Toward Efficient and Secure Public Auditing for Dynamic Big Data Storage on Cloud

by

Chang Liu

B. Sci. (Shandong University)

M. Eng. (Shandong University)

A thesis submitted to

Faculty of Engineering and Information Technology

University of Technology, Sydney

for the degree of

Doctor of Philosophy

December 2014

To my family and friends

CERTIFICATE OF ORIGINAL AUTHORSHIP

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Chang Liu

Production Note:
Signature removed
prior to publication.

21 July 2015

Acknowledgements

I sincerely express my deepest gratitude to my primary supervisor, A/Prof. Jinjun Chen, for his seasoned supervision and continuous encouragement throughout my PhD study. Prof. Chen was a consistent inspiration and taught me a great deal about how to become a good researcher and a good person. I also thank my associate supervisors Dr. Rajiv Ranjan, Prof. Yun Yang and Prof. Chengfei Liu for their whole-hearted supervision and continuous support of my study.

I thank the Australian Research Council (ARC), Australia Commonwealth Scientific and Industrial Research Organization (CSIRO) and the University of Technology, Sydney (UTS) for offering me a full research scholarship throughout my doctoral program. I also thank the CSIRO Digital Productivity Flagship (DPF) and the Research Committee of the UTS Faculty of Engineering and Information Technology (FEIT) for research publication funding support and for providing me with financial support to attend conferences.

My thanks also goes to staff members and researchers at UTS FEIT, CSIRO DPF, Swinburne University of Technology and Nanjing University for their help, suggestions, friendship and encouragement, particularly: Prof. Igor Hawryszkiewycz, A/Prof. Maolin Huang, Indrawati Nataatmadja, Xuyun Zhang, Chi Yang, Miranda Qian Zhang, Adrian Johannes, Nazanin Borhan, A. Ali, Xiao Liu, Dong Yuan, Qiang He, Rui Zhou, Jianxin Li, Minyi Li, Wei Dong, Dahai Cao, Miao Du, Feifei Chen, Prof. Wanchun Dou, Wenmin Lin.

Last but not least, I am deeply grateful to my parents Hongbo Liu and Li Zheng for raising me, teaching me to be a good person, and supporting my studies abroad. Sadly, my dear father Hongbo passed away in 2013 during my PhD study. May he rest in peace.

Abstract

Cloud and Big Data are two of the most attractive ICT research topics that have emerged in recent years. Requirements of big data processing are now everywhere, while the pay-as-you-go model of cloud systems is especially cost efficient in terms of processing big data applications. However, there are still concerns that hinder the proliferation of cloud, and data security/privacy is a top concern for data owners wishing to migrate their applications into the cloud environment. Compared to users of conventional systems, cloud users need to surrender the local control of their data to cloud servers. Another challenge for big data is the data dynamism which exists in most big data applications. Due to the frequent updates, efficiency becomes a major issue in data management. As security always brings compromises in efficiency, it is difficult but nonetheless important to investigate how to efficiently address security challenges over dynamic cloud data.

Data integrity is an essential aspect of data security. Except for server-side integrity protection mechanisms, verification from a third-party auditor is of equal importance because this enables users to verify the integrity of their data through the auditors at any user-chosen timeslot. This type of verification is also named 'public auditing' of data. Existing public auditing schemes allow the integrity of a dataset stored in cloud to be externally verified without retrieval of the whole original dataset. However, in practice, there are many challenges that hinder the application of such schemes. To name a few of these, first, the server still has to aggregate a proof with the cloud controller from data blocks that are distributedly stored and processed on cloud instances and this means that encryption and transfer of these data within the cloud will become time-consuming. Second, security flaws exist in the current designs. The verification processes are insecure against various attacks and this leads to concerns about deploying these schemes in practice. Third, when the dataset is large, auditing of dynamic data becomes costly in terms of communication and storage. This is especially the case for a large number of small data updates and data updates on

multi-replica cloud data storage.

In this thesis, the research problem of dynamic public data auditing in cloud is systematically investigated. After analysing the research problems, we systematically address the problems regarding secure and efficient public auditing of dynamic big data in cloud by developing, testing and publishing a series of security schemes and algorithms for secure and efficient public auditing of dynamic big data storage on cloud. Specifically, our work focuses on the following aspects: cloud internal authenticated key exchange, authorisation on third-party auditor, fine-grained update support, index verification, and efficient multi-replica public auditing of dynamic data. To the best of our knowledge, this thesis presents the first series of work to systematically analysis and to address this research problem. Experimental results and analyses show that the solutions that are presented in this thesis are suitable for auditing dynamic big data storage on cloud. Furthermore, our solutions represent significant improvements in cloud efficiency and security.

The Author's Publications

Book Chapters:

1. **C. Liu**, R. Ranjan, X. Zhang, C. Yang and J. Chen, *A Big Picture of Integrity Verification of Big Data in Cloud Computing*, Handbook on Data Centers (Book), Springer, in press, 2014.

2. X. Zhang, **C. Liu**, S. Nepal, C. Yang and J. Chen, *Privacy Preservation over Big Data in Cloud Systems*, Security, Privacy and Trust in Cloud Systems (Book), Springer, in press, ISBN: 978-3-642-38585-8, 2013.

Journals:

3. **C. Liu**, R. Ranjan, C. Yang, X. Zhang, L. Wang and J. Chen, *MuR-DPA: Top-down Levelled Multi-replica Merkle Hash Tree Based Secure Public Auditing for Dynamic Big Data Storage on Cloud*, IEEE Transactions on Computers, accepted on 27 October, 2014

4. **C. Liu**, N. Beaugeard, C. Yang, X. Zhang and J. Chen, *HKE-BC: Hierarchical Key Exchange for Secure Scheduling and Auditing of Big Data in Cloud Computing*, Concurrency and Computation: Practice and Experience, accepted on 3 October, 2014

5. X. Zhang, W. Dou, J. Pei, S. Nepal, C. Yang, **C. Liu** and J. Chen, *Proximity-Aware Local-Recoding Anonymization with MapReduce for Scalable Big Data Privacy Preservation in Cloud*, IEEE Transactions on Computers, accepted on 18 August, 2014.

6. **C. Liu,** C. Yang, X. Zhang and J. Chen, *External Integrity Verification for Outsourced Big Data in Cloud and IoT: A Big Picture*, Future Generation

Computer Systems (FGCS), Elsevier, to appear, accepted on 16 August, 2014. doi: 10.1016/j.future.2014.08.007

7. W. Lin, W. Dou, Z. Zhou and **C. Liu**, *A Cloud-based Framework for Home-diagnosis Service over Big Medical Data*, Journal of Systems and Software (JSS), to appear, accepted on 22 May, 2013. (ERA Rank A)

8. C. Yang, **C. Liu**, X. Zhang, S. Nepal and J. Chen, *A Time Efficient Approach for Detecting Errors in Big Sensor Data on Cloud*, IEEE Transactions on Parallel and Distributed Systems (TPDS), to appear, accepted on 7 December, 2013. (ERA Rank A*)

9. **C. Liu**, J. Chen, L. T. Yang, X. Zhang, C. Yang, R. Ranjan and K. Ramamohanarao, *Authorized Public Auditing of Dynamic Big Data Storage on Cloud with Efficient Verifiable Fine-grained Updates*, IEEE Transactions on Parallel and Distributed Systems (TPDS), vol. 25, no. 9, pp. 2234-2244, 2014. (ERA Rank A*)

10. X. Zhang, **C. Liu**, S. Nepal, C. Yang, W. Dou and J. Chen, *A Hybrid Approach for Scalable Sub-Tree Anonymization over Big Data using MapReduce on Cloud*, Journal of Computer and System Sciences (JCSS), vol. 80, no. 5, pp. 1008–1020, 2014. (ERA Rank A*)

11. C. Yang, X. Zhang, C. Zhong, **C. Liu**, J. Pei, K. Ramamohanarao and J. Chen, *A Spatiotemporal Compression based Approach for Efficient Big Data Processing on Cloud*, Journal of Computer and System Sciences (JCSS), to appear, 2013. (ERA Rank A*)

12. C. Yang, X. Zhang, **C. Liu**, J. Pei, K. Ramamohanarao and J. Chen, *A Spatiotemporal Compression based Approach for Efficient Big Data Processing on Cloud*, Journal of Computer and System Sciences (JCSS), to appear, 2013. (ERA Rank A*)

13. X. Zhang, **C. Liu**, S. Nepal, C. Yang, W. Dou and J. Chen, SaC-FRAPP: *A Scalable and Cost-effective Framework for Privacy Preservation over Big Data on Cloud*, Concurrency and Computation: Practice and Experience (CCPE), vol. 25, no. 18, pp. 2561-2576, 2013. (ERA Rank A)

14. X. Zhang, L. T. Yang, **C. Liu** and J. Chen, *A Scalable Two-Phase Top-Down Specialization Approach for Data Anonymization using MapReduce on Cloud*, IEEE Transactions on Parallel and Distributed Systems (TPDS), 25(2): 363-373, 2014. (ERA Rank A*)

15. X. Zhang, **C. Liu**, S. Nepal and J. Chen, *An Efficient Quasi-identifier Index based Approach for Privacy Preservation over Incremental Data Sets on Cloud*, Journal of Computer and System Sciences (JCSS), 79(5): 542-555, 2013. (ERA Rank A*)

16. X. Zhang, **C. Liu**, S. Nepal, S. Panley and J. Chen, *A Privacy Leakage Upper-bound Constraint based Approach for Cost-effective Privacy Preserving of Intermediate Datasets in Cloud*, IEEE Transactions on Parallel and Distributed Systems (TPDS), vol. 24, no. 6, pp. 1192-1202, 2013. (ERA Rank A*)

17. **C. Liu**, X. Zhang, C. Yang and J. Chen, *CCBKE - Session Key Negotiation for Fast and Secure Scheduling of Scientific Applications in Cloud Computing*, Future Generation Computer Systems (FGCS), Elsevier, vol. 29, no. 5, pp. 1300-1308, 2013. (ERA Rank A)

Conferences:

18. **C. Liu**, R. Ranjan, X. Zhang, C. Yang, D. Georgakopoulos and J. Chen, *Public Auditing for Big Data Storage in Cloud Computing -- A Survey*, in Proc. The 16th IEEE International Conference on Computational Science and Engineering (CSE 2013), pp. 1128-1135, December, 2013, Sydney, Australia.

19. C. Yang, **C. Liu**, X. Zhang, S. Nepal and J. Chen, *Querying Streaming XML Big Data with Multiple Filters on Cloud*, in Proc. The 2nd International Conference on Big Data and Engineering (BDSE 2013), pp. 1121-1127, December, 2013, Sydney, Australia.

20. X. Zhang, C. Yang, S. Nepal, **C. Liu**, W. Dou and J. Chen, *A MapReduce Based Approach of Scalable Multidimensional Anonymization for Big Data Privacy Preservation on Cloud*, in Proc. 3rd International Conference on Cloud and Green Computing (CGC 2013), pp: 105 - 112, September, 2013, Karlsurhe, Germany.

21. **C. Liu**, X. Zhang, C. Liu, Y. Yang, R. Ranjan, D. Georgakopoulos and J. Chen, *An Iterative Hierarchical Key Exchange Scheme for Secure Scheduling of Big Data Applications in Cloud Computing*, in Proc. 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom2013), pp: 9-16, July, 2013, Melbourne, Australia. (ERA Rank A)

22. X. Zhang, **C. Liu**, S. Nepal, C. Yang, W. Dou and J. Chen, *Combining Top-Down and Bottom-Up: Scalable Sub-Tree Anonymization over Big Data using MapReduce on Cloud*, in Proc. 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom2013), pp: 501-508, July, 2013, Melbourne, Australia. (ERA Rank A)

23. X. Zhang, **C. Liu**, S. Nepal, W. Dou and J. Chen, *Privacy-preserving Layer over MapReduce on Cloud*, in Proc. 2nd International Conference on Cloud and Green Computing (CGC 2012), pp: 304-310, November, 2012, Xiangtan, China.

24. G. Zhang, Y. Yang, X. Zhang, **C. Liu** and J. Chen, *Key Research Issues for Privacy Protection and Preservation in Cloud Computing*, in Proc. 2nd International Conference on Cloud and Green Computing (CGC 2012), pp: 304-310, November, 2012, Xiangtan, China.

25. G. Zhang, Y. Yang, X. Zhang, **C. Liu** and J. Chen, *An Association Probability based Noise Generation Strategy for Privacy Protection in Cloud Computing*, in Proc. 10th International Conference on Service Oriented Computing (ICSOC2012), November, 2012, Shanghai, China. (ERA Rank A)

26. **C. Liu**, X. Zhang, J. Chen and C. Yang, *An Authenticated Key Exchange Scheme for Efficient Security-Aware Scheduling of Scientific Applications in Cloud Computing*, In Proc. 9th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC2011). pp: 372-379, December, 2011, Sydney, Australia.

27. X. Zhang, **C. Liu**, J. Chen and W. Dou, *An Upper-Bound Control Approach for Cost-Effective Privacy Protection of Intermediate Dataset Storage in Cloud*, In Proc. 9th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC2011). pp: 518-525, December, 2011, Sydney, Australia.

28. C. Yang, K. Ren, Z. Yang, P. Gong and **C. Liu**, *A CSMA-based Approach for Detecting Composite Data Aggregate Events with Collaborative Sensors in WSN*, In Proc. 15th International Conference on Computer Supported Cooperative Work in Design (CSCWD2011). pp: 489-496, June, 2011, Lausanne, Switzerland.

29. C. Yang, Z. Yang, K. Ren and **C. Liu**, *Transmission Reduction based on Order Compression of Compound Aggregate Data over Wireless Sensor Networks*, In Proc. 6th International Conference on Pervasive Computing and Applications (ICPCA2011). October, 2011, Port Elizabeth, South Africa.

Table of Contents

Figures

Tables

Chapter 1

Introduction

This thesis is concerned with developing efficient and secure public auditing schemes for dynamic big data storage in cloud. A suite of novel frameworks, strategies, algorithms and protocols is designed and developed with the support of new concepts, solid theorems and innovative algorithms. Theoretical analyses and experimental evaluation demonstrates that our work helps to dramatically bring down overheads and effectively improves the security of public auditing schemes in the cloud.

This chapter introduces the background and key issues of this research. The chapter is organised as follows. Section 1.1 gives a brief introduction to big data and cloud. Section 1.2 presents the key research issues around security and privacy in big data and cloud. Section 1.3 outlines the research problems in the public auditing area along with the problems we try to address in this thesis. Section 1.4 provides an overview of the remainder of this thesis.

1.1 Big Data and Cloud Computing

In recent years, big data has become one of the most attractive research topics in information technology. People from almost all major industries are increasingly realising the value of their explosively growing datasets. Primary examples of big data applications may be seen in the areas of government, manufacturing, media, science and research. Research challenges in big data are always summarised into 4 V's: Velocity, Variety, Veracity and Volume. Velocity means big data is always in a dynamic status and flowing at a high speed; Variety means there are various types of

data in big data storage; Veracity indicates the uncertainty of big data; and Volume indicates that the size of big data storage is always at a large scale -- 40 Zeta bytes of data is estimated to be created by 2020, an increase of 300 times from 2005 . Besides this, there is another V -- Value, which is considered to be a fundamental aspect of the other V's. Within the explosively growing datasets, there are almost limitless value that is being discovered by the developing data mining techniques (Wu et al., 2014). All in all, it can be seen within these 5 V's that efficiency is an essential factor in big data processing, and cloud can help in a big way with all of the various challenges.

Cloud computing is a new-generation distributed computing platform that is extremely useful for big data storage and processing. Many big data applications as mentioned earlier are being migrated or have been migrated into clouds. One of the cloud's core concepts is 'X as a Service' (XaaS), including Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS), which means that both individual and enterprise users can use IT services in a pay-as-you-go fashion. Compared to traditional distributed systems, this new concept of cloud computing brings outstanding advantages. First, a considerable amount of investment is saved because there is no need for users to purchase and maintain their own IT facilities. Second, it brings exceptional elasticity, scalability and efficiency for task executions, especially in big data applications (Agrawal et al., 2011, Chaudhuri, 2012). With its virtualisation technology, pay-as-you-go payment model and elastic and scalable resource allocation of XaaS, cloud computing is widely recognised as the most promising technological backbone for solving big data related problems (Armbrust et al., 2010). Indeed, it is envisaged that cloud computing, with its capacity to provide computational resources, can one day be integrated into our daily life as closely as other resource utilities such as electricity, gas and water (Buyya et al., 2009). The exceptional scalability and elasticity of cloud make it the ideal platform for processing big data streams and handling the complexities of big data applications.

Datasets in big data applications are always dynamic. In fact, except for a few

examples of large static datasets such as those in libraries and e-archives, datasets in most big data applications need constant updating. In many big data applications, such as in social networks and business transactions, data updates are often very frequent. Therefore it is of extreme importance for cloud security mechanisms, such as the public auditing schemes studied in this thesis, to be able to support dynamic data updates in a secure and efficient way.

1.2 Security and Privacy Concerns in Cloud

Security/privacy is one of the major concerns in terms of the utilisation of cloud computing (Mather et al., 2009, Subashini and Kavitha, 2010). As data is no longer under the users' direct control, users are reluctant to move their valuable data onto the cloud -- especially the public cloud with its high consolidation and multi-tenancy. Also, from an efficiency perspective, querying and retrieving from cloud servers require a lot more effort than it does in local servers. Amongst the many technological aspects, the three main dimensions of data security research are confidentiality, integrity and availability.

In this thesis, we will focus on data integrity which is concerned with ensuring that data is stored and maintained in its original form. In practice, integrity breaches are not only caused by deliberate malicious attacks, but also uncontrollable disasters or server/disk failures. Integrity verification and protection is an active research area; numerous research problems belonging to this area have been studied intensively in the past. The three different aspects form an organic whole. While our focus is integrity verification, we will also focus on ensuring that efficiency, confidentiality and availability is incorporated in our designs.

1.3 Public Auditing of Dynamic Cloud Data

Aiming at integrity assurance, public auditing of cloud data has been an extensively investigated research problem in recent years. As user datasets stored on

cloud storage servers (CSS) are out of the cloud users' reach, auditing from the client herself or a third party auditor is a common request, no matter how secure and powerful the server-side mechanisms claim to be. With provable data possession (PDP) and proofs of retrieveability (POR), the data owner or a third-party auditor is able to verify the integrity of their data without having to retrieve their data. In such schemes, a small piece of metadata called 'homomorphic authenticator' or 'homomorphic tags' are stored along with each data block. When the client needs to verify data integrity, the server will generate a proof with the authenticators of the selected data blocks, and data auditing is done by the client or a third-party auditor through verifying the proof with public keys.

As stated above, the majority of datasets in big data applications are dynamic. Therefore, it is of great importance for public auditing schemes to be scalable and capable of supporting dynamic data updates. Existing public auditing schemes can already support verification of various kinds of full dynamic data updates (Wang et al., 2011b, Liu et al., 2014b). However, there are security and efficiency problems that we aim to address in our research. Some issues investigated in this thesis are stated as follows.

1) Efficiency issues. First, with virtualisation technology, existing key exchange schemes such as Internet Key Exchange (IKE) becomes time-consuming when directly deployed in the background of the cloud computing environment (i.e., the encrypted communications between virtualised cloud servers and the cloud controller), especially for large-scale computing tasks that involve intensive user-cloud interactions such as public data auditing. In order to construct an aggregated integrity proof, the cloud service provider needs to retrieve authenticators from different storage instances. Upon recognising this concern, we developed key exchange algorithms in the cloud background which sought to provide efficient server-side processing. Second, there is inefficiency in processing small updates because the existing authenticated data structures (ADS) only support whole-block insertion, deletion and modification, and they lack the ability to support

updates with an arbitrary length and starting offset. To achieve this, we designed a novel scheme utilising a flexible data segmentation strategy and a weighted Merkle hash tree, based on the detailed definition and analysis on fine-grained updates. Third, not much work has been done in supporting multiple replicas. Storing multiple replicas is a common strategy for data reliability and availability in the cloud. For highly dynamic data, each update will lead to updates of every replica. Given the fact that update verifications in current verification schemes are of O(log$n$) communication complexity, verifying these replicas one by one will be very costly in terms of communication. Accordingly, we developed a multi-replica public auditing scheme based on a novel multi-replica Merkle hash tree.

2) Security issues. First, the challenge message is too simple which may enable malicious exploits in practice. To make the public auditing scheme more secure and robust, we address this problem by adding an additional authorisation process among the three participating parties of the client, CSS and a third-party auditor (TPA). Second, current schemes for dynamic public auditing are susceptible to attacks from dishonest servers due to the lack of support for verification of block indices. To address this problem, we developed a novel public auditing scheme with a new ADS which incorporates authentication of level and rank information.

The research problems we try to address are analysed in detail under a systematic lifecycle in Chapter 3, and our solutions are presented in the consecutive chapters. The main contributions of this thesis are summarised as follows.

1) This thesis proposes a thorough investigation into the research problem of dynamic public auditing for big data in cloud, and presents a systematic framework along with a series of algorithms incorporating analysis and experimental results. To the best of our knowledge, this is the first sustained work to systematically analyse and address this research problem.

2) Two novel key exchange schemes for encrypted cloud background communications are presented. The schemes are based on the Diffie-Hellman key

exchange scheme. Experimental results provided in Section 7 have shown that key exchange plays an essential role in the efficiency of secure public auditing and data transfer as a whole. Moreover, analysis and experimental results have also shown that the newly proposed schemes greatly outperform the existing scheme without compromising the level of security.

3) We present a public auditing scheme based on the BLS signature and Merkle hash tree (MHT) that can support fine-grained update requests. For the first time, we formally analyse different types of fine-grained dynamic data update requests on variable-sized file blocks in a single dataset. Compared to existing schemes, our scheme supports updates with a size that is not restricted by the size of file blocks. It thereby offers extra flexibility and scalability compared to existing schemes. Also, for better security, our scheme incorporates an additional authorisation process with the aim of eliminating threats of unauthorised audit challenges from malicious or pretending third-party auditors.

4) To address the efficiency problem in verifiable updates for cloud storage with multiple replicas, we present a multi-replica public auditing (MuR-DPA) scheme which is based on a novel rank-based multi-replica Merkle hash tree (RMR-MHT). To support full dynamic data updates and authentication of block indices, we have included rank and level values in the computation of MHT nodes. Experimental results show that our scheme can drastically reduce communication overheads for the update verification of cloud data storage with multiple replicas. Also, as the previous usage of the Merkle hash tree (MHT) in public auditing of dynamic data does not involve authentication of node indices, such schemes are susceptible to cheating behaviours from a dishonest server. With the support of RMR-MHT, we propose the first MHT-based dynamic public auditing scheme with authentication of index information that is secure against dishonest servers. The main strategy is top-down levelling and the verification of indices from both sides.

