analysis of large data sets based on k-means and k-medoids algorithm. *Journal of Taiyuan Normal University*. 2016;**15**(2):56-59
- [139] Park HS, Jun CH. A simple and fast algorithm for k-medoids clustering. *Expert Systems with Applications*. 2009;**36**(2):3336-3341
- [140] Johnson SC. Hierarchical clustering schemes. *Psychometrika*. 1967;**32**(3):241
- [141] Zhang T, Ramakrishnan R, Livny M. Birch: An efficient data clustering method for very large databases. In: *ACM SIGMOD International Conference on Management of Data*; 1996. pp. 103-114
- [142] Tu Y, Liu Y, Li Z. Online segmentation algorithm for time series based on birch clustering features. In: *International Conference on Computational Intelligence and Security*; 2010. pp. 55-59
- [143] Guha S, Rastogi R, Shim K, Guha S, Rastogi R, Shim K. Cure: An efficient clustering algorithm for large

- databases. *Information Systems*. 2001; **26**(1):35-58
- [144] Sander J. *Density-Based Clustering*. US: Springer; 2011
- [145] Casteleiro J, Lopes GP, Silva J. Bilingually learning word senses for translation. 2014;**8404**: 283-295
- [146] Delbary F, Knudsen K. Numerical nonlinear complex geometrical optics algorithm for the 3D calderon problem. *Inverse Problems & Imaging*. 2017;**8**(4): 991-1012
- [147] Burdette DL, Monroe KM, Sotelo-Troha K, Iwig JS, Eckert B, Hyodo M, Hayakawa Y, Vance RE. Sting is a direct innate immune sensor of cyclic DI-GMP. *Nature*. 2011; **478**(7370):515
- [148] Tokutaka H, Fujimura K, Yoshihara K. Applications of self-organizing maps (SOM) to the round-robin CoNi alloy spectra data. By that, is it possible to see the characteristics of the measurement instruments? *Hyomen Kagaku*. 1999; **20**(3):190-197
- [149] Kohonen T. An introduction to neural computing. *Neural Networks*. 1988;**1**(1):3-16
- [150] Hinton GE, Osindero S, Teh YW. A fast learning algorithm for deep belief nets. *Neural Computation*. 2006;**18**: 1527-1554
- [151] Krizhevsky A, Sutskever I, & Hinton GE. ImageNet classification with deep convolutional neural networks. *International Conference on Neural Information Processing Systems*. 2012; **60**:1097-1105
- [152] McCulloch WS, Pitts WH. A logical calculus of ideas imminent in nervous activity. *Bulletin of Mathematical Biophysics*. 1943;**9**:127-147
- [153] Kingma DP, Adam JB. A method for stochastic optimization. *Computer Science*. 2014;**arXiv**:1412.6980
- [154] Hornik K, Stinchcombe M, White H. *Multilayer Feedforward Networks are Universal Approximators*. Amsterdam, Netherlands: Elsevier Science Ltd.; 1989
- [155] Le Cun Y, Boser B, Denker JS, Howard RE, Hubbard W, Jackel LD, Henderson D. Handwritten digit recognition with a back-propagation network. In: *Advances in Neural Information Processing Systems*; 1990. pp. 396-404
- [156] Lecun Y, Bengio Y. Convolutional networks for images, speech, and time-series. In: *The Handbook of Brain Theory and Neural Networks*; 1995
- [157] Lecun Y, Bengio Y. *Convolutional Networks for Images, Speech, and Time Series*. Cambridge, Massachusetts, USA: MIT Press; 1998
- [158] He K, Zhang X, Ren S, Sun J. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis & Machine Intelligence*. 2015;**37**(9):1904-1916
- [159] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*. 2014;**15**(1):1929-1958
- [160] Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. *Nature*. 1986;**323**(6088):533-536
- [161] Saad Y. *Iterative Methods for Sparse Linear Systems*. Philadelphia, Pennsylvania, USA: SIAM; 2009
- [162] Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, WardeFarley D, Ozair S,