1.4 Thesis Overview

This thesis presents novel data structures, algorithms and concepts with solid theorems and analyses to form a series of systematic solutions to address the research problem of the efficient and secure public auditing of big data storage in cloud. The thesis structure is depicted in Figure 1-1.

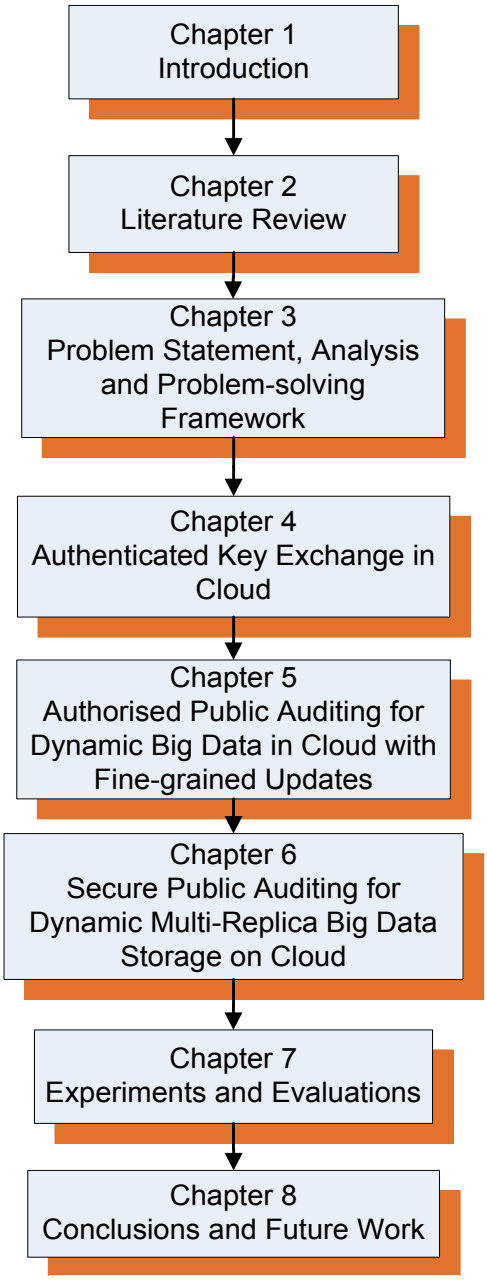


Figure 1-1 Thesis structure.

In Chapter 2, a comprehensive literature review of existing research is provided. Specifically, this literature review includes current research on: 1) Cloud computing, big data applications, 2) Security and privacy of cloud data, 3) Authenticated key exchange, and 4) Integrity verification mechanisms: digital signatures, and our focus on authenticator-based public verification.

In Chapter 3, a detailed analysis of the series of research problems we try to address in this thesis is provided, where representative approaches are also introduced. Lastly, I present a framework and lifecycle to systematically address these problems, and then outline the problems that will be addressed in the thesis.

In Chapter 4, I present server-side key exchange schemes which aim at supporting efficient proof generation in the public auditing of cloud data. First, we propose a key exchange scheme based on the randomness-reuse strategy and Internet Key Exchange (IKE) scheme, named CCBKE, for efficient and secure data transfer in the background of the cloud. Second, we propose a novel hierarchical key exchange scheme, namely Hierarchical Key Exchange for Big data in Cloud (HKE-BC), where we have developed a two-phase layer-by-layer iterative key exchange strategy to achieve more efficient AKE without sacrificing the level of data security. These key exchange schemes will also benefit other security mechanisms that involve symmetric encryptions, such as security-aware scheduling. Security and efficiency analyses for the new schemes are also provided.

In Chapter 5, I present the first dynamic public auditing scheme that supports fine-grained updates, authorisation of third-party auditors and public auditing at the same time. I provide a formal analysis for possible types of fine-grained data updates and propose a scheme that can fully support authorised auditing and fine-grained update requests. Based on this scheme, I also propose an enhancement that can dramatically reduce communication overheads for verifying small updates. After scheme description, I provide security and efficiency analysis for the new schemes.

In Chapter 6, I present research on efficient auditing for dynamic big data

storage with multiple replicas. The new scheme incorporates a novel authenticated data structure based on the Merkle hash tree (MHT), which is named RMR-MHT. For support of full dynamic data updates and authentication of block indices, rank and level values in computation for MHT nodes are included. As opposed to existing schemes, level values of nodes in RMR-MHT are assigned in a top-down order, and all replica blocks for each data block are organised into the same replica sub-tree for efficient verification of updates for multiple replicas. Security and efficiency analyses for this scheme are also presented in this chapter.

In Chapter 7, I present experimental results and analysis for our schemes to quantitatively demonstrate our research contributions. I firstly introduce U-Cloud, the cloud computing environment which is used for all the experiments. Then, I show the results from the experiments conducted on U-Cloud. The efficiency improvements of our schemes are demonstrated by comparing the computation time, storage overheads and communication overheads with existing schemes or their direct extensions.

Finally, in Chapter 8, I summarize the research presented in this thesis, major contributions of all the presented modules that constitute this thesis, and future research directions.

In order to improve the accessibility of this thesis, I also provide a list of acronyms in Appendix A and a notation index in Appendix B.

Chapter 2

Literature Review

In Chapter 2, I will provide a comprehensive literature review on existing research and highlight their respective problems. This chapter is organised as follows. Section 2.1 discusses the existing research on cloud computing and big data applications, along with the security and privacy of cloud data in general. Section 2.2 discusses existing research on integrity verification mechanisms including digital signatures and our focus on authenticator-based public verification. Section 2.3 discusses authenticated key exchange which greatly impacts the server-side performance of public auditing schemes and is also important for all security systems that involve symmetric-key encryption.

2.1 Security and Privacy Research in Cloud and Big Data

Big data is one of the most popular research topics in recent years (Agrawal et al., 2011, Chaudhuri, 2012, 2013, Wu et al., 2014). Examples for big data applications are social networks (Naone, 28 September, 2010), scientific research applications (Keahey et al., 2008, Heath, 2012), real-time streaming data, big sensor data (Yang et al., 2013a, Cuzzocrea et al., 2013, Yang et al., 2013b) and data in the Internet of Things (Ma et al., 2012), etc. In recent years, the development of distributed systems, or alternatively, cyberinfrastructures (Wang and Fu, 2010), has been the main platform for the processing of large-scale big data tasks such as scientific applications. To address this kind of big data problems, the cloud is currently considered to be the most potent and cost-effective platform. Currently, cloud computing is already being widely utilised for large-scale computation tasks in big data processing because of its outstanding cost-effectiveness (Agrawal et al., 2011,

Chaudhuri, 2012).

As the most popular paradigm among recently emerging hybrid environments, cloud has a significant cost advantage compared to traditional distributed systems such as clusters and grids (Armbrust et al.). Scientific applications can utilise this advantage by migrating to the cloud (Deelman et al., 2008), which is attracting rapidly growing research interest. Recent cloud computing projects for scientific applications such as Nimbus (Keahey et al., 2008) and Aneka (Vecchiola et al., 2012) as well as some very recent research work (Iosup et al., 2011, Srirama et al., 2012) all aim at the transformation from a traditional cluster or data centre to a cloud architecture. Since the advent of cloud computing, a number of scheduling algorithms have been developed for the purposes of achieving a cost-effective cloud computing environment. The most recent examples are the work of Garg et.al (Garg et al., 2011) and Yuan et.al (Yuan et al., 2011); the former work assesses the time and cost in cloud QoS, while the latter work investigates the trade-off between data storage, computation and economical cost in the cloud. However, neither has yet taken into account the cost of security enhancement. Data flows are unprotected in their models and this means that data security is totally neglected. Some outstanding security issues in cloud computing have been surveyed in (Subashini and Kavitha, 2010, Zissis and Lekkas, 2011, Mather et al., 2009).

Data security/privacy, which represents an important metric of QoS, are of great concern for cloud users (Zhang et al., 2013c, Mather et al., 2009, Zhang et al., 2013b). Therefore, data security/privacy constitute some of the most pressing concerns related to the cloud (Zissis and Lekkas, 2011, Schmidt, 2012, Yao et al., 2010) and big data. Generally speaking, as two sides of one coin, privacy and security aim at different goals, although both of them roughly aim at the protection of data content. Privacy research mainly aims at protect the data user's sensitive information but this technical solution is only one aspect amongst many non-technical aspects include policy, legislation, etc.. Technical data privacy research can be divided into syntactic privacy and differential privacy (Dwork, 2008,

Fan and Xiong, 2012, Clifton and Tassa, 2013). Analysis shows that these two areas are irreplaceable by each other (Clifton and Tassa, 2013). Privacy challenges in cloud data include parallel and distributed anonymization (Li et al., 2008, Roy et al., 2010, Zhang et al., 2011, Zhang et al., 2013c, Zhang et al., 2013b), hiding of data relations through encryption (Zhang et al., 2013a), hiding of access patterns through oblivious RAM (Williams et al., 2012, Stefanov et al., 2013, Stefanov and Shi, 2013), etc.. Data security research, on the other hand, is mainly technical, definitive and does not depend on data content (privacy goals can also be definitive (Wang et al., 2013c) in some cases). Encryptions to ensure data confidentiality and digital signatures for data integrity verifications are typical examples.

Data security is a widespread concept that can be everywhere in a computer and network system, especially for a complex distributed computer system such as the cloud. For example, Zhao et, al. have designed a security framework for big data applications in the cloud (Zhao et al., 2014). Along with time, cost and throughput etc., security can also be considered an important aspect of QoS. Great efforts have been placed on cost-efficient scheduling for distributed computing systems including cloud computing (Tang et al., 2011, Young Choon Lee, 2011). For example, the work of Lee et.al (Young Choon Lee, 2011) focuses on efficient scheduling with low energy consumption in cloud computing and distributed systems. However, they have not taken into account the additional costs in enhancing data security. There is also research on security-aware scheduling schemes. For example, Tang et.al (Tang et al., 2011) proposes a cost-effective security-aware scheduling algorithm for distributed systems. Although their approach achieves high efficiency by grading data security into several levels, their scheme, in fact, compromises security for higher efficiency, and it suffers from the inherent cost-inefficiency of existing KE schemes. By investigating key exchange efficiency problems, we are therefore making solid steps toward not only efficient auditing, but also security-aware QoS and scheduling for cloud computing.

2.2 Integrity Verification and Public Auditing

In the past, intensive research has been undertaken to enhance cloud data security/privacy with technological approaches on the cloud server side, such as (Liu et al., 2012, Zhang et al., 2012). They are of equal importance to external verification approaches such as our focus of public auditing . Although in this thesis we focus on public auditing based on homomorphic authenticators (Yang and Jia, 2012, Liu et al., 2014d, Liu et al., 2013a), please also note that there are other auditing methods, for example, log-based database auditing (Hwang et al., 2014, Lu et al., 2013).

Integrity verification for outsourced data storage has attracted extensive research interest. The concept of proofs of retrievability (POR) and its first model was proposed by Jules et, al. (Juels and B. S. Kaliski, 2007). Unfortunately, their scheme can only be applied to static data storage such as an archive or library. In the same year, Ateniese et, al. proposed a similar model named 'provable data possession' (PDP) (Ateniese et al., 2007). Their schemes offers 'blockless verification' which means the verifier can verify the integrity of a proportion of the outsourced file through verifying a combination of pre-computed file tags which they call homomorphic verifiable tags (HVTs) or homomorphic linear authenticators (HLAs). Work by Shacham et, al. (Shacham and Waters, 2008) proposed the first public verification scheme in the literature that is based on the BLS signature scheme (Boneh et al., 2004). In this scheme, the generation and verification of integrity proofs are developed from signing and verification of BLS signatures. When wielding the same security strength (say, 80-bit security), a BLS signature (160 bit) is much shorter than an RSA signature (1024 bit), which in turn brings a shorter proof size for a POR scheme. They have also provided an improved POR model with stateless verification. and proved the security of both their schemes and the PDP scheme by Ateniese et, al. (Ateniese et al., 2007, Ateniese et al., 2011). The security model for auditing schemes is further analysed in (Yu et al., 2014b). Ateniese et, al.

extended their scheme for enhanced scalability (Ateniese et al., 2008), but only partial data dynamics and a predefined number of challenges is supported.

Although the schemes discussed above can support blockless verification and public verifiability, data dynamics is not supported. Erway et, al. proposed the first PDP scheme that can support verification for full dynamic data updates (Erway et al., 2009) while retaining blockless verifiability. A modified authenticated data structure (ADS) is used for verification of updates, which became a common way of supporting verifiable updates in the following PDP/POR works. The ADS they used is called rank-based authenticated skip list (RASL). However, public auditability and variable-sized file blocks are not supported in their framework. Yang et, al. proposed a scheme (Yang and Jia, 2013) that claims to support secure public verifiability over dynamic data. However, there are a number of problems in this work. Ni et, al. have shown that their scheme is not secure against an active adversary (Ni et al., 2014). Furthermore, no index verification is provided. Therefore, a dishonest sever is able to manipulate the tags and original data to cheat the client. Wang et, al. (Wang et al., 2011b) proposed a scheme based on the BLS signature that can support public auditing (especially from a third-party auditor, TPA) and full data dynamics. To support verification of updates, they used another ADS called the Merkle hash tree (MHT). However, their usage of ADS also has security problems regarding the non-existence of authentication of block indices. A follow-up work by Wang et al. (Wang et al., 2010) added a random masking technology on top of (Wang et al., 2011b) to ensure the TPA cannot infer the raw data file from a series of integrity proofs. In their scheme, they also incorporated a strategy first proposed in (Shacham and Waters, 2008) to segment file blocks into multiple ‘sectors’ for trading-off of storage and communication costs. However, none of the above schemes has considered the commonly employed multi-replica strategy in clouds. For availability, data in the cloud is usually stored with multiple replicas distributed on multiple servers. Curtmola et, al. proposed a scheme named MR-PDP (Curtmola et al., 2008) that can prove the integrity of multiple replicas along with the original data file. Although the scheme requires only one authenticator for each block, it has two severe

drawbacks. First, since the verification process requires secret material, there will be security problems when extending the MR-PDP scheme to support public auditing. Second, it does not support verification for dynamic data updates. In order to allow a third-party auditor to verify datasets with multiple replicas without any secret material, the client still needs to store and build different ADS for every replica, which will incur heavy communication overheads. As an improvement to MR-PDP, Barsoum et, al. proposed a series of PDP schemes (Barsoum and Hasan, 2011, Barsoum and Hasan, 2012). These schemes are based on the BLS signature with support of public verifiability, data dynamics and multiple replicas at the same time. However, they do not provide a verification process for updates. Furthermore, their construction of the MHT structure is not efficient for update verifications as each single update will incur updates on all branches. Etemad et, al. proposed a multi-replica PDP scheme (Etemad and Küpçü, 2013a, Etemad and Küpçü, 2013b) with index authentication based on Erway et, al.'s RASL. However, their scheme does not support public verifiability. Furthermore, their efficiency analysis is mainly about computation time, and does not include discussion of the important efficiency factors of communication overheads.

Research in this area also includes the work of Ateniese et, al. (Ateniese et al., 2009) on how to transform a mutual identification protocol to a PDP scheme; and a scheme by Zhu et, al. (Zhu et al., 2012) which allows different service providers in a hybrid cloud to cooperatively prove data integrity to the data owner (a security problem was later found in this scheme, as indicated in (Wang and Zhang, 2014)). As cloud data sharing is happening in many scenarios, Wang et, al. worked on secure data verification of shared data storage (Wang et al., 2013a, Wang et al., 2014) and also with efficient user management (Wang et al., 2013b) and user privacy protection (Wang et al., 2012, Wang et al., 2014). Zhang et, al. proposed a scheme with a new data structure called the update tree (Zhang and Blanton, 2012, Zhang and Blanton, 2013). Without conventional authenticated data structures such as MHT, the proposed scheme has a constant proof size and fully supports data dynamics. However, the scheme also does not support public auditing. Yuan et, al. proposed a

public auditing scheme with a de-duplication property (Yuan and Yu, 2013). Hanser et, al. proposed a robust public auditing scheme based on elliptic curves (Hanser and Slamanig, 2013). Cash et, al. (Cash et al., 2013) proposed a novel POR scheme based on oblivious RAM (ORAM). ORAM, or oblivious file system which was mostly used to hide data access patterns through shuffling and noise addition on outsourced data storage (Stefanov et al., 2013, Williams et al., 2012). Shi et, al. also proposed a more efficient scheme based on ORAM (Shi et al., 2013), but practical usage of such schemes is still under investigation. After our contributions in this thesis are proposed in publications, there are a number of new developments in public auditing for outsourced data (Yu et al., 2014b, Yu et al., 2014a, Worku et al., 2014, Hwang et al., 2014, Camenisch et al., 2014). This indicates that the problem has not been completely solved. Accordingly, this area of research remains very active and attractive to computer scientists.

2.3 Authenticated Key Exchange in Cloud

Recently, much research work has been done to address cloud security/privacy issues. Most of the approaches, however, focus on cloud storage service. Yao et.al (Yao et al., 2010) propose a scheme to ensure cloud storage security by separating the encryption keys from the stored data which are encrypted by the keys. In (Cao et al., 2011), a privacy-preserving cloud data querying scheme is generally proposed from a data prospect of view, which aims to protect privacy-sensitive outsourced data. In (Wang et al., 2011b) Wang et.al propose a scheme to protect data integrity based on bilinear-pairing-based public-key cryptology, which allows a third-party authority to check the outsourced data on the cloud service users' behalf. However, to the best of our knowledge, none has analysed data encryption on the backstage of cloud computing from an efficiency point of view. Our work starts to try and bridge the gap.

Encryption-based data security protection approaches for cloud *storage* services have been proposed, such as (Yao et al., 2010, Puttaswamy et al., 2011). However, in a cloud *computing* environment, original data input needs to be processed on the cloud side. Some encryption schemes allow processing over encrypted data, and it can still be decrypted using the same key thereafter. However, in these schemes only limited operations are allowed. For example, the approach in (Wong et al., 2009) only allows the *k*-NN (*k* nearest neighbours) algorithm to be applied over encrypted data. Fully homomorphic encryption (Gentry, 2009) allows all operations on an encrypted dataset; However, no homomorphic encryption scheme with reasonable complexity has yet been published. Therefore, the efficiency of key exchange still remains a major obstacle to efficiency overall.

Key exchange over a distrustful communication environment has been an extensively researched problem in public-key cryptography since Diffie and Hellman proposed the very first key exchange scheme (Diffie and Hellman, 1976) in 1976. Our problem here, essentially, is looking for an efficient key exchange scheme for exchanging different keys over participating parties. Although the topic of extending

key exchange schemes in multi-party scenarios has been studied a lot in the past, to the best of our knowledge, this problem has not been well-addressed. This is probably because there was no requirement for doing research on this problem in the past, as a single key exchange process takes almost no time on modern hardware facilities. However, with thousands of independent instances executing different tasks in the cloud, it is now essential to develop an optimised key exchange scheme under this scenario if we are to use hybrid encryption to enhance cloud data security. Some existing key exchange schemes try to optimise the multi-party-same-key scenario with users joining and leaving dynamically, such as (Bresson et al., 2002, Zhou and Huang, 2010, Katz and Shin, 2005, Katz and Yung, 2007). In (Küsters and Tuengerthal, 2009, Zhao and Gu, 2010), extended security standards are formalised and researched for key exchange schemes for the basic 2-party scenario. Some of the most recent research on authenticated key exchange schemes focus on password-based key exchange (Groce and Katz, 2010, Katz and Vaikuntanathan, 2011), which allows two parties to share a session key through exchanging a low-entropy password. These approaches are advantageous when humans are involved because a low-entropy password can be easily remembered. Unfortunately, no human action is required in our research problem. Many existing key exchange schemes try to optimise the multi-party-same-key scenario as we do. Kurosawa (Kurosawa, 2002) and then Bellare et.al (Bellare et al., 2003) studied the problem of asymmetric-key encryption in the multi-user-different-data scenario with the randomness reuse strategy. Due to the low efficiency of asymmetric-key encryption over large datasets, their schemes cannot be directly applied into cloud computing environments. However, this problem is essentially the KE problem in the background of the cloud. IKE is a widely-adopted authenticated key exchange scheme which is used along with IPSec as the default network-level data protection standard in TCP/IP Suite. The latest updated description of IKE can be found in (Kaufman et al., September 2010). It is known for its high security level (Canetti and Krawczyk, 2002), but it lacks efficiency in distributed environments, especially in the cloud. We developed a key exchange scheme named CCBKE for security-aware

scheduling in cloud computing (Liu et al., 2013c). When deployed in cloud, CCBKE invokes significantly lower time consumption compared to IKE, but it still takes a considerable amount of time. Therefore, we developed another KE scheme for further performance evaluation . We will introduce these schemes in Chapter 4.

Chapter 3

Background, Problem Analysis and Framework

In this chapter, I will provide framework and detailed analysis on the research problems we aim to address in this thesis. This chapter is organized as follows. Section 3.1 introduces preliminaries for the remainder of this thesis. Section 3.2 provides motivating examples and the framework of our research. Section 3.3 provides a brief introduction of existing representative public auditing approaches for the purposes of background knowledge. Section 3.4 presents problem analysis for the specific research problems that are addressed in this thesis.

3.1 Preliminaries

I will now introduce preliminaries in presenting the research in the area of public auditing on cloud data. The preliminaries include the Diffie-Hellman key exchange, RSA signature, bilinear pairing, BLS signature and authenticated data structure. Most of them are the foundation stones for not only public auditing schemes, but also cryptography and information security research in general.

3.1.1 Diffie-Hellman Key Exchange

The Diffie-Hellman key exchange scheme presented in 1976 (Diffie and Hellman, 1976) is commonly considered to be the earliest key exchange protocol, and the beginning of the public-key cryptography era. For two users, Alice and Bob,

sharing an insecure communication channel, they can communicate to establish a shared secret key with the protocol. Its security is based on the computational difficulty of the discrete logarithm problem.

The public keying materials are a big integer p and one of its primitive root g, i.e., g is a generator of group $\mathbb{Z}_p$. For key exchange, Alice will choose its secret material x and send $g^x \mathrm{mod} p$ to Bob, while Bob will choose its secret material y and send $g^y \mathrm{mod} p$ to Alice. Through these procedures, Alice can obtain the key $g^{xy} \mathrm{mod} p$ through $(g^y)^x$, and Bob can also obtain $g^{xy} \mathrm{mod} p$ through $(g^x)^y$. If p is sufficiently large, compute x with $g^x \mathrm{mod} p$ is computationally impossible. As x and y are kept secret, any third party cannot figure out the exchanged key through communication sniffing.

3.1.2 RSA Signature

The RSA signature is a classic and one of the earliest signature schemes; it is also one of the foundation stones of public-key cryptography. Its security is based on the computational difficulty of the factoring problem. While the textbook version is not semantically secure and not resilient to existential forgery attacks, there is a large body of research work on its improvements later on, and this ultimately makes it a robust signature scheme. For example, a basic improvement is to use $h(m)$ instead of m where h is a one-way hash function.

The setup is based on an integer $N = pq$ where p and q are two large primes, and two integers d and e where $ed = 1 \bmod N$; d is kept as the secret key and e is the public key. The signature σ of a message m is computed as $\sigma = m^d \bmod N$. Along with m, the signature can be verified through verifying whether the equation $m = \sigma^e \bmod N$ holds.

3.1.3 Bilinear Pairing and BLS Signature

Assume a group G is a gap Diffie-Hellman (GDH) group with prime order p. A bilinear map is a map constructed as $e: G \times G \rightarrow G_T$ where G_T is a multiplicative cyclic group with prime order. A useful e should have the following properties: bilinearity – $\forall\, m, n \in G \Rightarrow e(m^a, n^b) = e(m, n)^{ab}$; non-degeneracy – $\forall m \in G, m \neq 0 \Rightarrow e(m, m) \neq 1$; and computability – e should be efficiently computable. For simplicity, we will use this symmetric bilinear map in our scheme description. Alternatively, the more efficient asymmetric bilinear map $e: G_1 \times G_2 \rightarrow G_T$ may also be applied, as was pointed out in (Boneh et al., 2004).

BLS signature is proposed by Boneh, Lynn and Shacham (Boneh et al., 2004) in 2004. In addition to the basic soundness of digital signature, this scheme has a greatly reduced signature length, but also increased overheads due to the computationally expensive paring operations. Its security is based on the gap Diffie-Hellman problem on bilinear maps. Based on a bilinear map $e: G \times G \rightarrow G_T$, a basic BLS signature scheme works as follows. Keys are computed as $y = g^x$ where $g \in G$, x is the secret key and $\{g, y\}$ is the public key. Signature sig for a message m is computed as $sig = (h(m))^x$. People can then verify this signature through verifying whether $e(sig, g) = e(h(m), y)$.

3.1.4 Authenticated Data Structures

Authenticated data structures (ADS) are used to efficiently verify data position through verification of all data in the verification path from the root. It is employed in integrity verification schemes to enable the verifier to check whether the storage server has performed the data update correctly. Now we briefly introduce ADS used in integrity verification. Iterative hashing is the core idea in these ADS's; their security is based on pre-image resistance, second pre-image resistance and the collision resistance of the chosen cryptographic hash function.

Merkle Hash Tree (MHT) (Merkle, 1987) is an authenticated data structure which has been intensively studied in the past and later utilised to support verification of dynamic data updates. Similar to a binary tree, each node N will have a maximum of 2 child nodes. Information contained in one node N in a MHT T is $\mathcal{H}$ -- a hash value. T is constructed as follows. For a leaf node LN based on a message m_i, we have $\mathcal{H} = h(m_i)$, $r_{LN} = s_i$; A parent node of $N_1 = \{\mathcal{H}_1, r_{N1}\}$ and $N_2 = \{\mathcal{H}_2, r_{N2}\}$ is constructed as $N_P = \{h(\mathcal{H}_1||\mathcal{H}_2)\}$ where $\mathcal{H}_1$ and $\mathcal{H}_2$ are information contained in N_1 and N_2 respectively. A leaf node m_i's AAI Ω_i is defined as a set of hash values chosen from all of its upper level (only one per level) so that the root value R can be computed through $\{m_i, \Omega_i\}$. For example, for the MHT demonstrated in Fig. 3-1, m_1's AAI $\Omega_1 = \{h(m_2), h(e), h(b)\}$.

Rank-based authenticated skip list (RASL) (Erway et al., 2009) is an authenticated data structure that can be authenticated not only through the content, but also the indices of the data block. Based on this structure, Erway et, al. proposed the first PDP scheme that can support full dynamic data operations. An example can also be found in Fig. 3-1, where the 'rank' value of a node is defined as the maximum number of leaf nodes it can reach. Its average complexity is also logarithmic to the number of blocks, similar to MHT.

There are also other authenticated data structures. Mo et, al. designed Merkle B+ tree (Mo et al., 2012) which also has logarithm complexity for updates. Our work on fine-grained updates (Liu et al., 2014b) have proposed the ranked Merkle hash tree (RMHT) for fine-grained updates. However, the rank value is not for authentication of indices, but for authentication of variable block sizes. To make this concept clear, we will term it the weighted Merkle hash tree (WMHT) and introduce it in detail in Chapter 5. In Chapter 6, we will also introduce another authenticated data structure, namely the rank-based multi-replica Merkle hash tree (RMR-MHT), designed for efficient index verification and the update of multiple replicas.

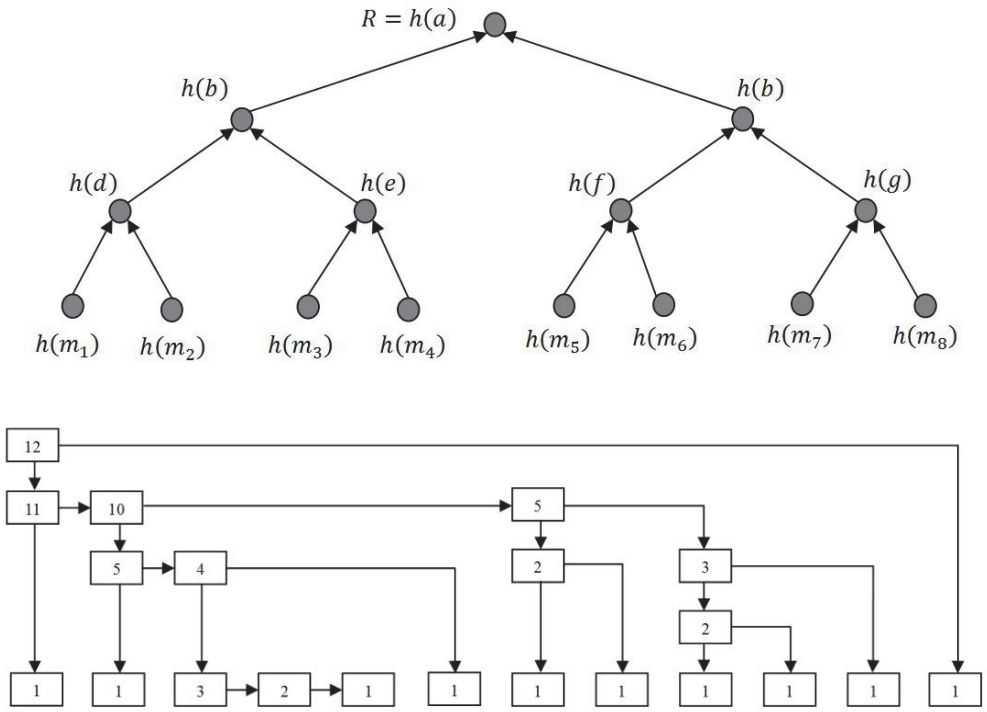


Figure 3-1 ADS examples: MHT and RASL.

3.2 Motivating Examples and Research Framework

3.2.1 Motivating Examples

Big data and cloud computing are currently receiving more and more attention from both industry and academia. They have been recently listed as important strategies by the Australian Government (Department of Finance and Deregulation, 2013, Department of Finance and Deregulation, 2011). One example in big data applications is sensor data and surveillance data gathered from sensors and surveillance cameras. Each individual sensor has the potential to stream back gigabytes of data per second. Much of this data is "unstructured" or "semi-structured," which means it is difficult or costly to either store in traditional data warehouses or routinely query and analyse. This data tsunami phenomenon is being described as a new grand challenge in computing: The 'Big Data' problem, which is defined as the practice of collecting complex data sets so large that it becomes difficult to analyse

and interpret manually or use on-hand data management applications (e.g., Microsoft Excel). Mining industry faces big challenges in quickly manipulating large volumes of data and mining them for relevant information. Supporting rapid decision making for key operations must be obtained in real time. These operations in the mining industry, for example, include yield modelling, production optimisation, fleet optimisation, and mine collapse detection. Hence, efficient Information and Communication technologies (ICT) that store, distribute, index, and analyse hundreds of petabytes of heterogeneous data streaming from a variety of sensors are needed -- in a way that does not compromise QoS in terms of data availability, data search delay, data analysis delay, and the like.

Big data applications are always data-intensive and time-critical. Data from scientific research is another important source of big data. Here is an example in astrophysics. Australian astrophysics researchers operate a gigantic 64-metre Parkes telescope which generates a large amount of data through constant observation. Scientists usually need to access the results as early as possible, as a late-coming result may cause an enormous waste of resources and loss of scientific discovery. An example in astrophysics is gravitational wave detection (Kawata et al., 2007) which is especially time-critical. Due to its nature of real-time and streaming, delay in returning a result of a task may cause missed detection of an incoming gravitational wave; thousands of hours of data-intensive computation would be in vain, which is a terrible waste both economically and environmentally. Online web service is another example. Although there are less computational tasks in normal web services than in scientific applications, user requests normally demand the servers' response in a few seconds, such as in the search engine, etc.. Hence, efficiency is of extreme importance in cloud scheduling for most big data applications.

To address big data problems, cloud computing is believed to be the most potent platform. In Australia, big companies such as Vodafone Mobile and News Corporation are already moving their business data and its processing tasks to Amazon cloud - Amazon Web Services (AWS) (2012). Email systems of many

Australian universities are using public clouds as the backbone. To tackle the large amount of data in scientific applications, CERN, for example, is already putting the processing on petabytes of data into cloud computing (Heath, 2012). There has also been a lot of research regarding scientific cloud computing, such as in (Vecchiola et al., 2012, Wang et al., 2011a, Wang et al., 2008). For big data applications within cloud computing, data security is a problem that should always be properly addressed. In fact, data security is one of the biggest reasons why people are reluctant in using cloud (Schmidt, 2012, Yao et al., 2010, Zissis and Lekkas, 2011). Therefore, more effective and efficient security mechanisms are direly in need to help people establish their confidence in all-round cloud usage.

Cost-efficiency brought by elasticity is one of the most important reasons why cloud is being widely adopted for processing big data applications. For example, Vodafone Australia is currently using the Amazon cloud to provide their users with mobile online-video-watching services. According to their statistics, the number of video requests per second (RPS) can reach an average of over 700 during less than 10% of the time such as Friday nights and public holidays, compared to a mere 70 on average for the rest (i.e. 90%) of the time. The variation in demand is more than 9 times (2012). Without cloud computing, Vodafone cannot avoid purchasing computing facilities that can process 700 RPS, but it would be a total waste for most of the time. This is where cloud computing can save a significant amount of expense -- cloud's elasticity allows the user-purchased computation capacity to scale up or down on-the-fly at any time. Therefore, user requests can be fulfilled without wasting investment on computational powers. Two other large companies who own *news.com.au* and *realestate.com.au*, respectively, are using the Amazon cloud for the same reason (2012). We can see through these cases that scalability and elasticity are of extreme importance for the processing of the big data application in cloud computing. As stated above, efficiency, as an important factor of QoS, must not be compromised. Support for data dynamics is an important aspect in these examples. Therefore, the capability and efficiency in supporting data dynamics is essential for big data applications in the cloud, which is also applicable to public auditing

mechanisms for cloud data storage.

Many big data applications will keep user data stored on the cloud for small-sized but very frequent updates. A typical example is Twitter, where each tweet is restricted to 140 characters long (which equals 140 bytes in ASCII code). They can add up to a total of 12 terabytes of data per day (Naone, 28 September, 2010). Storage of transaction records in banking or securities markets is a similar and more security-heavy example. Moreover, cloud users may need to split large-scale datasets into smaller chunks before uploading to the cloud for privacy-preservation (He et al., 2011) or efficient scheduling (Yuan et al., 2010). In this regard, efficiency in processing small updates will affect the performance of many big data applications. To better support scalability and elasticity of cloud computing, some recent public data auditing schemes support data dynamics. However, the types of updates that are supported are limited. Therefore previous schemes may not be suitable for some practical scenarios. Besides, there is a potential security threat in the existing schemes. We will discuss these problems in detail in the following chapters.

To sum up, the motivation for many research works on cloud data security is the fact that data is stored in cloud and away from the user's direct control. Therefore, cloud data is susceptible to more types of malicious attacks, including the malicious cloud server. As in conventional systems, data security in cloud and big data is also an endless game of attack and defend with constant evolvements in the spear and shield. Therefore, there is always the possibility for new security exploits to be discovered from conventional security models. In this thesis, the main focus is public data auditing. The desired properties discussed in these examples are the main design motivations and evaluation factors for our innovative schemes.

3.2.2 Research Problems with Public Auditing of Cloud Data -- Lifecycle and Framework

Data security includes many dimensions; the three main dimensions are confidentiality, integrity and availability. In this thesis, we will focus on data integrity. Data integrity means that data is needed to be maintained in its original form. Integrity verification and protection is an active research area; numerous research problems belonging to this area have been studied intensively in the past. As a result, the integrity of data storage can now be effectively verified in traditional systems through the deployments of Reed-Solomon code, checksums, trapdoor hash functions, message authentication code (MAC), digital signatures, etc. However, as stated above, the data owner (cloud user) still needs a method to verify their data stored remotely on a semi-trusted cloud server, no matter how secure the cloud claims to be. In other words, a cloud service provider must enable verifications from an external party that is independent to the cloud. The party could be the client herself, or a third party auditor. A straightforward approach is to retrieve and download from the server all the data the client wants to verify. Unfortunately, when data size is large, it is very inefficient in the sense of both time consumption and communication overheads. Moreover, when the client needs a third party to verify the data on her behalf, all data will be exposed to the third party. To address these problems, scientists are developing schemes based on traditional digital signatures to help users verify the integrity of their data without having to retrieve it, which they term as provable data possession (PDP) or proofs of retrievability (POR).

In this thesis, we will focus on integrity protection and verification from external parties, which we also term 'public auditing' of data. We only discuss the auditing to data itself; other auditing methods, such as log auditing (Hwang et al., 2014, Waters et al., 2004), are out of the scope of this thesis. There are 3 participating parties in an integrity verification game: client, CSS and TPA. The client stores her data on CSS, while TPA's objective is to verify the integrity of the

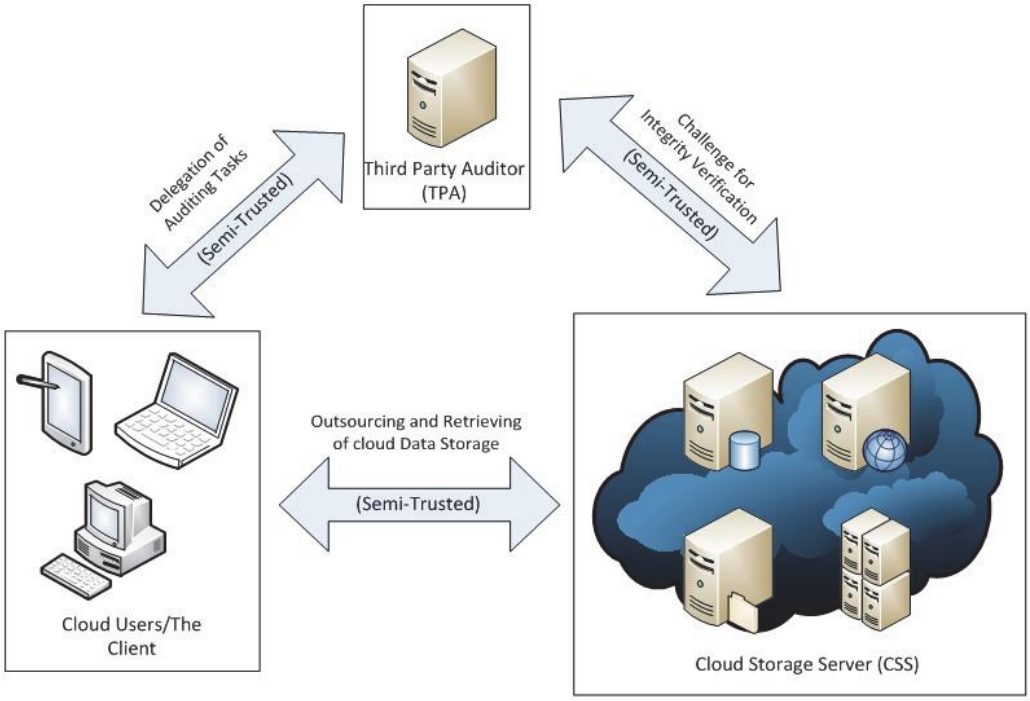


Figure 3-2 Participating parties in public auditing of cloud data.

Relations between the participating parties in public auditing of cloud data. The client authorises the TPA to audit data stored on CSS, where the three parties are not fully trusted by each other.

client's data stored on CSS. Having a specialised TPA to verify data integrity is more efficient, but it may also introduce additional risks as the third-party auditor may not be completely trustworthy by itself. Fig 3-2 shows the relations between the participating parties in public auditing, which demonstrates that the three parties in a public auditing game -- the client, the cloud service provider and third-party auditor -- do not fully trust each other. This has been a widely researched problem over recent years.

A framework of integrity protection on cloud data is presented in Fig. 3-3, where we can see ensuring data integrity can involve many aspects, ranging from internal and external verifications, data encryption and data anonymization. While this framework is only a wide static overview for the research area, we present a common lifecycle for the detailed dynamic process of a remote integrity verification

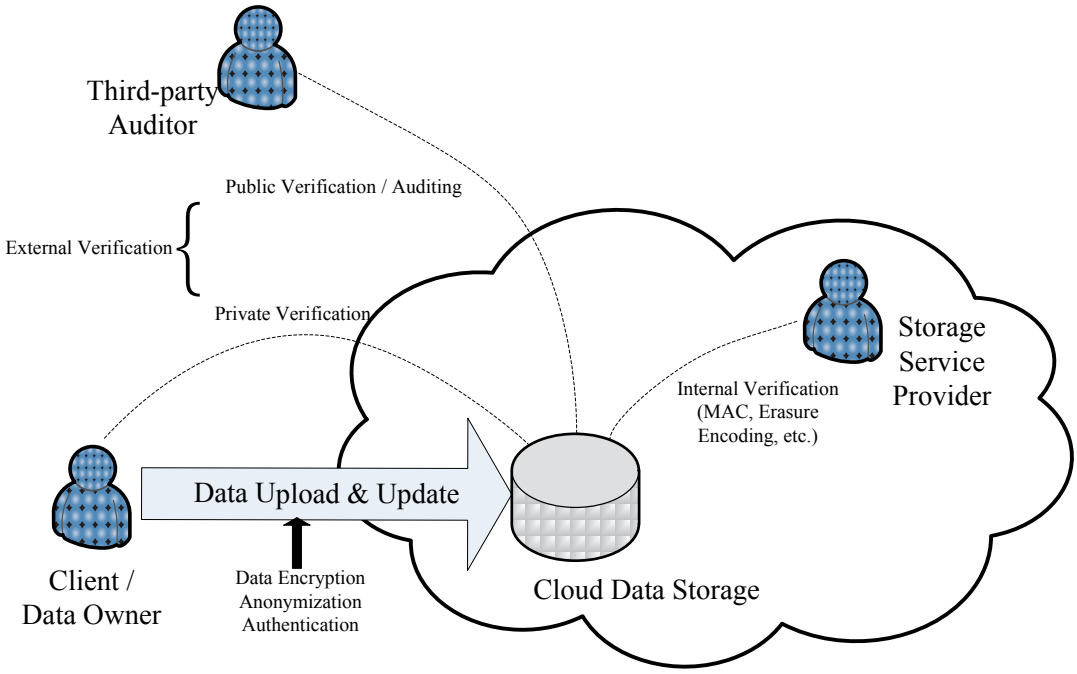


Figure 3-3 Integrity verification for outsourced data -- a framework

scheme (with support for dynamic data updates). Security of a public auditing scheme may be jeopardised in every step in the entire auditing process. In the mean time, efficiency of the entire auditing scheme will benefit from efficiency improvement at every step. Therefore, the lifecycle that describes every step of the verification process is essential for analysing the research problems in this area. The lifecycle can be analysed in the following steps: Setup and data upload; Authorization for TPA; Challenge for integrity proof; Proof integration; Proof verification; Updated data upload; Updated metadata upload; and Verification of updated data. The relationship and order of these steps are illustrated in Fig. 3-4. We now analyse in detail how these steps work and why they are essential to integrity verification of cloud data storage.

Setup and data upload: In cloud, the user data is stored remotely on CSS. In order to verify the data without retrieving it, the client will need to prepare verification metadata, namely homomorphic linear authenticator (HLA) or homomorphic verifiable tag (HVT), based on homomorphic signatures (Johnson et

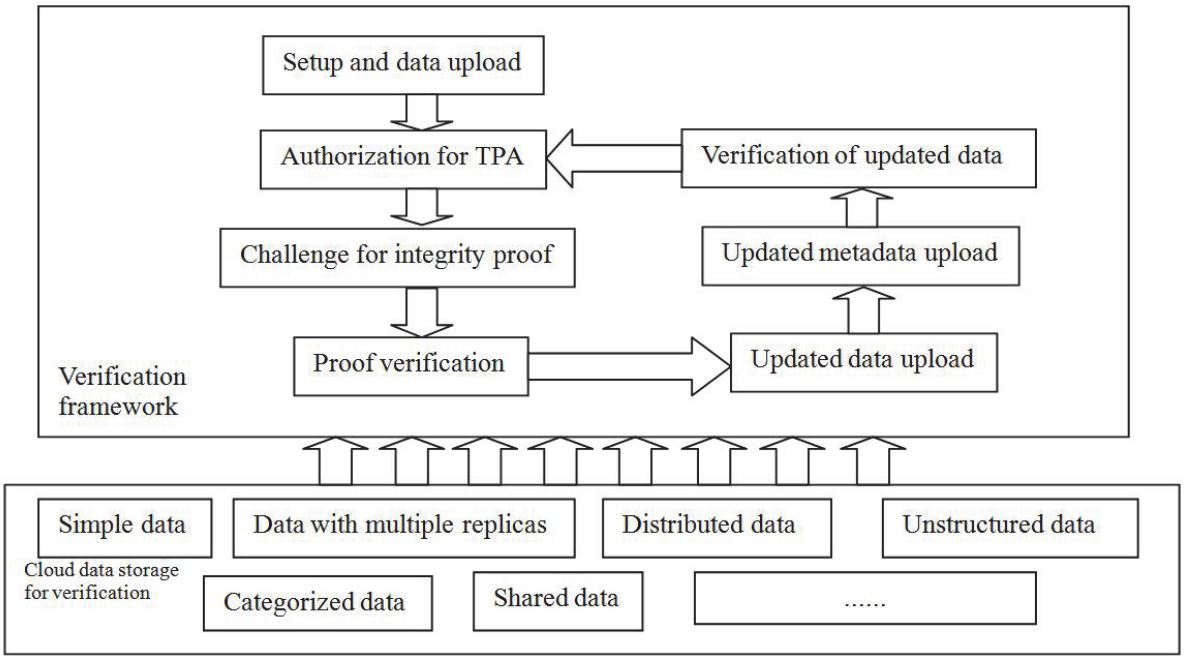


Figure 3-4 Integrity verification for outsourced data -- the lifecycle.

al., 2002). Then, the metadata will be uploaded and stored alongside the original datasets. These tags are computed from the original data; they must be small in size in comparison to the original dataset for practical use.