- Courville A, Bengio Y. Generative adversarial networks. *Advances in Neural Information Processing Systems*. 2014;**3**:2672-2680
- [163] Sutskever I, Vinyals O, Le QV. Sequence to sequence learning with neural networks. *Computer Science*. 2014;**4**:3104-3112
- [164] Hochreiter S, Schmidhuber J. Long short-term memory. *Neural Computation*. 1997;**9**(8):1735
- [165] Sutton RS, Barto AG. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks*. 2005;**16**(1): 285-286
- [166] Watkins CJCH. Learning from delayed rewards. *Robotics & Autonomous Systems*. 1989;**15**(4): 233-235
- [167] Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G. Human-level control through deep reinforcement learning. *Nature*. 2015;**518**(7540):529
- [168] Pang L, Li H, Wang Y. NMIBAS: A novel multi-receiver id-based anonymous signcryption with decryption fairness. *Computing and Informatics*. 2013;**32**(3):441-460
- [169] Pang L, Li H, Pei Q. Improved multicast key management of chinese wireless local area network security standard. *IET Communications*. 2012; **6**(9):1126-1130
- [170] Nali D, Adams CM, Miri A. Using threshold attribute-based encryption for practical biometric-based access control. *International Journal of Network Security*. 2005;**1**(3):173-182
- [171] Goyal V, Pandey O, Sahai A, Waters B. Attribute-based encryption for fine-grained access control of encrypted data. In: *Proceedings of the 13th ACM Conference on Computer and Communications Security*; 2006. pp. 89-98
- [172] Bethencourt J, Sahai A, Waters B. Ciphertext-policy attribute-based encryption. In: *Proceedings of the 2007 IEEE Symposium on Security and Privacy*; 2007. pp. 321-334
- [173] Ostrovsky R, Sahai A, Waters B. Attribute-based encryption with non-monotonic access structures. In: *Proceedings of the 2007 ACM Conference on Computer and Communications Security*; 2007. pp. 195-203
- [174] Lewko AB, Sahai A, Waters B. Revocation systems with very small private keys. In: *Proceedings of the 31st IEEE Symposium on Security and Privacy*; 2010. pp. 273-285
- [175] Attrapadung N, Libert B, de Panafieu E. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In: *Proceedings of the 14th International Conference on Practice and Theory in Public Key Cryptography*; 2011. pp. 90-108
- [176] Goyal V, Jain A, Pandey O, Sahai A. Bounded ciphertext policy attribute based encryption. In: *Proceedings of the 35th International Conference on Automata, Languages and Programming*; 2008. pp. 579-591
- [177] Waters B. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In: *Proceedings of the 14th International Conference on Practice and Theory in Public Key Cryptography*; 2011. pp. 53-70
- [178] Cheung L, Newport CC. Provably secure ciphertext policy ABE. In: *Proceedings of the 2007 ACM Conference on Computer and Communications Security*; 2007. pp. 456-465

- [179] Nishide T, Yoneyama K, Ohta K. Attribute-based encryption with partially hidden encryptor-specified access structures. In: Proceedings of the International Conference on Applied Cryptography and Network Security. 2008. pp. 111-129
- [180] Emura K, Miyaji A, Nomura A, Omote K, Soshi M. A ciphertext-policy attribute-based encryption scheme with constant ciphertext length. In: Information Security Practice and Experience-5th International Conference; 2009. pp. 13-23
- [181] Liang X, Cao Z, Lin H, Xing D. Provably secure and efficient bounded ciphertext policy attribute based encryption. In: Proceedings of the 2009 ACM Symposium on Information, Computer and Communications Security; 2009. pp. 343-352
- [182] Ibraimi L, Tang Q, Hartel PH, Jonker W. Efficient and provable secure ciphertext-policy attribute-based encryption schemes. In: Proceedings of the 5th International Conference on Information Security Practice and Experience; 2009. pp. 1-12
- [183] Lewko AB, Okamoto T, Sahai A, Takashima K, Waters B. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In: Proceedings of the 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques; 2010. pp. 62-91
- [184] Zhang J, Zhang Z. A ciphertext policy attribute-based encryption scheme without pairings. In: Proceedings of the 7th International Conference on Information Security and Cryptology; 2011. pp. 324-340
- [185] Sahai A. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: Proceedings of the 40th Annual Symposium on Foundations of Computer Science; 1999. pp. 543-553
- [186] Canetti R, Halevi S, Katz J. Chosen-ciphertext security from identity-based encryption. In: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques; 2004. pp. 207-222
- [187] Boneh D, Boyen X, Goh E-J. Hierarchical identity based encryption with constant size ciphertext. In: Proceedings of the 24th International Conference on the Theory and Applications of Cryptographic Techniques; 2005. pp. 440-456
- [188] Shoup V. Lower bounds for discrete logarithms and related problems. In: Proceedings of the International Conference on Theory and Application of Cryptographic Techniques; 1997. pp. 256-266
- [189] Bellare M, Rogaway P. Random oracles are practical: A paradigm for designing efficient protocols. In: Proceedings of the 1st ACM Conference on Computer and Communications Security; 1993. pp. 62-73
- [190] Schwartz JT. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*. 1980;27(4):701-717
- [191] Zippel R. Probabilistic algorithms for sparse polynomials. In: Proceedings of an International Symposium on Symbolic and Algebraic Computation; 1979. pp. 216-226
- [192] Fugkeaw S, Sato H. Scalable and secure access control policy update for outsourced big data. *Future Generation Computer Systems*. 2017; 79:364-373
- [193] Jakobsson M. On quorum controlled asymmetric proxy re-encryption. In: Proceedings of Second International Workshop on Practice and