Authorisation for TPA: This step is not required in a two-party scenario where clients verify their data for themselves, but it is important when users require a semi-trusted TPA to verify the data on their behalf. If a third party can infinitely ask for integrity proofs over a certain piece of data, there will always be security risks in existence such as plaintext extraction. To address this problem, we present a novel solution in Chapter 5.

Challenge and verification of data storage: This step is where the main requirement -- integrity verification -- is fulfilled. The client will send a challenge message to the server, and the server will compute a response over the pre-stored data (HLA) and the challenge message. This response is computed and is based on all message blocks, which we call 'proof integration' in the cycle. The client can then verify the response to find out whether the data is intact. The scheme has public

verifiability if this verification can be done without the client's secret key. When data is dynamic and the auditing is from a third party, a malicious server may cheat the client with other healthy blocks when the challenged block is corrupted. We will discuss this further in Chapter 6. Note that if the data storage is static, the whole process would have ended here. However, as discussed earlier, data is always dynamic in many big data contexts (often denoted as velocity, one of the four v's). In these scenarios, we will need the rest of the steps to complete the lifecycle.

Data update: Occurs in dynamic data contexts. The client needs to perform updates to some of the cloud data storage. The updates could be roughly categorised as insertion, deletion and modification; if the data is stored in blocks with varying size for efficiency reasons, there will be more types of updates to address, which we will discuss in Chapter 5. Also, when there are multiple replicas in storage, one update will impact all replicas. We will discuss how to improve efficiency in Chapter 6.

Metadata update: In order to keep the data storage stay verifiable without retrieving all the data stored and/or re-running the entire setup phase, the client will need to update the verification metadata (HLA or HVT's) according to the existing keys.

Verification of updated data: This is also an essential step in the dynamic data context. As the CSS is not completely trusted, the client needs to verify the data update process to see if the updating of both user data and verification metadata has been performed successfully in order to ensure the updated data can still be verified correctly in the future. Note that communications inside the cloud will take place in both this step and the challenge/verification step for integrating a proof, where encryption and key exchange are needed to ensure security. We will discuss how to improve the efficiency of key exchange schemes inside the cloud without compromising security in Chapter 4.

We will show in Sections 3.3 and 3.4 how each step in this lifecycle was

developed and how it evolved. This will be done by analysing representative approaches in this research area and showing our contributions in the following chapters.

3.3 Representative Public Auditing Schemes In Sketch

Now we introduce and analyse some representative schemes. These schemes are basically presented in chronological order, and the scheme presented later will support improved properties which will be analyse at the end of this section. Note that all computations are within the cyclic group $\mathbb{Z}_p$ or $\mathbb{Z}_N$.

3.3.1 PDP

Proposed by Ateniese, et, al. in 2007, PDP (provable data possession) can provide authors with efficient verification over their outsourced data storage (Ateniese et al., 2007, Ateniese et al., 2011). It is the first scheme to provide blockless verification and public verifiability at the same time.

The tag construction is based on the RSA signature, therefore all computations are modulo N by default. Let N, e, d be defined as the same as in RSA signature (See Section 3.1), g is a generator of QR_N, and v is a random secret value; $\{N, g\}$ is the public key and $\{d, v\}$ is the secret key. The tag is computed as $\sigma_i = (h(v||i) \cdot g^{m_i})^d$. To challenge CSS, the client sends the indices (or, coordinates) of the blocks they want to verify, and correspondingly chooses a set of coefficients a_i, as well as a $g_s = g^s \, mod \, N$ where s is a random number, and these are sent to CSS along with the indices. To prove data integrity, CSS will compute $\sigma = \prod_i \sigma_i^{a_i}$, along with a value $\rho = H(\prod_i g_s^{a_i m_i})$, and this will be sent back $\{\sigma, \rho\}$ as the proof. To verify this proof, the client (or TPA) will compute $= \frac{\sigma^e}{\prod_i h(v||i)^{a_i}}$, then verify if $\rho = H(\tau^s \, mod \, N)$.

The authors have also proposed a light version called E-PDP, in contrast to the

formal S-PDP scheme, for better efficiency. The basic idea is to throw away the coefficients a_i and compute the proof ρ as $\rho = H(\prod_i g_s^{m_i})$. The verification equation is therefore $\tau = \frac{\sigma^e}{\prod_i h(v||i)}$. However, the light version has proved to be not secure under the compact POR model. Nevertheless, as a milestone in this research area, a lot of settings continue to be used by the following work. Mixing in random coefficients is one of the examples. Another example is that the paper proposed a probability analysis and found that only a constant small number of blocks are to be verified, and if the client needs to have 95% or even 99% confidence, the integrity of the entire file is good. This analysis has also become a default setting in the following schemes.

3.3.2 Compact POR

Compact POR is proposed by Shacham, et, al. in 2008 (Shacham and Waters, 2008). Compared to the original POR, the authors have provided an improved rigorous security proof. The schemes they introduced in the paper also suit the PDP model.

They firstly proposed a construction for private verification. In this case, data can only be verified with the secret key, therefore no other party can verify it except for the client. The metadata HVT is computed as $\sigma_i = f_k(i) + \alpha m_i$, where $f_k()$ is a pseudo-random function (PRF). α and the PRF key k is kept as the secret key. When the server is challenged with a set of block coordinates and a set of corresponding public coefficients v_i (same definition as a_i in PDP above), it will compute $\sigma = \sum_i v_i \sigma_i$ and $\mu = \sum_i v_i m_i$ to return $\{\sigma, \mu\}$ as the proof. Upon receiving the proof, the client can simply verify if $\sigma = \alpha\mu + \sum_i (v_i f_k(i))$. The scheme is efficient because it admits short response length and fast computation.

The other construction with public verification is even more impressive compared to schemes at that time. It is the first BLS-based scheme that supports public verification. Due to the shortened length of BLS signature, the proof size is

also greatly reduced compared to RSA-based schemes. Similar to BLS signature, the tag construction is based on a bilinear map $e: G \times G \rightarrow G_T$ where G is a group of prime order p. Two generators g and u of $\mathbb{Z}_p$ are chosen to be the public key, as another value $v = g^{\alpha}$ where α is the secret key for the client. The tag is computed as $\sigma_i = (H(i)u^{m_i})^{\alpha}$,. Just as the one with private verification, a set of coefficients v_i is also chosen with the designated block coordinates. When challenged, the proof $\{\sigma, \mu\}$ is computed as $\sigma = \prod_i \sigma_i^{v_i}$ and $\mu = \sum_i v_i m_i$. The client can then verify the data integrity through verifying if $e(\sigma, g) = e(\prod_i (H(i)^{v_i}) \cdot u^{\mu}, v)$.

Another great contribution of this work is the rigorous security framework it provides. In their model, a verification scheme is secure only when it is secure against an arbitrary adversary with a polynomial extraction algorithm to reveal the message from the integrity proof. To prove the security, they also defined a series of interactive games under the random oracle model. Compared to the previous security frameworks in first PDP and first POR schemes, the adversary defined in this framework is stronger and stateless, and the definition of the extraction algorithm (therefore the overall soundness) is stronger. Also, their framework perfectly suits the public verification, and even the multi-replica storage and multi-prover scenarios. To date, this model is considered the strongest and it is very frequently used to prove the security of newly-proposed verification algorithms.

3.3.3 DPDP

DPDP (Dynamic PDP), proposed in 2009, is the first integrity verification scheme to support full data dynamics (Erway et al., 2009). It is from this scheme that the processes in integrity verification schemes started to form a self-closed lifecycle. They utilised another authenticated data structure -- rank-based authenticated skip list -- for verification of updates. A rank-based skip list is similar to MHT in the sense that they both incur a logarithm amount of operations when an update occurs. All types of updates -- insertion, deletion and modification -- are supported for the first time. This design is essentially carried on by all the following

schemes with dynamic data support. However, public verifiability was not supported by the scheme, and there was no follow-up work to fill the blank. Therefore, we only give a brief introduction here. The readers can refer to the next subsection to see how data dynamics is supported with an authenticated data structure such as MHT.

3.3.4 Public Auditing of Dynamic Data

As the DPDP scheme did not provide support for public verifiability, Wang, et, al. proposed a new scheme that can support both dynamic data and public verifiability at the same time (Wang et al., 2011b). They term the latter 'public auditability', as the verification is often done by a sole-duty third-party auditor (TPA). As this scheme offers no optimisation for auditing of multiple replicas for dynamic datasets, we name the scheme SiR-DPA (Single-Replica Dynamic Public Auditing).

A MHT is utilised to verify the updates where the root R is critical authentication information. The tree structure is constructed on blocks, and the structure is stored along with the verification metadata. Compared to compact POR, it computes the tags using $H(m_i)$ instead of $H(i)$ in order to support dynamic data, otherwise all tags in the following blocks must be changed upon each insertion or deletion update, which would be very inefficient. Aside from this, the tag construction and verification are similar: $\sigma_i = (H(m_i)u^{m_i})^{\alpha}$. The proof is also computed as $= \prod_i \sigma_i^{v_i}$ $\mu = \sum_i v_i m_i$. While the verification is to verify whether $e(\sigma, g) = e(\prod_i (H(m_i)^{v_i}) \cdot u^{\mu}, v)$, TPA will first verify $H(R)$'s signature to ensure the MHT is correct at the server side.

To verify data updates, the client will first generate the tag for the new block: $\sigma_i' = \left(H(m_i')u^{m_i'}\right)^{\alpha}$, then upload it to CSS along with the update request. CSS will update the metadata as requested, and send back R' along with the old block $H(m_i)$, the AAI Ω_i (note Ω_i will stay unchanged if m_i is the only block that has changed) and the client-signed old MHT root $H(R)$. The client can then verify the

signed $H(R)$ to ensure CSS has not manipulated it, then it can verify R' with m_i' and Ω_i to see if the update of data and metadata is correct. Apart from the main scheme, they also proposed a scheme that can perform efficient batch auditing with experimental results.

There was also a follow-up work to improve this scheme for privacy preserving public auditing (Wang et al., 2013c). When computing integrity proof, they added a random masking technique to prevent the part of the original file from being extracted from several integrity proofs over this specific part of the data.

Although it is one of the earliest works to support public auditability and data dynamics at the same time, there are still weaknesses which exist. The main weaknesses have been introduced in Section 3.3. We will show how these problems are addressed with our newly proposed public auditing schemes in the following chapters.

3.4 Detailed Analysis of Research problems

3.4.1 Authenticated Key Exchange in Cloud

As stated in Chapter 1, cloud users will need to audit their data in cloud servers during utilisation of cloud services, where efficiency is also an important factor. From a server's perspective, if a user wants to retrieve the data for verification, CLC must gather data blocks from distributed storage or virtualised instances, and integrate them to respond to the user. To ensure data confidentiality, the data also needs to be encrypted during transit between CLC and the cloud server, just the same as for task scheduling. Although current research allows users to verify data integrity without retrieving the dataset itself (Wang et al., 2011b, Ateniese et al., 2007, Liu et al., 2014b), the server still needs to compute an integrity proof based on the pre-stored authenticator and the dataset itself. For example, in (Wang et al., 2011b, Liu et al.,

2014b), a part of the proof is computed as $\mu = \sum v_i m_i$ where m_i are the cloud data blocks and v_i are random vectors selected by the verifier. In cloud, data blocks are distributedly stored on different storage servers. Therefore, CLC needs to retrieve data from their distributed locations (virtualised instances) and compute μ before it can respond to the auditing request, no matter it is from the user or a third-party auditor. For the same reason discussed before, this user data also needs to be encrypted. In public auditing schemes, while the verification itself is usually very fast because of pre-processing, efficiency in communications between CLC and instances becomes a predominant factor. At the same time, the data must be well-protected to avoid additional risks to data security within the auditing process itself. For these reasons, efficiency of key exchange between CLC and instances also greatly affects the overall efficiency of integrity verification and public auditing schemes, which is why this research is a part of this thesis.

The unique characteristics of virtualisation, consolidation and multi-tenancy bring unpredictable challenges to data security. For example, a malicious party can easily be another legitimate user who is using the same cloud and has even more opportunities for successful malicious behaviours (Ristenpart et al., 2009). As discussed in Section I, data in scientific research represents valuable intellectual property which can either be people's privacy-sensitive information or directly related to scientific discovery. Therefore, we suggest that all user data always stays encrypted in the cloud. Decryptions may only be applied right before data is used for task execution.

In a typical cloud computing infrastructure, a central server is employed for not only receiving and processing user requests at the front, but also being responsible for scheduling and splitting tasks through MapReduce at the back. This server is named cloud controller (CLC) in the Eucalyptus system (Nurmi et al., 2009), we will use this denotation in this paper. Virtualised server instances running on clusters of servers are responsible for processing the divided tasks in a parallel fashion and returning the results afterwards, and then CLC is capable of assembling the results and return to the

user. For more effective data management and processing, a structure of additional hierarchical levels is often employed between CLC and end server instances as well. We will discuss KE schemes in both the two-layered control structure and multi-layered structure in our CCBKE and HKE-BC schemes which will be presented in Chapter 4.

Because of encryptions, interaction between users and cloud servers in big data applications requires constant and repeated key exchange operations. As a result, a large percentage of time is devoted to the security system. As demonstrated in (Liu et al., 2013c), the standardised IKE key exchange scheme can take up to 76% of the total time consumption in the security system (depending on the actual parameters) when the size of user datasets and the number of instances involved are large. This is why we need to improve the efficiency of key exchange schemes. In every KE session, a distinct session key is needed for every virtual machine. This is because the risk of additional information being exposed against malicious users needs to be minimised. The existence of virtual machine hijacks (Ristenpart et al., 2009) further intensifies this risk. For example, if a single session key is utilised for data encryption on 100 virtualised instances, the information on all 100 nodes will all be exposed when only one of the instances is hijacked and the key is revealed. If we use different keys for different instances, the total information leakage will be reduced by 99%. For this reason, the computation cost and time consumption of key exchange operations in cloud are much more than those in other distributed computing systems.

Computations on server instances in key exchange processes can be completed almost instantly, because there is only one exponentiation needed for each instance. In addition, data communications in KE schemes via networking take almost no time because only kilobytes of data need to be transferred between the cloud controller and server instances in order to complete key exchange. Digital signatures are always necessary in key exchange schemes for identity authentication. In key exchange schemes, messages to be signed are usually of a short fixed length (typically 128 bits which is the output size of a hash-based message authentication code (HMAC)). In

this regard, time consumption in signing and verification of messages is negligible when compared to modular exponentiations over 1024-bit keying materials related to key exchanges. Based on this view, we know that the modular exponentiations in KE operations act as the predominant factor in the efficiency of a distributed KE scheme.

For scheduling purposes, a large-scale cloud computing infrastructure often employs a hierarchical control structure, which fits the philosophy of distributed storage and computation within the cloud. Following the acronyms defined in the early Eucalyptus cloud system (Nurmi et al., 2009), a typical cloud computing structure employs a CLC (cloud controller) as the interface between user and cloud, several CC (cluster controllers) for cluster control, a bunch of NC (node controllers) for virtualisation, and then virtualised instances for actual task execution. There are at least three layers for control, and the number of control layers can increase further with the scaling of the cloud environment (see Fig. 3-5 for an example of a hybrid cloud which consists of multiple clouds with multiple control layers). In CCBKE (Liu et al., 2013c) (which we also introduce in Chapter 4 in this thesis), CLC needs to perform all the KE operations for exchanging a distinct key for each instance, while the intermediate layers are required to do nothing other than pass messages. In this regard, the efficiency of KE will be further improved by re-designing the scheme to distribute the modular exponentiations to other control nodes.

3.4.2 Public Auditing of Verifiable Fine-grained Updates

Some of the existing public auditing schemes can already support full data dynamics (Erway et al., 2009, Wang et al., 2011b, Wang et al., 2010). In their models, only insertions, deletions and modifications on fixed-sized blocks are discussed. Particularly, in BLS-signature-based schemes (Wang et al., 2010, Wang et al., 2011b, Zhu et al., 2012, Shacham and Waters, 2008) with 80-bit security, the size of each data block is either restricted by the 160-bit prime group order p, as each block is segmented into a fixed number of 160-bit sectors. This design is inherently unsuitable to support variable-sized blocks, despite their remarkable advantage of

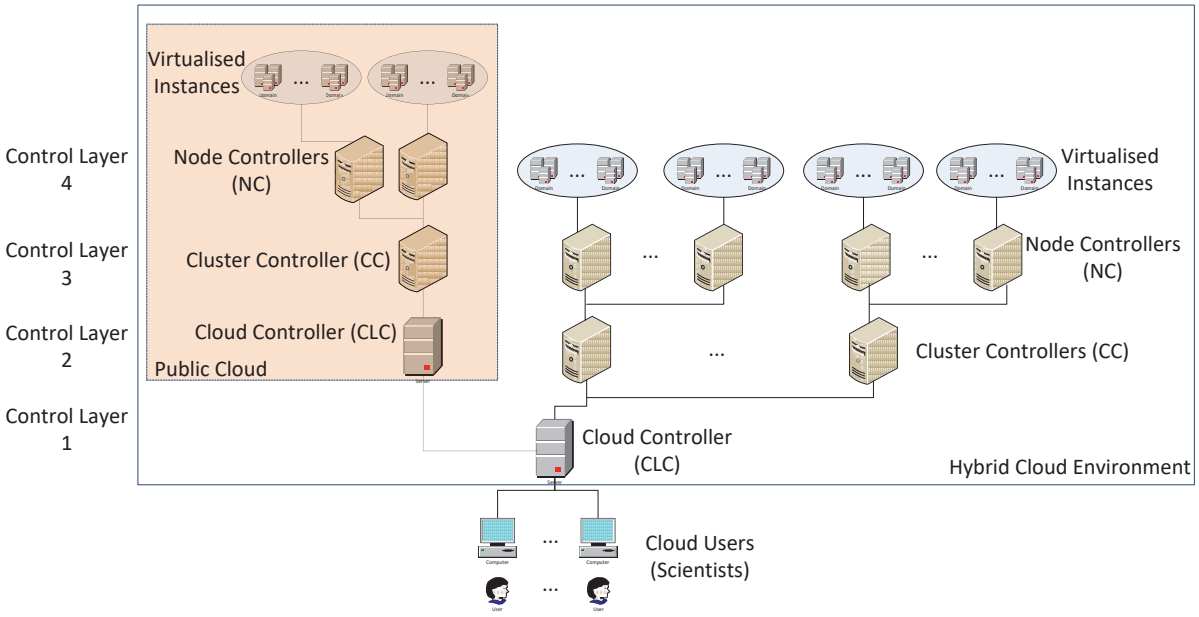


Figure 3-5 An example of hybrid cloud structures.

shorter integrity proofs. In fact, as described in Section 2, existing schemes can only support insertion, deletion or modification of one or multiple fixed-sized blocks, which we call 'coarse-grained' updates.

Although support for coarse-grained updates can provide an integrity verification scheme with basic scalability, data updating operations in practice can always be more complicated. For example, the verifiable update process introduced in (Wang et al., 2011b, Erway et al., 2009) cannot handle deletions or modifications in a size lesser than a block. For insertions, there is a simple extension that enables insertion of an arbitrary-sized dataset – CSS can always create a new block (or several blocks) for every insertion. However, when there are a large number of small upgrades (especially insertions), the amount of wasted storage will be huge. For example, in (Wang et al., 2011b, Erway et al., 2009) the recommended size for a data block is 16k bytes. For each insertion of a 140-byte Twitter message, more than 99% of the newly allocated storage is wasted -- they cannot be reused until the block is deleted. These problems can all be resolved if fine-grained data updates are supported. According to this observation, supporting fine-grained updates can bring

not only additional flexibility, but also improved efficiency. Details are provided when the FU-DPA scheme is introduced in Chapter 5.

3.4.3 Multi-replica Big Data in Cloud

For availability, storing multiple replicas is a default setting for cloud service providers. Storing replicas at different servers and/or locations will make user data easily recoverable from service failures. A straightforward way for users to verify the integrity of multiple replicas is to store them as separate files and verify them one by one. Currently, the most common technique used to support dynamic data is authenticated data structure (ADS). Given the $\log(n)$ communication complexity and storage complexity of ADS (n is the total number of blocks, a very large number when the file is large), there are different replicas. More importantly, an update for each data block will require an update of the corresponding block in every replica. If all replicas are indexed in their own separated ADS, the client must verify these updates one by one to maintain verifiability. The 'proof of update' for each block contains $\log(n)$ hash values as auxiliary authentication information (AAI). Therefore, the communication cost in update verifications will easily become disastrous for users whose cloud datasets are highly dynamic. In previous schemes, researchers have considered support for public auditing, data dynamics and efficient verification of multiple replicas, but none have considered them all together. In this work, we try to address this problem with a new ADS which links together all replicas for each block.

In (Curtmola et al., 2008), the authors proposed a multi-replica verification scheme, named MR-PDP, with great efficiency by associating only one authenticator (HLA) for each block and all replica blocks. Although this approach can bring great benefits such as lower storage cost at the server side and less pre-processing time at the client side, their scheme is not secure when replacing the verifier with a TPA. The verification process needs to privately keep the random padding values $r_{i,j}$ (or at least the pseudo-random function ψ_j that is used to generate them). If they are leaked, another party will know how to compute the original message based on any replica as

well as how to compute an arbitrary replica based on an original file block. To make things worse, if $r_{i,j}$ is known by the cloud server (or if there are collusions between cloud server and TPA), the cloud server will be able to fake an integrity proof of a given replica block based on any other replica block, even if the challenged replica block is corrupted. Therefore, the MR-PDP scheme is not secure in a setting with public verifications.

To sum up, from our considerations, desired properties of a multi-replica verification scheme should (simultaneously) include the following:

1. Public Auditability and Support for Dynamic Data -- Enables a third-party auditor to do the regular verification for the client without requiring any secret material, and allows the client to verify data updates. It will be unreasonable for the client to conduct verification herself on a regular basis, where she only wants to know when something went wrong with her data. Meanwhile, support for dynamic data is important as it exists in most big data applications.

2. All-round Auditing -- Enables efficient verification for all replicas at once so that the verifier will feel more confident. If any of the replicas fails, the server will be notified in time.

3. Single-Replica Auditing -- Enables verification for an arbitrary replica for some specific blocks; because the verifier may only want to know if at least one replica is intact for less important data.

3.4.4 Security of Public Auditing Schemes

Authorised Public Auditing

Fig 3-2 displays relations between the three parties in auditing games, where both CSS and TPA only semi-trust the client. In the old model, the challenge

message is very simple so that everyone can send a challenge to CSS for the proof of a certain set of file blocks, which can enable malicious exploits in practice. First, a malicious party can launch distributed denial-of-service (DDOS) attacks by sending multiple challenges from multiple clients at a time in order to cause additional overheads on CSS and congestion to its network connections, and thereby degeneration of service quality. Second, an adversary may get privacy-sensitive information from the integrity proofs returned by CSS. By challenging the CSS multiple times, an adversary can either get considerable information about user data (due to the fact that returned integrity proofs are computed with client-selected data blocks), or gather statistical information about cloud service status. To this end, traditional PDP models cannot quite meet the security requirements of 'auditing-as-a-service', even though they support public verifiability. This problem will be addressed in the FU-DPA scheme in Chapter 5.

Security Against Distrustful Server

Fig. 3-2 shows the relations between the participating parties in public auditing, which demonstrates that the three parties in a public auditing game -- the client, the cloud service provider and third-party auditor -- do not fully trust each other. Authenticated data structures (ADS) such as MHT or RASL can enable other parties to verify the content and updates of data blocks. The authentication for a block is accomplished with the data node itself and its auxiliary authentication information (AAI) which is constructed with node values on or near its verification path. Without verification of block indices, a dishonest server can easily take another intact block and its AAI to fake a proof that could pass authentication. This will cause several security holes. First, the proofs of updates are no longer reliable. A dishonest server can store a new data block anywhere, as long as it transfers back a consistent pair of hash $H(m_i)$ and AAI that can be used to compute the correct root value. Second, for auditing of dynamic data, $H(m_i)$, the hash value of the block itself, is needed in authenticator computation instead of a hash of any value that contains block indices such as $H(i)$ or $H(v||i)$, otherwise an insert/delete will cause changes to

authenticators of all the following blocks, which will be disastrous, especially if the client is the only one who can compute authenticators. Therefore, in order for each authenticator to include a block-specific hash value, $H(m_i)$ seems to be the only viable choice. In this case, as the verifier (client or TPA) does not possess the original dataset, the client will solely rely on the cloud server -- which keeps the actual dataset -- to compute $H(m_i)$ for verification of data integrity. As the only way for the client to verify the correctness of $H(m_i)$ is through ADS, the server can cheat the client with another hash and AAI pair. In other words, the server can take any other healthy block to replace the block that should be verified, which denies the primary aim of integrity verification. To the best of our knowledge, there is no existing public auditing scheme that supports full dynamic data which can deal with this problem.

Erway et, al.'s RASL (Erway et al., 2009) (see Fig. 3-1 for an example) can provide authentication for indices, which is resilient to the above attacks. Aside from the effective ADS, they propose a scheme where the authenticator/tag T is computed as $T = g^m$ where g is a generator and m is the message to be audited, however this is too simple to support public auditing. Without a hash value, they can be over-easily integrated or separated. In fact, the RASL cannot be directly applied into a public auditing scheme supporting dynamic data. As stated earlier, $H(m_i)$ -- the hash value of message block m_i -- is to be used in authenticators for support of dynamic data. Therefore, the client needs $H(m_i)$ computed by (and later transferred from) the cloud server for verification. In order to achieve verifiability of index information, the leaf nodes no longer store the hash value of file blocks, but the hash value of a concatenation of multiple values in the form of $f(v) = H(l(v), r(v), x(v), f(\text{rgt}\,(v)))$. Therefore, the server needs to send back both values of $f(v)$ and $H(m)$, and the client will need to verify $H(m)$. In an RASL, a common case is that multiple leaf nodes are in the same verification path, such as n_3, n_4, n_5 in Fig. 3-1. Let's say n_3, n_4, n_5 represents message blocks m_3, m_4, m_5. As stated earlier, the client needs $H(m_i)$ computed by and transferred from the cloud server for verification. In this case, if verification of m_3 is needed, the server not only

needs to return all 3 values on n_3, n_4, n_5 as part of AAI, but also needs to compute and transfer all $H(m_3), H(m_4)$ and $H(m_5)$. As there is only a small fraction of blocks (460 for 99% confidence when auditing a 1GB file), it is not likely that these consecutive blocks will be chosen for one audit, which means there will be excessive overheads. Also，the bottom-up levelling restricts the insertions. If leaf nodes are level 0 as defined in (Erway et al., 2009), any insertion that creates a new level below level 0 will cause an update of all level values (therefore all hash values of all nodes), which is hardly possible for the client to verify. For these reasons, in our MuR-DPA scheme introduced in Chapter 6, we choose to use MHT with top-down levelling instead of RASL to design the new ADS. Now that the leaf nodes are on different levels, we need both the client and verifier to remember the total number of blocks and verify the block index from both directions (leftmost to rightmost, rightmost to leftmost) to make sure the server does not cheat the client with another node on the verification path.

Chapter 4

Authenticated Key Exchange Schemes in Cloud

As analysed in Chapters 2 and 3, an authenticated key exchange (AKE) scheme in the background of the cloud is essential for secure proof integration where symmetric encryption is involved. Its efficiency will greatly impact the overall efficiency for public auditing as well as the other security-aware mechanisms of cloud. In Chapter 4, I will demonstrate two AKE schemes which have been designed for efficient and secure cloud auditing.

This chapter is organised as follows. Section 4.1 presents a key exchange scheme with the randomness-reuse strategy; the work is published in (Liu et al., 2011, Liu et al., 2013c). Section 4.2 presents a more efficient hierarchical key exchange scheme; the work is published in (Liu et al., 2014a, Liu et al., 2013b). Section 4.3 presents security and efficiency analyses for both of the schemes presented in this chapter.

4.1 CCBKE: Cloud Computing Background Key Exchange

4.1.1 System setup

The system chooses a large prime integer p to construct a Diffie-Hellman group, and a generator g of group $\mathbb{Z}_p^*$, i.e., g is a primitive root modulo p. Normally p is a Sophie Germain prime where $(p-1)/2$ is also prime, so that the group Z_p^*

maximises its resilience against square root attack to the discrete logarithm problem. A certificate authority (CA) as in PKI is still needed in our security framework so that communicating parties can identify each other through exchanging verifiable certificates $Cert_c$ and $Cert_{s_i}$, as the certificates contain public keys which can be used to verify the session partners' signatures, and thereby their identities. Certificates are relatively long-term data which are issued to all participants of communication before the commencement of communication, and CA won't participate itself unless re-verification of identities and revocation and re-issuing certificates for participants are needed. As these should be done in a much lower frequency (e.g. once a day) than key exchanging (e.g. re-exchanging the key in every new session), they won't affect the efficiency of a key exchange scheme for scheduling in general. Therefore, we will ignore all communications involving CA in our scheme and won't be discussing further details on issuing and revoking certificates.

4.1.2 Key Exchange

Initial exchange is used when a new task is to be executed, because that is when CLC needs to decide how to distribute this new task to be executed on existing computation infrastructure, i.e., which of the server instances are involved. CLC picks a secret value $x < p$, computes its public keying material g^x in $\mathbb{Z}_p^*$, and broadcasts the following message to the domain of server instances S which contains n instances $S_1, \dots, S_n$:

Round 1, *C -> S*: HDR_c , SA_{c_1} , g^x, $nonce_c$

where HDR and SA are for algorithm negotiation, g^x is for Diffie-Hellman key exchange, and $nonce$ is for freshness verification. The initiator of a normal IKE scheme will generate n secret values $x_1, x_2, \dots, x_n$, then compute and send out $g^{x_1}, g^{x_2}, \dots, g^{x_n}$, either through multicast or one by one, to establish separated security channels with each receiver. In our scheme, although we still establish one SA for each server instance S_i where $i = 1,2, \dots, n$, we use only one single secret

value x for CLC in all n messages in order to reduce cost. We further analyse security and cost reduction for this variation in section 4 and 5, respectively.

Upon receiving Message 1, each server instance generates their secret value $y_i < p$, compute key material g^{y_i}, then responds within Round 2 as follows:

Round 2, *S -> C*: HDR_{S_i} , $SA_{S_{i_1}}$, g^{y_i} , $nonce_{S_i}$, $CertReq_c$, for $i = 1, \dots, n$

Note that round 2 involves n different messages sent from $S_1, \dots, S_n$ separately. After exchanging the first two rounds of messages, the session keys $g^{xy_1}, \dots, g^{xy_n}$ are computed for all parties as follows:

$$C: g^{xy_1} = (g^{y_1})^x, \dots, g^{xy_n} = (g^{y_n})^x$$

$$S_1: g^{xy_1} = (g^x)^{y_1}$$

$$\dots$$

$$S_n: g^{xy_n} = (g^x)^{y_n}$$

The session keys are now shared between CLC and each server instance for the use of encryption of later communications. Although the Diffie-Hellman key exchange is completed, the CCBKE initial exchange is not finished as the participants have to authenticate each other in order to prevent man-in-the-middle (MITM) attacks. Similar to IKE, CLC generates signatures Sig_{c_i} which are the signatures for these n messages, and use its secret key from the key pair issued by CA:

$M_{c_i} = \text{prf}\big(\text{prf}\big(N_c||N_{S_i}||g^{xy_i}\big)||g^x||g^{y_i}||SA_c||ID_c\big)$, for $i = 1, \dots, n$
and broadcasts the following message to S:

Round 3, *C -> S*:

HDR_c ,
$\{ID_c, SA_{c_2}, Cert_c, CertReq_{s_1}, \text{Sig}_c\}_{g^{xy_1}} || \{ID_c, SA_{c_2}, Cert_c, CertReq_{s_2}, \text{Sig}_c\}_{g^{xy_2}}$
$|| \dots || \{ID_c, SA_{c_2}, Cert_c, CertReq_{s_n}, \text{Sig}_c\}_{g^{xy_n}}$, for $i = 1, \dots, n$

The server instances can then verify the identity of the initiator of this conversation by using its session key g^{xy_i} to decrypt its own part of this message. Signatures can be verified through the public key contained in the certificate. Similarly, server instances will send out their own encrypted ID, signature and certificate to CLC for verification:

Round 4, *S -> C*: HDR_{S_i} , $\left\{ID_c, SA_{s_{i_2}}, Cert_{s_i}, \text{Sig}_{s_i}\right\}_{g^{xy_i}}$, for $i = 1, \dots, n$

where, similar to round 3 but only signed separately, Sig_{s_i} is signatures by S_i to messages:

$M_{s_i} = \text{prf}\left(\text{prf}\left(N_c || N_{S_i} || g^{xy_i}\right) || g^{y_i} || g^x || \left|SA_{s_i}\right| || ID_{s_i}\right)$, for $i = 1, \dots, n$

Note that this round involves n messages as well. After the identities of both CLC and server instances are authenticated through round 3 and 4, CLC will send to $S_1, \dots, S_n$ the split task data which are encrypted with session keys $g^{xy_1}, \dots, g^{xy_n}$ using symmetric encryption such as AES. After task execution, $S_1, \dots, S_n$ returns to CLC the results which are encrypted using $g^{xy_1}, \dots, g^{xy_n}$ as well. The prf function is often implemented as an HMAC function such as SHA-1 or MD5, which outputs a fixed-length short message (commonly 128 bits) and has high efficiency (around 200MB/s on today's desktop PCs) itself.

4.1.3 Rekeying

Rekeying is often accomplished by running initial exchange all over again. However, in the following cases, alternative strategies need to be applied. We'll also analyse in this section the efficiency of these strategies.

a) Failure Recovery:

If any message that constitutes the initial exchange fails to arrive, the CLC will simply start a one-on-one IKE key exchange session with this specific instance. As this is only an accidental situation and can be tackled on-the-run, this additional time consumption can be considered negligible.

b) Multi-step Tasks

In a multi-step task, data need to be transferred back and forth. In this situation it is not necessary for the participants to re-authenticate each other after the successful authentication in the first round because of the high dependency of data in a similar task. Therefore, only rounds 1 and 2 are needed to be performed, with new keying materials and minor changes to the *SA* and *HDR* fields. Following the analyses in Section 3, as rounds 3 and 4 only contain fast operations such as signature and verification over short messages as well as symmetric-key encryption/decryption and HMAC functions, the computational overhead of the rekeying process on the CLC is almost identical to the initial exchange from an efficiency point of view.

4.2 HKE-BC: Hierarchical Key Exchange for Big data in Cloud

An overview of the HKE-BC scheme is shown in Figs. 4-1 and 4-2. Generally speaking, the scheme can be described as a layer-by-layer structure just as its name indicates. In the first phase, every control node will exchange a temporary key KEY_UP with its parent node and KEY_DOWN with its child node, and then undertake mutual authentications. In the second phase, CLC will send the final session keying encrypted with the temporary keys established in the first phase. Through these operations, the expensive exponentiation operations can be securely distributed to the intermediate control nodes.

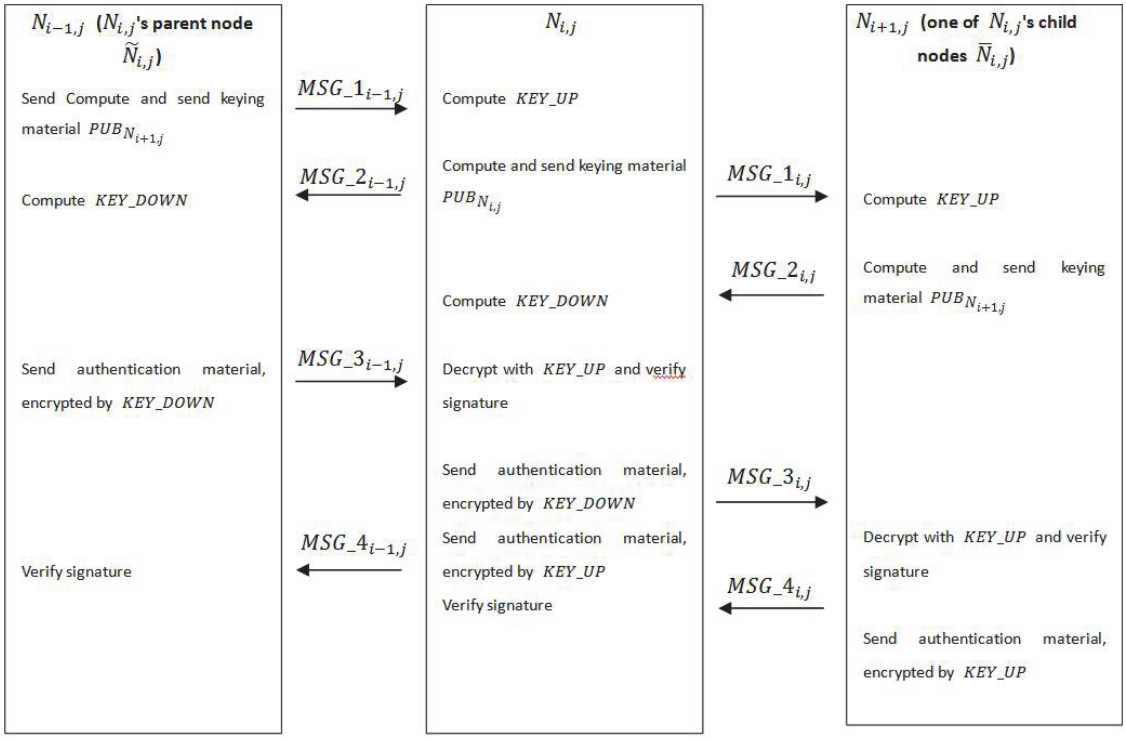


Figure 4-1 Process of HKE-BC Phase1.

4.2.1 System Setup

The system chooses a large prime integer p and selects a generator g of group $\mathbb{Z}_p^*$. Normally p is a Sophie Germain prime where $(p-1)/2$ is also prime, so that the group Z_p^* has maximum resilience against square root attack.

4.2.2 Key Exchange

This is a generalised description for a cloud infrastructure that has l control layers, from CLC to end NC. Layer i has n_i nodes, namely $N_{i,j}$, $i = 1, \dots, l$, $j = 1, \dots, n_i$. CLC is on layer 1, where $n_1 = 1$. Let $m_{i,j}$ be the numbers of sub-nodes for nodes $j = 1, \dots, n_i$ on layer i.

Overview: The scheme can be divided into two phases. Phase 1 is KE between control nodes, which aims at secure delivery of CLC's secret keying material to NCs;

while Phase 2 is for the actual key exchange between NC and the instances. NCs interact with virtualised instances on CLC's behalf, get responses from them, and then send back the results to CLC to deliver back the instances' keying materials to finalise the key exchange procedure. Brief graphs indicating the processes of both phases are provided in Figs. 4-1 and 4-2.

Phase 1: This phase is for KE between all control nodes from CLC (layer L_1) to the lth control layer (layer L_l, i.e., NC layer). This exchanged session key will be used for encrypting the real keying material in Phase 2.

All control nodes pick their own private key $x_{i,j}$ and one-time nonce $nonce_{i,j}$. They compute their public key for KE as follows:

$$PUB_{N_{i,j}} = g^{x_{i,j}}$$

Then CLC broadcasts the very first message $MSG\_1_{1,1}$:

$$MSG\_1_{1,1}: REQ\ , HDR_{1,1}\ , SA\ , PUB_{N_{1,1}}\ , nonce_{1,1}$$

to all nodes in layer 2. REQ is a flag for message identification, indicating the request for keying material.

For the nodes $N_{i,j}$ $(j = 1, \dots, n_i)$ in layer L_i, upon receiving message $MSG\_1_{(i-1),j}$ from their parent-node in L_{i-1} $(i = 2, \dots, l)$, they send messages $MSG\_1_{i,j}$ to their sub-nodes in the next layer L_{i+1}:

$$MSG\_1_{i,j}: REQ\ , HDR_{i,j}\ , SA\ , PUB_{N_{i,j}}, nonce_{i,j}$$

Meanwhile, they respond MSG_2 to their parent node:

$$MSG\_2_{i,j}: REQ\ , HDR_{i,j}\ , SA\ , PUB_{N_{i,j}}\ , CertReq\ , nonce'_{i,j}$$

After receiving MSG_1, every node in layers $L_2, \dots, L_l$ will know its parent node $\widetilde{N}$'s public key $PUB_{\widetilde{N}}$, and compute the session key for communicating with its parent:

$$KEY\_UP_{N_{i,j}} = (PUB_{\widetilde{N}})^{x_{i,j}}$$

where $i = 2, \dots, l; j = 1, \dots, n_i; k = 1, \dots, m_{i,j}$

For nodes in layer L_{i-1}, upon receiving MSG_2 from their sub-nodes, they'll know the public keys of their sub-nodes, namely $g^{x_{i+1,j}}$. We denote the public key of node $N_{i,j}$'s sub-nodes $\bar{N}_k$ as $PUB_{\bar{N}_k}, k = 1, \dots, m_{i,j}$. $N_{i,j}$ compute the following session keys for communicating with their sub-nodes:

$$KEY\_DOWN_{N_{i,j},k} = \left(PUB_{\bar{N}_k}\right)^{x_{i,j}},$$

where $i = 1, \dots, l-1; j = 1, \dots, n_i; k = 1, \dots, m_{i,j}$

For authentication, all nodes in L_i $(i = 1, \dots, l-1)$ broadcast MSG_3 to its sub-nodes $\bar{N}_k$:

$MSG\_3_{i,j}$:

$$HDR_{i,j}\ , \left\{ID_{N_{i,j}}\ , SA, Cert_{N_{i,j}}, CertReq_{\bar{N}_1}, \mathrm{Sig}_{N_{i,j}}\right\}_{KEY\_DOWN_{N_{i,j},1}} || \dots$$

$$||\left\{ID_{N_{i,j}}, SA, Cert_{N_{i,j}}, CertReq_{\bar{N}_k}, \mathrm{Sig}_{N_{i,j}}\right\}_{KEY\_DOWN_{N_{i,j},k}}$$

$$(i = 1, \dots, l-1; j = 1, \dots, n_i; k = 1, \dots, m_{i,j})$$

where the structure of message for signatures is also an output of prf(), similar to IKE. All nodes on L_i, $(i = 2, \dots, l)$ will receive this message, and respond with $MSG\_4_{i,j}$ if signature verification is successful:

$MSG\_4_{i,j}$:

$$HDR_{i,j}, \left\{ID_{N_{i,j}}, SA, Cert_{N_{i,j}}, \mathrm{Sig}_{N_{i,j}}\right\}_{KEY\_UP_{N_{i,j}}} \quad (i = 2, \dots, l; j = 1, \dots, n_i)$$

The reason that only the receiver of MSG_3 and MSG_4 can decrypt them is that, for every parent-child node pair N_{PARENT} and N_{CHILD}, we already have:

$$KEY\_DOWN_{N_{\mathrm{PARENT}}} = KEY\_UP_{N_{\mathrm{CHILD}}}$$

which concludes phase 1.

Phase 2: This phase is for the eventual goal of our scheme – KE between CLC and virtualised instances. The outcome of Phase 1 will play a vital role here.