Theory in Public Key Cryptography; 1999. pp. 112-121

[194] Zhou L, Marsh MA, Schneider FB, Redz A. Distributed blinding for distributed elgamal re-encryption. In: Proceedings of the 25th International Conference on Distributed Computing Systems; 2005. pp. 815-824

[195] Ivan A-A, Dodis Y. Proxy cryptography revisited. In: Proceedings of the Network and Distributed System Security Symposium; 2003

[196] Matsuo T. Proxy re-encryption systems for identity-based encryption. IACR Cryptology ePrint Archive. 2007; 2007: 361

[197] Shao J, Liu P, Wei G, Ling Y. Anonymous proxy re-encryption. Security and Communication Networks. 2012;5(5):439-449

[198] Kawai Y, Takashima K. Fully-anonymous functional proxy-re-encryption. IACR Cryptology ePrint Archive. 2013;2013: 318

[199] Deng H, Wu Q, Qin B, Susilo W, Liu JK, Shi W. Asymmetric cross-cryptosystem re-encryption applicable to efficient and secure mobile access to outsourced data. In: Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security; 2015. pp. 393-404

[200] Liang X, Cao Z, Lin H, Shao J. Attribute based proxy re-encryption with delegating capabilities. In: Proceedings of the 2009 ACM Symposium on Information, Computer and Communications Security; 2009. pp. 276-286

[201] Luo S, Hu J-B, Chen Z. Ciphertext policy attribute-based proxy re-encryption. In: Proceedings of 12th International Conference on the Information and Communications Security; 2010. pp. 401-415

[202] Liang K, Au MH, Liu JK, Susilo W, Wong DS, Yang G, Yu Y, Yang A. A secure and efficient ciphertext-policy attribute-based proxy re-encryption for cloud data sharing. Future Generation Computer Systems. 2015;52: 95-108

[203] Jiang H, Shen F, Chen S, Li K-C, Jeong Y-S. A secure and scalable storage system for aggregate data in IOT. Future Generation Computing Systems. 2015; 49: 133-141

[204] Zhou Y, Deng H, Wu Q, Qin B, Liu J, Ding Y. Identity-based proxy re-encryption version 2: Making mobile access easy in cloud. Future Generation Computer Systems. 2016;62: 128-139

[205] Xu P, Jiao T, Wu Q, Wang W, Jin H. Conditional identity-based broadcast proxy re-encryption and its application to cloud email. IEEE Transactions on Computers. 2016;65(1):66-79

[206] Fiat A, Shamir A. How to prove yourself: Practical solutions to identification and signature problems. In: Proceedings of the Conference on the Advances in Cryptology; 1986. pp. 186-194

[207] Delerablée C, Paillier P, Pointcheval D. Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys. In: Proceedings of First International Conference on Pairing-Based Cryptography; 2007. pp. 39-59

[208] Delerablée C. Identity-based broadcast encryption with constant size ciphertexts and private keys. In: Proceedings of 13th International Conference on the Theory and Application of Cryptology and Information Security; 2007. pp. 200-215

[209] Boyen X, Mei Q, Waters B. Direct chosen ciphertext security from identity-based techniques. In: Proceedings of ACM Conference on Computer and Communications Security; 2005. pp. 320-329



*Authored by Zhen Qin, Erqiang Zhou, Yi Ding,
Yang Zhao, Fuhu Deng and Hu Xiong*

Mobile internet data has the characteristics of large scale, variety of patterns, and complex association. On the one hand, it needs an efficient data processing model to provide support for data services, and, on the other hand, it needs certain computing resources to provide data security services. Due to the limited resources of mobile terminals, it is impossible to complete large-scale data computation and storage. However, outsourcing to third parties may cause risks in user privacy protection. This monograph focuses on key technologies of data service outsourcing and privacy protection, including the existing methods of data analysis and processing, fine-grained data access control through effective user privacy protection mechanisms, and data sharing in the mobile internet.

Published in London, UK

© 2018 IntechOpen
© Kshavratskaya / iStock

IntechOpen

ISBN 978-1-83881-840-1



9 781838 818401