CLC picks its secret value X as its keying material for KE with those virtualised instances. CLC encrypts X with the session key negotiated in phase 1 and broadcasts the following message to the next layer:

$$MSG\_1_{1,1}: \left\{X||ID_{N_{1,1}}\right\}_{KEY\_DOWN_{N_{1,1},1}} || \dots ||\left\{X||ID_{N_{1,1}}\right\}_{KEY\_DOWN_{N_{1,1},m_{1,1}}}$$

Upon receiving message $MSG\_1_{(i-1),j}$ from their parent-nodes in L_{i-1} $(i = 2, \dots, l)$, the nodes in L_i broadcast similar $MSG\_1_{i,j}$ to their sub-nodes in L_{i+1}:

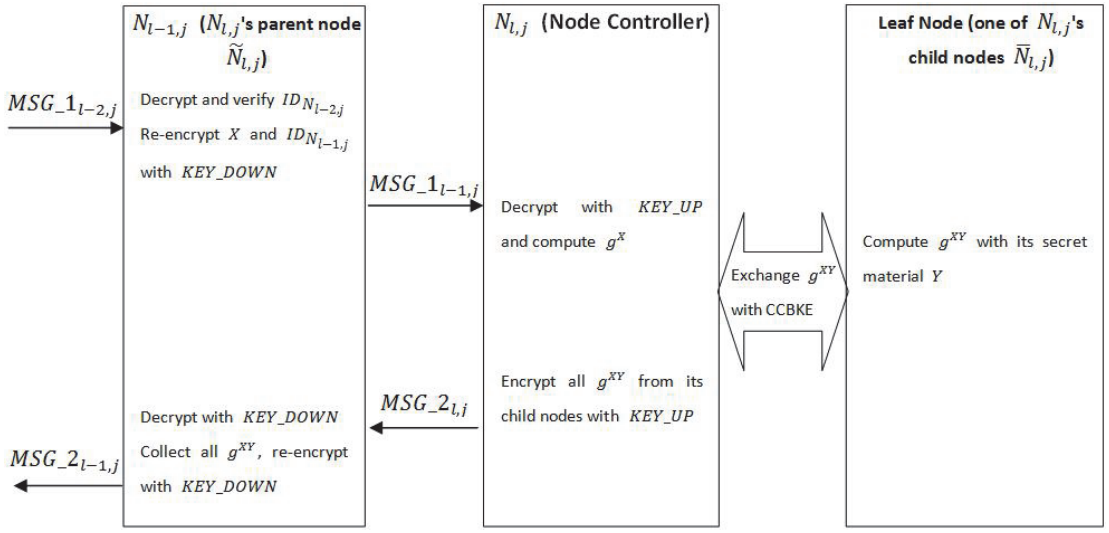


Figure 4-2 Process of HKE-BC Phase2.

$$MSG\_1_{i,j}: \left\{X||ID_{N_{i,j}}\right\}_{KEY\_DOWN_{N_{i,j},1}} || \dots || \left\{X||ID_{N_{i,j}}\right\}_{KEY\_DOWN_{N_{i,j},m_{i,j}}}$$

because the recipients can obtain X by decrypting the received MSG_1 using its $KEY_{UP_{N_{i,j},k}}$. For security reasons, all nodes in $L_2, \dots, L_{l-1}$ should destroy X after sending MSG_1 in Phase 2 where they re-encrypt X with KEY_{DOWN} and send to their sub-nodes.

After these operations, once nodes on layer L_l, i.e., NCs, get to know the X value. They now use this secret value to perform a 4-round CCBKE to finish the final KE:

NC-VM: HDR_{NC} , SA_{NC} , g^X, $nonce_c$

VM-NC: HDR_{VM_i} , SA_{VM_i} , g^{Y_i} , $nonce_{S_i}$, $CertReq$

NC-VM: HDR_{NC} ,

$\{ID_{NC}, SA_{NC}, Cert_{NC}, CertReq, \text{Sig}_{NC}\}_{g^{XY_1}}$

$|| \ldots ||\{ID_{NC}, SA_{NC}, Cert_{NC}, CertReq, \text{Sig}_{NC}\}_{g^{XY_\alpha}}$

VM-NC: HDR_{VM_i} , $\{ID_{VM_i}, SA_{VM_i}, Cert_{VM_i}, \text{Sig}_{VM_i}\}_{g^{XY_i}}$

The final session key for data encryption is g^{XY_i} where $i = 1, \ldots, \alpha$. After this step, not only α but all $m_{i,j}$ virtualised instances will have the desired session key for data encryption/decryption.

Now all virtualised instances have exchanged a key with their control nodes. For each NC, i.e. $N_{l,j}(j = 1, \ldots, n_l)$, they combine and encrypt the final session keys in this format:

$$MSG\_2_{l,j}: \left\{g^{XY_1} || \ldots || g^{XY_{m_{l,j}}}\right\}_{KEY\_UP_{N_{l,j}}}$$

and send it to its upper level. Then, nodes in every level from L_i, $i = 2, \ldots, i-1$ compute and send the following message to their parent nodes, after receiving from their sub-nodes:

$MSG\_2_{i,j}$:

$$\left\{DEC_{KEY_{DOWN_{N_{i,j,1}}}}\left(MSG_{2_{N_{i+1,1}}}\right) || \ldots || DEC_{KEY\_DOWN_{N_{i,j,m_{i,j}}}}\left(MSG\_2_{N_{i+1,m_{i,j}}}\right)\right\}_{KEY\_UP_{N_{i,j}}}$$

After this layer-by-layer operations, CLC, i.e., $N_{1,1}$, will know the session keys g^{XY_α} that have been negotiated with all virtualised instances, thereby concluding the KE scheme. The task data stored at CLC can now be split, encrypted and distributed to the virtualised instances for execution. After the execution, the server instances may follow an inverse procedure to exchange session keys with CLC and send back the encrypted results, or will keep using keys that have been changed in this procedure and re-exchanging keys in the next server-client interaction.

4.3 Security and Efficiency Analysis

The security of our schemes is analysed in Dolev-Yao's threat model with a bit extension. As we are dealing with communication security only, all the data stored on CLC and intermediate control nodes is assumed to be safe against the adversary in this model. We will analyse the security of our schemes in two ways, in that we will show that our scheme is safe against both outside and inside attackers while maintaining perfect forward secrecy. The abilities of the adversaries, or attackers, are defined as follows.

4.3.1 Security Proofs

As cryptanalysis on symmetric-key encryption algorithms is outside the scope of this thesis, the following discussions are under 2 standard cryptographic assumptions as follows.

Assumption 4.1: Any participant in our scheme cannot retrieve any data that was encrypted by any symmetric-key algorithm, unless it has the session key which was used to encrypt the data in the first place. i.e., cryptanalysis is beyond this security discussion.

Assumption 4.2 *(CDH assumption):* Given a cyclic group G of order$q$, a generator g of G and two random integers $a, b \in \{0, \dots, (q-1)\}$, retrieving g^{ab} in polynomial time using only $\{g, g^a, g^b\}$ is computationally impossible.

Similar to most security analyses of public-key communication protocols, we now define the capabilities of an outsider attacker and an inside attacker.

Definition 4.1 (*cloud outside attacker*): A malicious cloud outside attacker M_o aims to retrieve the session keys in exchange. An outside attacker M_o is an adversary who is capable of monitoring, intercepting, and changing all communication traffic in the whole cloud background structure, in order to gain access to protected data in

transit. However, M_o does not have access to any of the node machines, and its identity is not legitimate, i.e. M_o cannot obtain a valid certificate to let itself be authenticated by CLC or any server instance.

Definition 4.2 (*cloud inside attacker*): A malicious inside attacker M_i aims to steal the data of other users of the same cloud. M_i is an adversary who not only has the same ability as M_o, but also can be authenticated by the cloud and act as a legitimate server instance of the communication. However, M_i does not have access to any other legitimate participants' private information, including server instances' private information and CLC's private information.

We now prove that our scheme is secure against both these types of attackers.

Theorem 4.1: A *cloud outside attacker* M_o cannot retrieve in polynomial time any exchanged session key g^{xy_i} in CCBKE.

Proof: Following *Definition 1*, we know that an outside attacker M_o can gain access to all public keying materials $g, p, g^x, g^{y_1}, \dots, g^{y_n}$ by monitoring the entire network, but M_o cannot get secret information such as $x, y_1, \dots, y_n$. Considering the computational hardness of computational Diffie-Hellman (CDH) problem, we know that M_o cannot compute any g^{xy_i} where $i = 1, \dots, n$, in polynomial time, with $g^x, g^{y_1}, \dots, g^{y_n}$. □

Theorem 4.2: Assume $k = g^{xy_m}$ is the session key negotiated between a *cloud inside attacker* M_i and CLC. M_i cannot retrieve in polynomial time any session key g^{xy_i} other than k, unless there is a negligible probability.

Proof: According to *Definition 2*, M_i has its own secret value y_t, k in addition to information controlled by M_o. Since all secret values $y_1, \dots, y_n$ are generated by individual server instances themselves instead of allocated, there is a probability that the same secret value is generated and used for key exchange by different server instances, and we call this a 'collision'. For example, when a collision between M_i

and a legitimate server instance S_l (with its secret value y_l) happens, we have $y_l = y_m$. Therefore M_i can easily retrieve data sent by S_l using its own key g^{xy_m}. Since $y_1, \ldots, y_n$ are randomly chosen over $\mathbb{Z}_p^*$, the probability ϵ for a collision occurrence will be:

$$\epsilon = \prod_{i=1}^{n-1}\left(1 - \frac{i}{p}\right) \approx \prod_{i=1}^{n-1} e^{-\frac{i}{p}} = e^{-\frac{n(n-1)}{2p}}$$

This is a similar situation to the famous birthday attack where ϵ depends on $n/\sqrt{p}$ as well. In our CCBKE scheme, *p* is commonly 1024-bit and *n* (number of server instances) is usually several thousand which can be considered negligible compared to 2^{512}. To this end, $n/\sqrt{p}$ is very close to 0. Thus, the probability for a collision is negligible, which means an inside attacker cannot retrieve any of the others' session keys except when there is a negligible possibility. Combining this conclusion with *Theorem 1*, we have finished proving *Theorem 2*. □

Since all identity information are encrypted with the session keys in authentication rounds 3 and 4 and both outside attackers and inside attackers cannot retrieve the session key of others, we have the following lemma.

Lemma 4.1 Any pretended participant will fail authentication in rounds 3 and 4.

Once the authentication of rounds 3 or 4 fail, the communication will be terminated. Hence, a man-in-the-middle attack or any kind of identity forgery attack to our scheme will not be successful.

Derived from these theorems, we now have the following lemma regarding the security of the HKE-BC scheme:

Lemma 4.2: The adversaries defined above have a negligible chance of breaking the HKE-BC scheme. Specifically, a cloud outside attacker M_o cannot retrieve any session key, while a cloud inside attacker M_i cannot retrieve any session key other than her/his own.

Proof: The key exchange procedures for each node and its sub-nodes in both the HKE-BC Phase 1 and Phase 2 are actually minimised and iterative CCBKE processes. As CCBKE is secure against cloud inside and outside attackers according to Theorems 1 and 2, all the KE operations in HKE-BC scheme are secure against these attackers. Therefore, all the encrypted messages in our HKE-BC scheme are securely encrypted. Hence, we can say that our new HKE-BC scheme is secure against attackers from either outside or inside the cloud, as defined in *Definition 1*. □

In addition, if we use different parameters and keying materials for every execution and re-keying in the HKE-BC scheme, it will also hold *perfect forward security* just the same as in CCBKE and IKE.

4.3.2 Perfect Forward Secrecy

Similar to IKE, session keys used for encrypting communications are only used once until they are expired and destroyed. Thus, a previously used session key or secret keying material is worthless to a malicious opponent even if a previously-used key or a secret keying material is somehow exposed. This is one of the major advantages of using a key exchange scheme in hybrid encryption, which is why we did not choose to simply encrypt the session key with an asymmetric-key encryption algorithm, even though it can be easily done through PKI considering the fact that we

have adopted a CA in our CCBKE and architecture.

4.3.3 Efficiency Analysis for HKE-BC

We now analyse the efficiency improvements in the HKE-BC scheme when compared to its predecessors. As analysed in section II, the majority of time consumption is from modular exponentiations, e.g. $g^x \mathrm{mod} p$. Compared to them, the symmetric-key encryptions and decryptions in phase 2 take virtually no time because those concatenated keying materials to be encrypted are only several kilobytes long. Hence, we will analyse the efficiency advantage of our scheme by calculating the total number of modular exponentiations.

Let

$$M_i = \max_{1 \le j \le n_i} \{m_{i,j}\}$$

be the maximum number of sub-nodes for each node on level i. Starting from $M_1 = n_2$, we have

$$n_i = \sum_{j=1}^{n_{i-1}} m_{i-1,j} \le n_{i-1} M_{i-1}, i = 2, \dots, l$$

then the total number of VM instances is $\sum_{j=1}^{n_l} m_{l,j}$, with at most M_l VMs controlled by one NC. Assume the maximum time consumption of one modular exponentiation on one node is t_{max}, then the total time consumption of CCBKE is close to $\left(\sum_{j=1}^{n_l} m_{l,j}\right) t_{max}$ given that VM holds similar computational ability. In HKE-BC, the upper bound of the total time consumption in KE modular exponentiations in one round should be $\left(\sum_{i=1}^{l} M_i\right) t_{max}$. Given the fact that each NC can launch and control a large number of VMs (much more than the number of control nodes controlled by a higher-level control node), the following inequality will hold:

$$M_l > \sum_{i=1}^{l-1} M_i$$

Besides, because we have $l \geq 2$ (otherwise HKE-BC will have the exact same efficiency as CCBKE), we will have

$$\sum_{j=1}^{n_l} m_{l,j} \geq 2M_l$$

if the NCs have similar computational capability that can launch a similar amount of VMs. Therefore:

$$\sum_{j=1}^{n_l} m_{l,j} \geq 2M_l > M_l + \left(\sum_{i=1}^{l-1} M_i\right) = \sum_{i=1}^{l} M_i$$

then we have

$$\left(\sum_{j=1}^{n_l} m_{l,j}\right) t_{max} > \left(\sum_{i=1}^{l} M_i\right) t_{max}$$

which means in practical cloud settings, HKE-BC always has increased efficiency compared to CCBKE. In fact, in most cases we have:

$$M_l >> \sum_{i=1}^{l-1} M_i$$

then

$$\sum_{j=1}^{n_l} m_{l,j} \approx n_l \sum_{i=1}^{l} M_i$$

In this case, the time consumption of HKE-BC is even only a fraction of CCBKE. Although IKE, HKE-BC and CCBKE are all of linear time complexity to the scale of the task, the efficiency advantage of HKE-BC is nonetheless tremendous.

A detailed quantitative analysis with experimental results for the proposed key

exchange schemes is provided in Section 7.3.

Chapter 5

FU-DPA: Public Auditing for Dynamic Data with Fine-grained Updates

This chapter presents our research published in (Liu et al., 2014b) - the FU-DPA scheme for public auditing of dynamic cloud data storage. The chapter is organised as follows. Section 5.1 provides an introduction and states the main research contributions of this work. Section 5.2 presents a necessary preliminary for presenting our scheme -- the weighted Merkle hash tree. Section 5.3 provides our framework and definitions for the fine-grained updates supported in our scheme. Section 5.4 provides a detailed description of our proposed FU-DPA scheme. Section 5.5 provides a security and efficiency analysis.

5.1 Introduction

As analysed in Chapters 2 and 3, existing research work already allows data integrity to be verified without possession of the actual data file. As stated in Section 3, when the verification is done by a trusted third party, this verification process is also called data auditing, and the third party is called an auditor. However, such schemes in existence suffer from several common drawbacks. First, a necessary authorisation/authentication process is missing between the auditor and the cloud service provider, i.e., anyone can challenge the cloud service provider for a proof of the integrity of a certain file, which potentially puts the quality of the so-called 'auditing-as-a-service' at risk; Second, although some of the recent work based on the BLS signature can already support fully dynamic data updates over fixed-size

data blocks, they only support updates with fixed-sized blocks as basic units, which we call coarse-grained updates. As a result, every small update will cause re-computation and updating of the authenticator for an entire file block, which in turn causes higher storage and communication overheads. In this chapter, we provide a formal analysis for possible types of fine-grained data updates and propose a scheme that can fully support authorized auditing and fine-grained update requests. Based on our scheme, we also propose an enhancement that can dramatically reduce communication overheads for verifying small updates. Theoretical analysis and experimental results demonstrate that this scheme can offer not only enhanced security and flexibility, but also significantly lower overheads for big data applications with a large number of frequent small updates, such as applications in social media and business transactions. The research contribution of our scheme can be summarised as follows:

1. For the first time, we formally analyse different types of fine-grained dynamic data update requests on variable-sized file blocks in a single dataset. To the best of our knowledge, we are the first to propose a public auditing scheme based on the BLS signature and Merkle hash tree (MHT) that can support fine-grained update requests. Compared to existing schemes, our scheme supports updates with a size that is not restricted by the size of the file blocks, thereby it offers extra flexibility and scalability compared to existing schemes.

2. For better security, our scheme incorporates an additional authorisation process with the aim of eliminating threats of unauthorized audit challenges from malicious or pretended third-party auditors, which we term ‘authorised auditing’.

3. We investigate how to improve the efficiency in terms of verifying frequent small updates which exist in many popular cloud and big data contexts such as social media. Accordingly, we propose a further enhancement for our scheme to make it more suitable for this situation than existing schemes. Compared to existing schemes, both theoretical analysis and experimental results demonstrate that our

modified scheme can significantly lower communication overheads.

5.2 Preliminaries

5.2.1 Bilinear Pairing

Bilinear pairing is a foundation stone for the FU-DPA scheme introduced in this chapter. It was already introduced in Chapter 3. Therefore, details are omitted here to avoid duplication. Please refer to Section 3.1.3 for a detailed introduction of bilinear pairing.

5.2.2 Weighted Merkle Hash Tree

The Merkle Hash Tree (MHT) (Merkle, 1987) has been intensively studied in the past. In this thesis we utilise an extended MHT with weight values. The new authenticated data structure is named WMHT. Similar to a binary tree, each node N has a maximum of 2 child nodes. In fact, according to the update algorithm, every non-leaf node constantly has 2 child nodes. Information contained in one node N in an WMHT T is represented as $\{\mathcal{H}, r_N\}$ where $\mathcal{H}$ is a hash value and r_N is the weight of this node. T is constructed as follows. For a leaf node LN based on a message m_i, we have $\mathcal{H} = h(m_i)$, $r_{LN} = s_i$; A parent node of $N_1 = \{\mathcal{H}_1, r_{N1}\}$ and $N_2 = \{\mathcal{H}_2, r_{N2}\}$ is constructed as $N_P = \{h(\mathcal{H}_1||\mathcal{H}_2), (r_{N1} + r_{N2})\}$ where $||$ is a concatenation operator. A leaf node m_i's AAI Ω_i is a set of hash values chosen from every one of its upper level so that the root value R can be computed through $\{m_i, \Omega_i\}$. For example, for the WMHT demonstrated in Fig. 5-1, m_1's AAI $\Omega_1 = \{h(m_2), h(e), h(d)\}$. According to the property of WMHT, we know that the number of hash values included in Ω_i equals the depth of m_i in T.

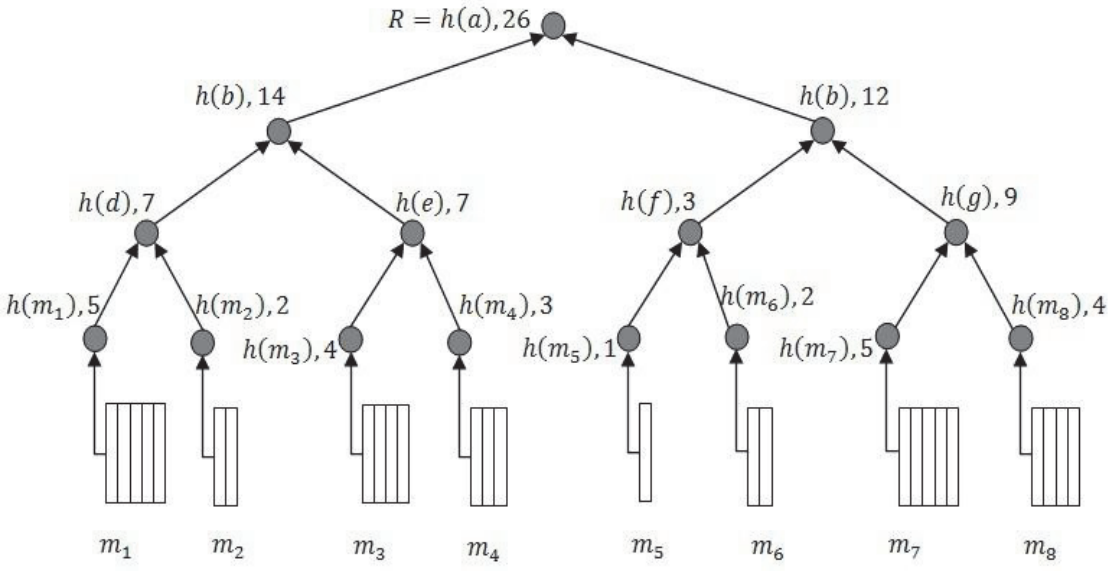


Figure 5-1 An example of a weighted Merkle hash tree (WMHT).

5.3 Framework and Definitions for Supporting Fine-grained Updates

We first define the following block-level fine-grained update operations:

Definition 1 (Types of Block-level Operations in Fine-grained Updates): Block-level operations in fine-grained dynamic data updates may contain the following 6 types of operations: partial modification $\mathcal{PM}$ -- a consecutive part of a certain block needs to be updated; whole-block modification $\mathcal{M}$ -- a whole block needs to be replaced by a new set of data; block deletion $\mathcal{D}$ -- a whole block needs to be deleted from the tree structure; block insertion $\mathcal{I}$ -- a whole block needs to be created on the tree structure to contain newly inserted data; and block splitting $\mathcal{SP}$ – a part of data in a block needs to be taken out to form a new block to be inserted next to it. [1]

[1] There are other possible operations such as block merging $\mathcal{ME}$ -- two blocks need to be merged

The framework of the public auditing scheme with data dynamics support consists of a series of algorithms. Similar to (Erway et al., 2009), the algorithms in our framework are: $KeyGen$, $FilePreProc$, $Challenge$, $Verify$, $GenProof$, $PerformUpdate$ and $VerifyUpdate$.

$KeyGen(1^k) \rightarrow \{sk, pk\}$: This algorithm is performed by the client for key generation. It outputs the secret key sk and public key pk based on a predefined security requirement.

$FilePreProc(F, sk, SegReq) \rightarrow \{\mathcal{F}, \Phi, T, R, sig, t\}$: This algorithm is performed by the client before uploading the file to CSS. It takes the file F, segmentation request $SegReq$ and the client's secret key as input, and outputs the segmented file $\mathcal{F}$, the set of homomorphic linear authenticators Φ, a WMHT T construction based on $\mathcal{F}$, the root hash R of T, R's signature sig, and a file tag t.

$PerformUpdate(UpdateReq, \mathcal{F}) \rightarrow \{\mathcal{F}', P_{update}\}$: This algorithm is performed by the CSS to perform an update request $UpdateReq$ from the client. An update request here means one operation in one single block in the format of $UpdateReq = \{Type, i, o, m_{new}\}$. $Type = \{\mathcal{PM}, \mathcal{M}, \mathcal{D}, \mathcal{I}, \mathcal{SP}\}$ indicates the type of operation as defined in *Definition 1*; i indicates the index of the block that needs to be verified (namely m_i); o is the starting offset of this update (only used when $Type = \mathcal{PM}, \mathcal{SP}$); and m_{new} is the new data that need to be added in $\mathcal{F}$ (only used in $Type = \mathcal{PM}, \mathcal{M}, \mathcal{I}$). We will show in Section 4.4 how these operations can compose fine-grained updates requests.

$VerifyUpdate(pk, P_{update}) \rightarrow \{TRUE, FALSE\}$: This is for the client to

---

into the first block before the second block is deleted, and data moving $\mathcal{MV}$ -- moves a part of data from one block to another, if the size of the second block does not exceed $s_{\max} \cdot \eta$ after this update. However, the fine-grained update requests discussed in this paper do not involve these operations, thus we will omit them in our current discussion. We will leave the problem of how to exploit them to future work.

verify dynamic data updating based on the proof P_{update} returned by CSS.

$GenChallenge(Acc,\ pk,\ sig_{AUTH}) \rightarrow \{chal\}$: The client or a third party authorised by it (e.g. a TPA) can use this algorithm to generate a challenge message $chal = \{sig_{AUTH}, VID, \{i, v_i\}_{i \in I}\}$ to the CSS to verify data integrity. Acc is an auditing accuracy parameter that is determined by the client, which will determine the subset I of $\mathcal{F}$ that needs to be verified this time.

$GenProof(pk, F, sig_{AUTH}, \Phi, chal) \rightarrow \{P, REJECT\}$: The CSS will use this algorithm to generate a proof P to respond to the verifier. The algorithm will return $REJECT$ if the verification of sig_{AUTH} with pk fails.

$Verify(pk, chal, P) \rightarrow \{TRUE, FALSE\}$: The verifier, either client or TPA, will verify the integrity proof P provided by CSS using this algorithm.

Based on this framework, we now present the main FU-DPA scheme.

5.4 The Proposed Scheme

5.4.1 First Scheme

We now describe our proposed scheme with the aim of supporting variable-sized data blocks, authorszed third-party auditing and fine-grained dynamic data updates.

Overview: Our scheme is described in three parts:

1) Setup: the client will generate keying materials via $KeyGen$ and $FilePreProc$, then upload the data to CSS. Unlike previous schemes, the client will store a WMHT instead of an MHT as metadata. Moreover, the client will authorise the TPA by sharing a value sig_{AUTH}.

2) Verifiable Data Updating: the CSS performs the client's fine-grained update

requests via $PerformUpdate$, then the client runs $VerifyUpdate$ to check whether CSS has performed the updates on both the data blocks and their corresponding authenticators (used for auditing) honestly.

3) Challenge, Proof Generation and Verification: Describes how the integrity of the data stored on CSS is verified by TPA via $GenChallenge$, $GenProof$ and $Verify$.

We now describe our scheme in detail as follows.

Setup: This phase is similar to the existing BLS-based schemes except for the segmentation of file blocks. Let $e: G \times G \to G_T$ be a bilinear map defined in Section 4.1, where G is a GDH group supported by $\mathbb{Z}_p$[2]. $H: (0,1)^* \to G$ is a collision-resistant hash function, and h is another cryptographic hash function.

After all parties have finished negotiating the fundamental parameters above, the client runs the following algorithms:

$KeyGen(1^k)$: The client generates a secret value $\alpha \in \mathbb{Z}_p$ and a generator g of G, then computes $v = g^\alpha$. A secret signing key pair $\{spk, ssk\}$ is chosen with respect to a designated provably secure signature scheme whose signing algorithm is denoted as Sig(). This algorithm outputs $\{ssk, \alpha\}$ as the secret key sk and $\{spk, v, g\}$ as the public key pk. For simplicity, in our settings, we use the same key pair for signatures, i.e., $ssk = \alpha$, $spk = \{v, g\}$.

$FilePreProc(F, sk, SegReq)$: According to the preemptively determined segmentation requirement $SegReq$ (including $s_{\max}$, a predefined upper-bound of the number of segments per block), segments file F into $\mathcal{F} = \{m_{ij}\}, i \in [1, l], j \in$

[2] Most exponential operations in this paper are modulo p. Therefore, from now on, for simplicity, we will use g^α instead of $g^\alpha \bmod p$ unless otherwise specified.

$[1, s_i], s_i \in [1, s_{\max}]$, i.e., F is segmented into a total of l blocks, with the ith block having s_i segments. In our settings, every file segment should be of the same size $\eta \in (0, p)$ and as large as possible (see (Shacham and Waters, 2008)). Since $|p| = 20$ bytes is used in a BLS signature with 80-bit security (sufficient in practice), $\eta = 20$ bytes is a common choice. According to $s_{\max}$, a set $U = \{u_k \in \mathbb{Z}_p\}, k \in [1, s_{\max}]$ is chosen so that the client can compute the HLAs σ_i for each block: $\sigma_i = \left(H(m_i)\prod_{j=1}^{s_i} u_j^{m_{ij}}\right)^{\alpha}$ which constitutes the ordered set $\Phi = \{\sigma_i\}_{i\in[1,l]}$. This is similar to signing a message with the BLS signature. The client also generates a root R based on the construction of a WMHT T over $H(m_i)$ and computes $sig = \left(H(R)\right)^{\alpha}$. Finally, let $u = \left(u_1 || \dots || u_{s_{\max}}\right)$, the client computes the file tag for F as $t = name||n||u||\text{Sig}_{ssk}(name||n||u)$ and then outputs $\{\mathcal{F}, \Phi, T, R, sig, t\}$.

Prepare for Authorisation: The client asks (her choice of) TPA for its ID VID (for security, VID is used for authorisation only). TPA will then return its ID, encrypted with the client's public key. The client will then compute $sig_{AUTH} = \text{Sig}_{ssk}(AUTH||t||VID)$ and sends sig_{AUTH} along with the auditing delegation request to TPA for it to compose a challenge later on.

Different from existing schemes, after the execution of the above two algorithms, the client will keep the WMHT 'skeleton' with only the weights of each node and indices of each file block in order to reduce fine-grained update requests to block-level operations. We will show how this can be done in Section 4.4. The client then sends $\{\mathcal{F}, t, \Phi, sig, AUTH\}$ to CSS and deletes $\{F, \mathcal{F}, t, \Phi, sig\}$ from its local storage. The CSS will construct a WMHT T based on m_i and keep T stored with $\{\mathcal{F}, t, \Phi, sig, AUTH\}$ for later verification, which should be identical to the tree spawned at the client-side shortly before.

Verifiable Data Updating: Same as *Setup*, this process will also be between the client and the CSS. We discuss 5 types of block-level updates (operations) that

will affect T: $\mathcal{PM}, \mathcal{M}, \mathcal{D}, \mathcal{I}$ and $\mathcal{SP}$ (see *Definition 1*). We will discuss how these requests can form fine-grained update requests in general in Section 4.4.

The verifiable data update process for a $\mathcal{PM}$-typed update is as follows (see Fig. 5-2):

1. The client composes an update quest $UpdateReq$ defined in Section 4.2 and sends it to CSS.

2. CSS executes the following algorithm:

$PerformUpdate(UpdateReq, \mathcal{F})$: CSS parses $UpdateReq$ and get $\{\mathcal{PM}, i, o, m_{new}\}$. When $Type = \mathcal{PM}$, CSS will update m_i and T accordingly, then output $P_{update} = \{m_i, \Omega_i, R', sig\}$ (note that Ω_i stays the same during the update) and the updated file $\mathcal{F}'$.

Upon finishing this algorithm, CSS will send P_{update} to the client.

3. After receiving P_{update}, the client executes the following algorithm:

$VerifyUpdate(pk, P_{update})$: The client computes m_i' using $\{m_i, UpdateReq\}$, then parses P_{update} to $\{m_i, \Omega_i, R', sig\}$, compute R (and $H(R)$) and R_{new} use $\{m_i, \Omega_i\}$ and $\{m_i', \Omega_i\}$ respectively. It verifies sig use $H(R)$, and checks if $R_{new} = R'$. If either of these two verifications fails, then the output is $FALSE$ and it returns to CSS, otherwise the output is $TRUE$.

If the output of the algorithms is $TRUE$, then the client computes $\sigma_{\mathrm{i}}' = \left(H(m_i') \prod_{j=1}^{s_i} u_j^{m_{ij}'}\right)^{\alpha}$ and $sig' = \left(H(R')\right)^{\alpha}$ then sends $\{\sigma_{\mathrm{i}}', sig'\}$ to CSS.

4. The CSS will update σ_i to σ_{i}' and sig to sig' accordingly and delete $\mathcal{F}$ if it receives $\{\sigma_{\mathrm{i}}', sig'\}$, or it will run $PerformUpdate()$ again if it receives *FALSE*. A cheating CSS will fail the verification and constantly receive $FALSE$ until it performs the update as the client requested.

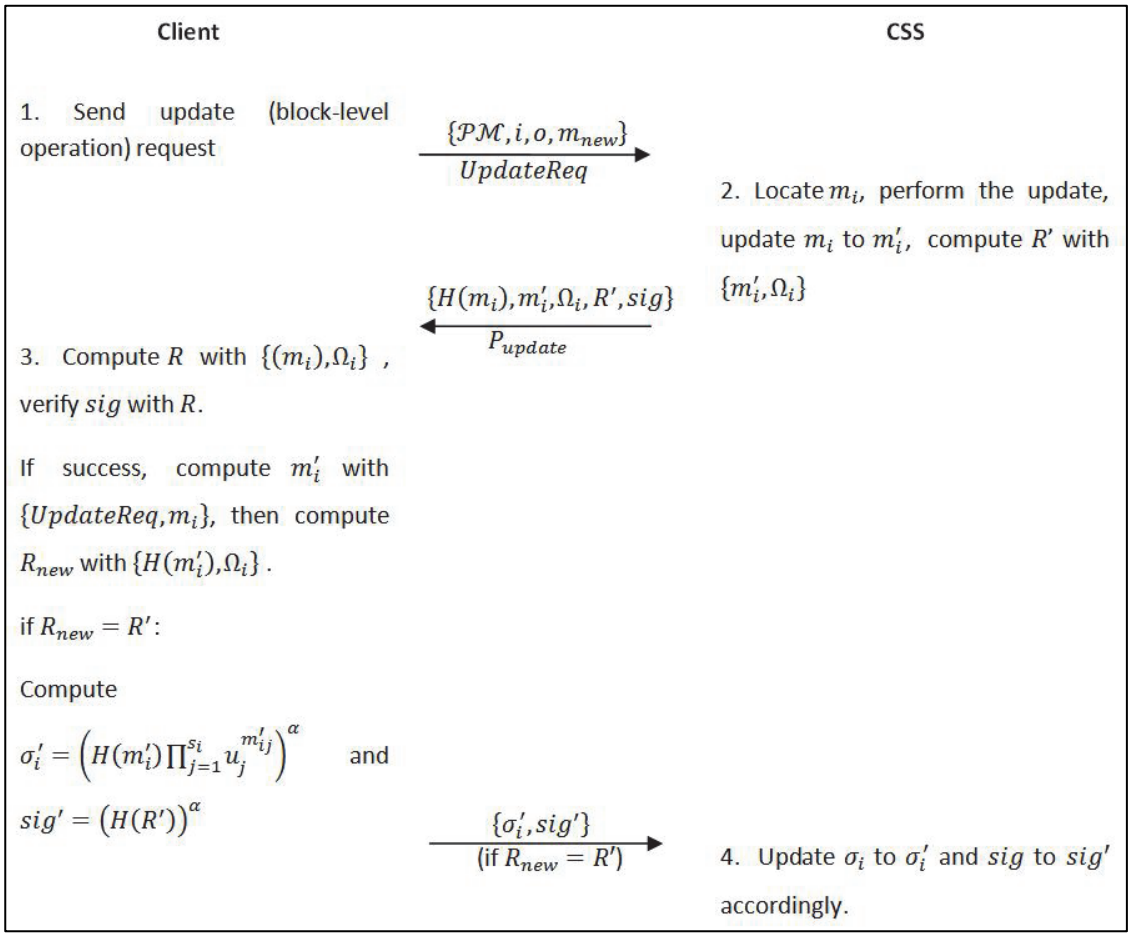


Figure 5-2 Verifiable *PM*-typed data update in FU-DPA.

Due to their similarity to the process described above, other types of operations are only briefly discussed as follows. For whole-block operations $\mathcal{M}, \mathcal{D}$ and $\mathcal{I}$, as in the model in the existing work (Wang et al., 2011b), the client can directly compute σ_i' without retrieving data from the original file F stored on CSS, thus the client can send σ_i' along with the $UpdateReq$ in the first phase. For responding to an update request, CSS only needs to send back $H(m_i)$ instead of m_i. Other operations will be similar to where $Type = \mathcal{PM}$. For a $\mathcal{SP}$-typed update, in addition to updating m_i to m_i', a new block m^* needs to be inserted to T after m_i'. Nonetheless, as the contents in m^* is a part of the old m_i, the CSS still needs to send m_i back to the client. The process afterwards will be similar to a $\mathcal{PM}$-typed upgrade, with the only exception that the client will compute R_{new}

using $\{m_i', h(m^*), \Omega_i\}$ to compare to R', instead of using $\{m_i', \Omega_i\}$ as in the $\mathcal{PM}$-typed update.

Challenge, Proof Generation and Verification: In our setting, TPA must show CSS that it is indeed authorised by the file owner before it can challenge a certain file.

1. TPA runs the following algorithm:

$GenChallenge(Acc,\ pk,\ sig_{AUTH})$: According to the accuracy required in this auditing, TPA will decide to verify c out of the total l blocks. Then, a challenge message $chal = \{sig_{AUTH}, \{VID\}_{PK_{CSS}}, \{i, v_i\}_{i\in I}\}$ is generated where VID is TPA's ID, I is a randomly selected subset of $[1, l]$ with c elements and $\{v_i \in \mathbb{Z}_p\}_{i\in I}$ are c randomly-chosen coefficients. Note that VID is encrypted with the CSS's public key PK_{CSS} so that CSS can later decrypt $\{VID\}_{PK_{CSS}}$ with the corresponding secret key.

TPA then sends $chal$ to CSS.

2. After receiving $chal$, CSS will run the following algorithm:

$GenProof(pk, F, sig_{AUTH}, \Phi, chal)$: Let $w = \max\{s_i\}_{i\in I}$. CSS will first verify sig_{AUTH} with $AUTH$, t, VID and the client's public key spk, and output $REJECT$ if it fails. Otherwise, CSS will compute $\mu_k = \sum_{i\in I} v_i m_{ik}, k \in [1, w]$ and $\sigma = \prod_{i\in I} \sigma_i^{v_i}$ and compose the proof P as $= \{\{\mu_k\}_{k\in[1,w]}, \{H(m_i), \Omega_i\}_{i\in I}, sig\}$, then output P. Note that during the computation of μ_k, we will let $m_{ik} = 0$ if $k > s_i$.

After execution of this algorithm, CSS will send P to TPA. 3. After receiving P, TPA will run the following algorithm:

$Verify(pk, chal, P)$: TPA will compute R using $\{H(m_i), \Omega_i\}$ and then verify sig using the public keys g and v by comparing $e(sig, g)$ with

$(H(R), v)$. If they are equal, let $\omega = \prod_{i \in I} H(m_i)^{v_i} \cdot \prod_{k \in [1,w]} u_k^{\mu_k}$, TPA will further check if $e(\sigma, g)$ equals $e(\omega, v)$, which is similar to verifying a BLS signature. If all the two equations hold then the algorithm returns $TRUE$, otherwise it returns $FALSE$.

An illustration of Challenge and Verification processes can be found in Fig. 5-3.

5.4.2 Analysis on Fine-grained Dynamic Data Updates

Following the settings in our proposed scheme, we now define a fine-grained update request for an outsourced file divided into l variable-sized blocks, where each block consists of $s_i \in [1, s_{\max}]$ segments of a fixed size η each. Assume a WMHT T is built upon $\{m_i\}_{i \in [1,l]}$ for authentication, which means T must keep updated with each WMHT operation in order for CSS to send back the root R for the client to verify the correctness of this operation (see Section 4.3). We now try to define and categorise all types of fine-grained updates, and then analyse the WMHT operations with $Type = \mathcal{PM}, \mathcal{M}, \mathcal{D}, \mathcal{I}$ or $\mathcal{SP}$ that will be invoked along with the update of the data file.

Definition 2 (Fine-grained Data Update Request): A fine-grained update request is defined as $FReq = \{o, len, m_{new}\}$, where o indicates the starting offset of this update in F, len indicates the data length after o that needs to be updated (so that $\{o, len\}$ can characterise an exact proportion of the original file F that needs to be updated, which we will later call m_{old}), and m_{new} is the new message to be inserted into F from offset o.

We assume the data needed to be obsolete and the new data to be added shares a common starting offset o in F, as otherwise it can be split into multiple updates defined in *Definition 2* commencing in sequence. We now introduce a rule to follow during all update processes:

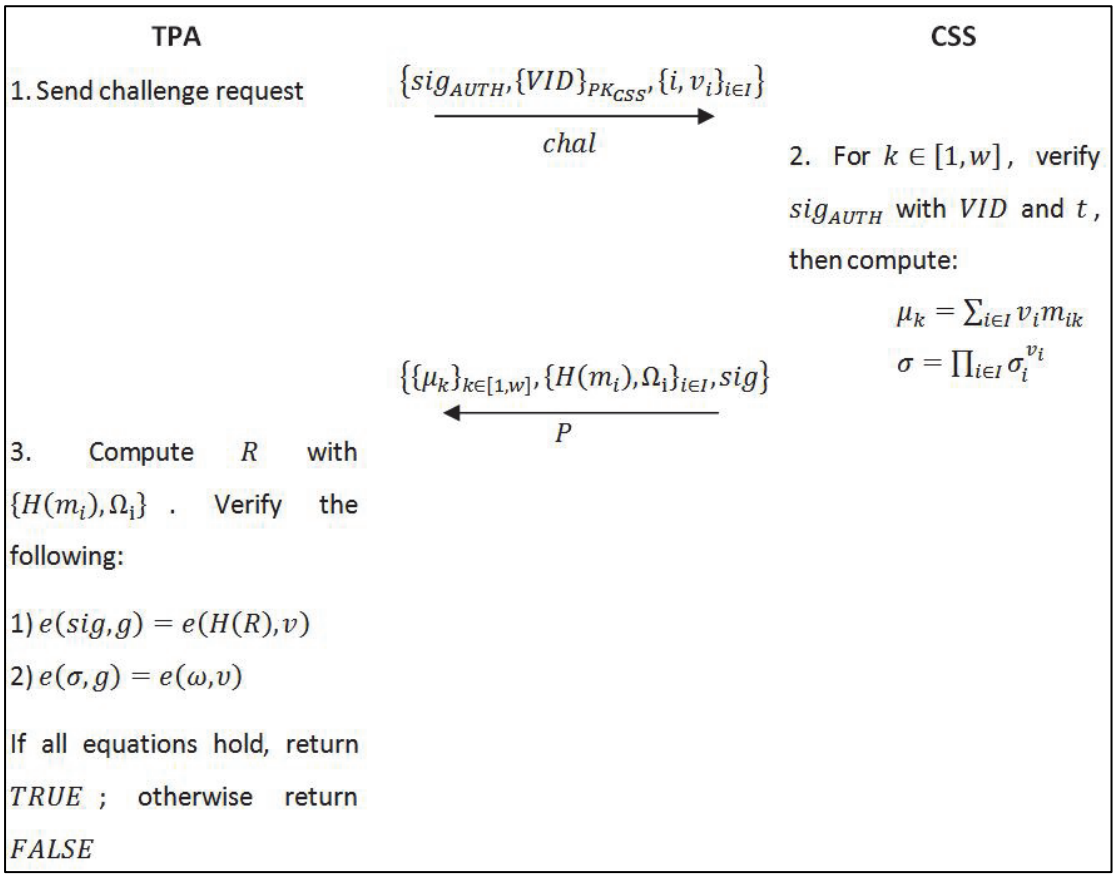


Figure 5-3 Challenge, proof generation and verification in FU-DPA.

Condition 1 (Block Size Limits in Updates): An update operation must not cause the size of any block to exceed $s_{\max}$; After any operation, a block that has 0 bit data remaining must be deleted from T.

For the convenience of a clearer discussion and implementation, we add an additional parameter indicating the type of update. A request is thereby formatted as $Req = \{TYPE, o, len, m_{new}\}$. $TYPE = \{Modify, Insert, Delete\}$ indicates whether this update is a *Modify*, *Insert* or *Delete*. We assume the offset o is located in block m_i, which can be efficiently located using o and T through algorithm findBlock() (see Fig. 5-4). That is to say, let ξ_{Req} be the length of data between o and the start offset of m_j, i.e., we assume $\xi_{Req} = (o - \sum_{j=1}^{i-1}|m_j|) \in [0, |m_i|]$. The situation where $\xi_{Req} = 0$ or $\xi_{Req} = |m_i|$ will be a little different from the majority

```
findBlock(int offset, RMHT T, segment_size η)
Node N = T → root;
while((N → left ≠ NULL) & (N → right ≠ NULL) )
    if (offset/η) < r_{N→left}
        N = N → left;
    endif
return N
```

Figure 5-4 The algorithm to find a block in *F* with a given offset *o*.

of cases where $\xi_{Req} \in (0, |m_i|)$. Note that when we have an average of several hundreds of segments per block, the probability for a random update request Req to satisfy $\xi_{Req} = 0$ or $\xi_{Req} = |m_i|$ is so slim that it can be neglected[3]. Nevertheless, for completeness, we will still include these cases in the discussion, but we will not consider these cases in the efficiency analysis. We now analyse the three types of fine-grained updates separately.

1) $TYPE = Insert$: In this case we have $len = |m_{old}| = 0$. As *Condition 1* must be complied, we will classify the update into several possible cases as follows and discuss them separately. We will use a variable $\rho_I = (|m_{new}| + |m_i|)$ for measurement.

(1-a) We first discuss the majority case where $\xi_{Req} \in (0, |m_i|)$.

(1-a-i) When $\rho_I \in (0, s_{\max} \cdot \eta]$, a direct insertion into m_i will not cause a breach of the upper bound $s_{\max} \cdot \eta$. Therefore, only one $\mathcal{PM}$ operation is needed.

(1-a-ii) When $\rho_I \in (s_{\max} \cdot \eta, 2s_{\max} \cdot \eta]$, a direct insertion will contradict with *condition 1*, therefore one split operation $\mathcal{SP}$ and 2 $\mathcal{PM}$ operations are required.

[3] The probability is $\frac{l+1}{|F|}$ when o is counted in bits. If o is counted in segments, this probability will become $\frac{l+1}{\sum_{i=1}^{l} s_i}$ which is still a tiny percentage when the average number of segments per block is high.

The split point in m_i depends on the length of m_{new}. As a result, m_i is split into 2 blocks m_i' and m^* with data in m_{new} inserted into m_i and the new block m^* separately. Note that after the update $|m_i'|$ will reach the upper bound $s_{\max} \cdot \eta$.

(1-a-iii) When ρ_I being larger, e.g., $\rho_I \in (2s_{\max} \cdot \eta, ns_{\max} \cdot \eta]$ for some $n > 2, n \in \mathbb{N}$, there will be a total of $n-1$ new blocks $\{m_1^*, \dots, m_{n-1}^*\}$ inserted between the old m_i and m_{i+1} to store the remaining of m_{new} after the operations in (1-a-ii). This will be a straightforward extension to where $\rho_I \in (s_{\max} \cdot \eta, 2s_{\max} \cdot \eta]$ -- there will be a total of 1 $\mathcal{SP}$ operation, 2 $\mathcal{PM}$ operations, and $(n-2)$ whole-block insertion ($\mathcal{I}$) operations.

We now discuss the case when $\xi_{Req} = 0$ and $\xi_{Req} = |m_i|$. For simplicity, we will only provide the results.

(1-b) When $\xi_{Req} = 0$, $\rho_I \in (0, s_{\max} \cdot \eta]$ there will be 1 $\mathcal{PM}$ operation, same as (1-a); for $\rho_I \in (s_{\max} \cdot \eta, 2s_{\max} \cdot \eta]$ there will be 1 $\mathcal{SP}$ operation, 1 $\mathcal{PM}$ operation and 1 $\mathcal{I}$ operation if $\rho_I \in \big(s_{\max} \cdot \eta, (s_{\max} \cdot \eta + |m_i|)\big)$, or 1 $\mathcal{PM}$ operation and 1 $\mathcal{I}$ operation if $\rho_I \in [(s_{\max} \cdot \eta + |m_i|), 2s_{\max} \cdot \eta]$; for $\rho_I \in (2s_{\max} \cdot \eta, ns_{\max} \cdot \eta]_{i>2,}$, there will be one $\mathcal{PM}$ operation and $(n-1)$ $\mathcal{I}$ operations.

(1-c) When $\xi_{Req} = |m_i|$, $\rho_I \in (0, s_{\max} \cdot \eta]$ there will be 1 $\mathcal{PM}$ operation, same as (1-a); for $\rho_I \in (s_{\max} \cdot \eta, ns_{\max} \cdot \eta]_{i>1}$ there will be one $\mathcal{PM}$ operation and n $\mathcal{I}$ operations.

An example of this type of update (1-a-ii, to be representative) is demonstrated in Fig. 5-5.

2) $TYPE = Delete$: In this case $m_{new} = 0$, and the measurement variable we use here is $\rho_D = \big(\xi_{Req} + len\big)$.

(2-a) As in 1), we will first discuss the case when $\xi_{Req} \in (0, |m_i|)$.

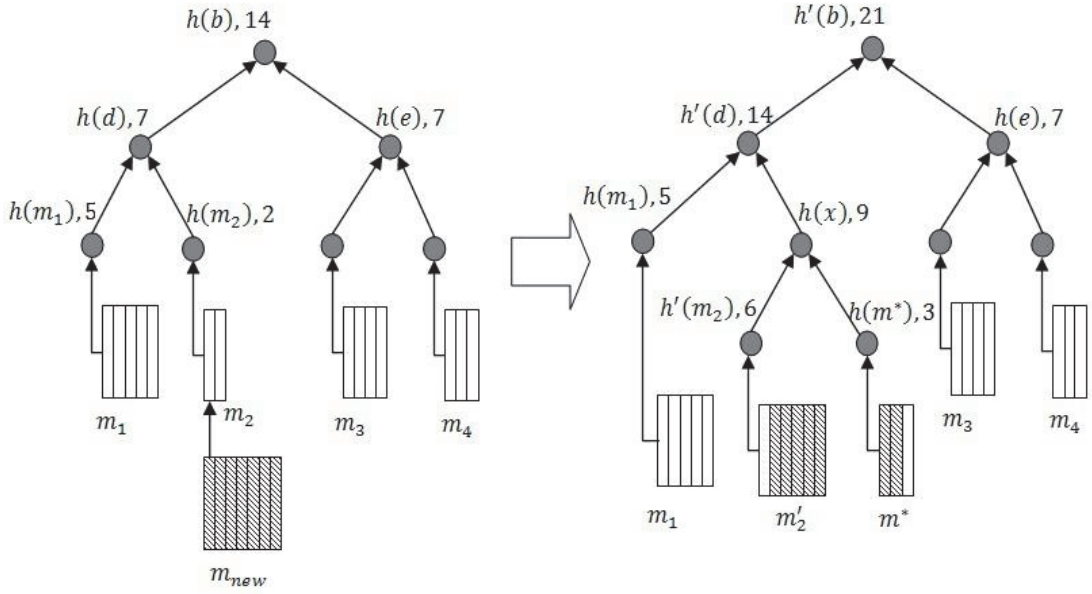


Figure 5-5 Example: fine-grained insertion.

An example of a (1-a-ii)-insert operation for $Req = \{INSERT, 120, 0, m_{new}\}$ where $\xi_{Req} = 20$, $|m_{new}| = 140$, $\rho_I = 180$ and $i = 2$, in a WMHT where $s_{max} = 6, \eta = 20$(bytes).

(2-a-i) When $\rho_D \in (0, |m_i|]$, similar to (1-a-i), there will be one $\mathcal{PM}$ operation on m_i.

(2-a-ii) When $\rho_D \in (|m_i|, (|m_i| + |m_{i+1}|))$, there will be 2 $\mathcal{PM}$ operations as both of the two blocks m_i and m_{i+1} have data remaining after deletion.

(2-a-iii) When $\rho_{\mathcal{D}} \in [(|m_i| + |m_{i+1}|), \sum_{j=i}^{i+2}|m_j|)$, on top of t he $\mathcal{PM}$ operation in m_i, all the contents in block m_{i+1} needs to be erased, thus one $\mathcal{D}$ operation of m_{i+1} is required. In addition, when $\rho_{\mathcal{D}} > (|m_i| + |m_{i+1}|)$, another $\mathcal{PM}$ operation is needed because a part of the next block m_{i+2} needs to be deleted as well. Similarly, we will generalise this case to that of when $\rho_{\mathcal{D}} \in \left(|m_i|, \sum_{j=i}^{i+n}|m_j|\right)$ & $\rho_{\mathcal{D}} \neq \sum_{j=i}^{i+n-1}|m_j|$ for some $n > 1, n \in \mathbb{N}$, there will be 2 $\mathcal{PM}$ operations and $(n-1)$ $\mathcal{D}$ operations in total; when $\rho_{\mathcal{D}} = \sum_{j=i}^{i+n-1}|m_j|$, there will be 1 $\mathcal{PM}$ operation and $(n-1)$ $\mathcal{D}$ operations in total.

(2-b) When $\xi_{Req} = 0$, there will be one more $\mathcal{D}$ operation and one less $\mathcal{PM}$ operation when $\rho_{\mathcal{D}} \geq |m_i|$, the remaining of the operations will stay the same.

(2-c) When $\xi_{Req} = |m_i|$, the discussion will be parallel to where $\xi_{Req} = 0$, which we will omit here.

An example for (2-a-iii)-deletion is shown in Fig.5-6.

3) $TYPE = Modify$: We will only discuss the case where $len = |m_{old}| = |m_{new}| \neq 0$ and $\rho_M = \left(o - \sum_{j=1}^{i-1}|m_j| + len\right) \in (0, |m_i|]$. If $|m_{old}| \neq |m_{new}|$, a $TYPE = Modify$ request can be easily split into a $TYPE = \mathcal{M}$ request where $|m_{old}| = |m_{new}|$ and a $TYPE = Insert$ or $Delete$ request. Therefore, we believe a discussion on the case of where $|m_{old}| = |m_{new}|$ will be sufficient. In this case, we choose $\rho_M = \rho_D = \left(o - \sum_{j=1}^{i-1}|m_j| + len\right) \in (0, |m_i|]$, then the classified discussion will be extremely similar to where $TYPE = Delete$, therefore we will only list the results as follows.

(3-a) When $\xi_{Req} \in (0, |m_i|)$:

(3-a-i) When $\rho_M \in (0, |m_i|]$, similar to 1-a), there will be one $\mathcal{PM}$ operation in m_i.

(3-a-ii) When $\rho_M \in \left(|m_i|, \sum_{j=i}^{i+n}|m_j|\right)$ & $\rho_M \neq \sum_{j=i}^{i+n-1}|m_j|$ for some $n \in \mathbb{N}$, there will be 2 $\mathcal{PM}$ operations and $(n-1)$ $\mathcal{M}$ operations in total; when $\rho_M = \sum_{j=i}^{i+n-1}|m_j|$, there will be 1 $\mathcal{PM}$ operation and $(n-1)$ $\mathcal{M}$ operations in total.

(3-b) When $\xi_{Req} = 0$, there will be one more $\mathcal{M}$ operation and one less $\mathcal{PM}$ operation when $\rho_M \geq |m_i|$, the remaining of the operations will stay the same.

(3-c) When $\xi_{Req} = |m_i|$, the discussion will be just parallel to where $\xi_{Req} = 0$ therefore which we will omit here.

An example for the (3-a-ii)-modify operation is shown in Fig.5-7.

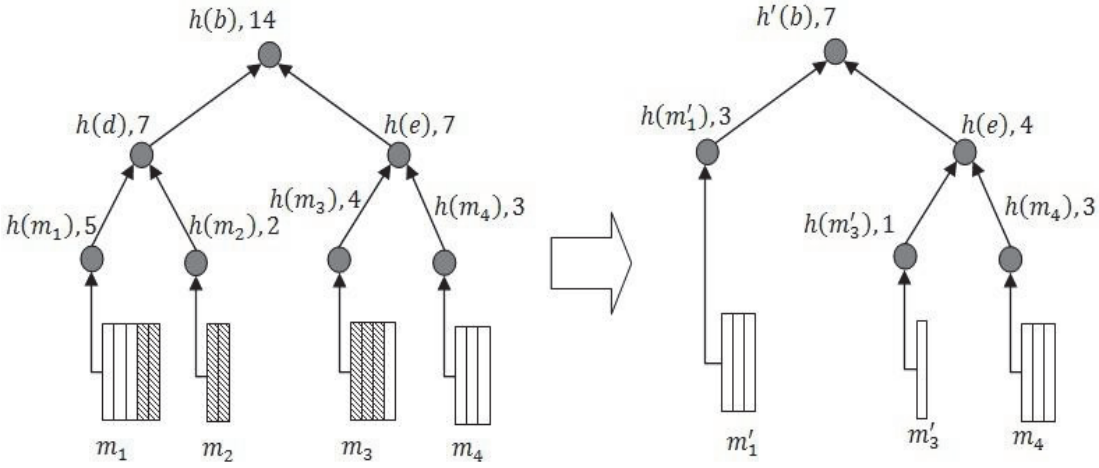


Figure 5-6 Example: fine-grained deletion.

An example of a (2-a-iii)-delete operation for $Req = \{DELETE, 60, 140, NULL\}$ where $\xi_{Req} = 60$, $\rho_D = 2000$ and $i = 1$, in a WMHT where $s_{\max} = 6, \eta = 20$ (bytes).

Theorem 1: Any valid fine-grained update request $FReq$ that is in the form of $\{o, len, m_{new}\}$ can either directly belong to, or be split into some smaller requests that belong to, the following 5 types of block-level update requests: $\mathcal{PM}, \mathcal{M}, \mathcal{I}, \mathcal{D}$ and $\mathcal{SP}$.

Proof: Let $\zeta = \left|len - |m_{new}|\right|$, $\lambda = \min\{len, |m_{new}|\}$, then we can always reduce $FReq$ to Req_1 and Req_2 where $Req_1 = \{Modify, o, \lambda, m_{new}\}$. If $len > |m_{new}|$, then $Req_2 = \{Delete, (o + \lambda), \zeta, 0\}$; if $len < |m_{new}|$, then $Req_2 = \{Insert, (o + \lambda), 0, m_{new}\}$; if $len = |m_{new}|$, then $FReq = Req_1$. Therefore, according to our analysis in this section, Req_1 and Req_2 can be split into the 5 block-level operations in all cases, which concludes our proof. □

Through the analysis above, we know that a large number of small updates, no matter whether they are insert, delete or modify, will always invoke a large number of $\mathcal{PM}$ operations. We now try to optimise $\mathcal{PM}$ operations in the next section to make them more efficient.

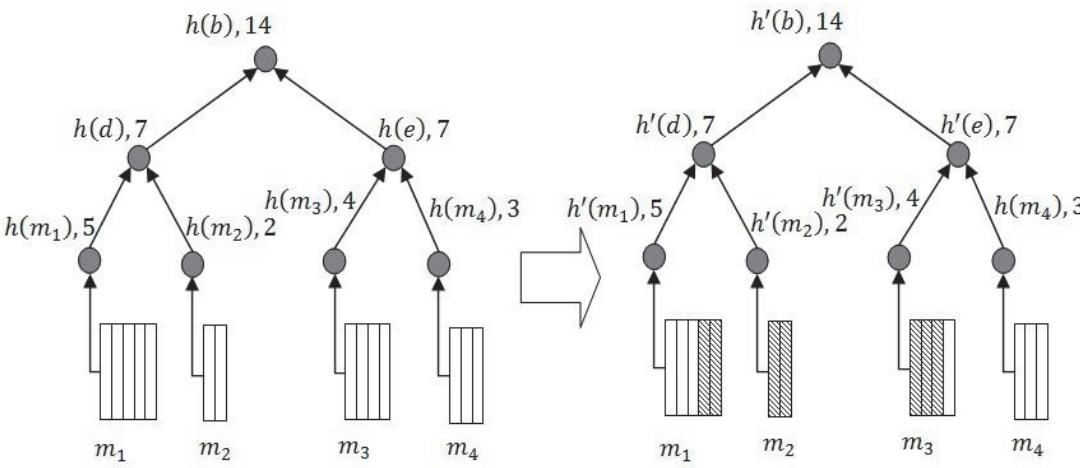


Figure 5-7 Example: fine-grained modification.

An example of a (3-a-ii)-modify operation for $Req = \{MODIFY, 60, 140, m_{new}\}$ where $\xi_{Req} = 60$, $\rho_D = 2000$ and $i = 1$, in a WMHT where $s_{\max} = 6, \eta = 20$ (bytes).

5.4.3 Further Modification for Better Support of Small Updates

Although our proposed scheme can support fine-grained update requests, the client still needs to retrieve the entire file block from CSS in order to compute the new HLA, in the sense that the client is the only party that has the secret key α to compute the new HLA but clients do not have F stored locally. Therefore, the additional cost in communication will be immense for frequent updates. In this section, we will propose a modification to address this problem, utilising the fact that CSS only needs to send back data in the block that stayed unchanged.

The framework we use here is identical to the one used in our scheme introduced in Section 4.2 (which we will also name as 'the basic scheme' hereafter). Changes are made in $PerformUpdate$ and $VerifyUpdate$; Setup, Challenge, Proof Generation and Verification phases are the same as in our basic scheme. Therefore, we will only describe the two algorithms in the following phase:

Verifiable Data Updating: We also discuss $\mathcal{PM}$ operations here first.

$PerformUpdate$: After CSS has received the update request $UpdateReq$

from the client, it will parse it as $\{\mathcal{PM}, i, o, m_{new}\}$ and use $\{o, |m_{new}|\}$ to gather the sectors that are not involved in this update, which we denote as $\{m_{ij}\}_{j\in M}$. CSS will then perform the update to get m_i', then compute R', then send the proof of update $P_{update} = \left\{\{m_{ij}\}_{j\in M}, H(m_i), \Omega_i, R', sig\right\}$ to the client.

$VerifyUpdate$: After the client receives $H(m_i)$, it will first compute R using $H(m_i), \Omega_i$ and verify sig, then it will compute m_i' using $\left\{\{m_{ij}\}_{j\in M}, m_{new}\right\}$ and then computeR_{new} with $\{m_i', \Omega_i\}$ and compare R_{new} with R'. If $R_{new} = R'$, then the client will return $\{\sigma_i', sig'\}$ to CSS for it to update accordingly.

For an $\mathcal{SP}$ operation the process will be the same as our basic scheme as there are no new data inserted into T, therefore the retrieving of the entire data block is inevitable when computations of σ_i' and σ^* are required. For other types of operations, no old data is involved in new blocks; therefore the processes will also remain the same. The process is shown in Fig. 5-8.

5.4.4 Further Discussions

Our strategy can also be applied in RSA-based PDP or POR schemes to achieve authorised auditing and fine-grained data update requests. As RSA can inherently support variable-sized blocks, the process will be even easier. The batch auditing variation in (Wang et al., 2011b, Wang et al., 2010) can also be applied to our scheme, as we did not change the construction of HLAs and the verifications on them.

For the same reason, the random masking strategy for privacy preservation proposed in (Wang et al., 2010) can also be incorporated into our scheme to prevent TPA from parsing the challenged file blocks through a series of integrity proofs to a same set of blocks. Alternatively, we can also restrict the number of challenges to the same subset of data blocks. When data updates are frequent enough, the success rate of this attack will drop dramatically, because there is a high probability that one or

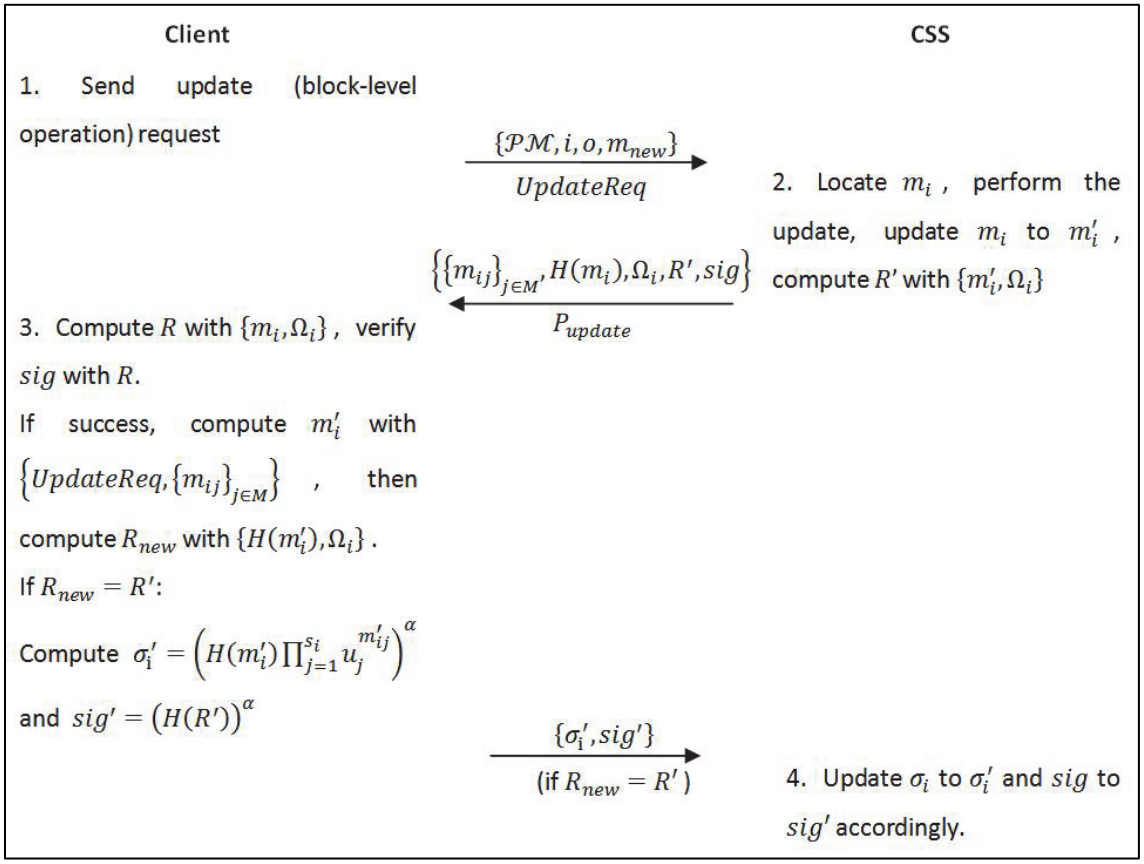


Figure 5-8 Verifiable *PM*-typed data update in modified (final) FU-DPA.

many of the challenged blocks have already updated before c challenges are completed, which is the reason we did not incorporate this strategy into our scheme.

5.5 Security and Efficiency Analysis

5.5.1 Security Analysis

In this section, the soundness and security of our scheme are discussed separately in each phase, as the aim and behaviour of the malicious adversary in each phase of our scheme is different. Our model assumes the following:

Assumption 5.1: CSS will honestly answer all data queries to its clients. In

other words, if a user asks to retrieve a certain piece of her data stored on CSS, CSS will not try to cheat her with an incorrect answer.

This assumption -- reliability -- is a reasonable one because it should be provided as a basic service quality guarantee by all cloud service providers.

Challenge, Verification and TPA Authorisation

In the challenge/verification process of our scheme, we try to secure the scheme against a malicious CSS who tries to cheat the verifier TPA about the integrity status of the client's data, which is the same as previous work on both PDP and POR. In this step, aside from the new authorisation process (which will be discussed in detail later in this section), the only difference compared to (Wang et al., 2011b) is the WMHT and variable-sectored blocks. Therefore, the security of this phase can be proven through a process highly similar to (Wang et al., 2011b), using the same framework, adversarial model and interactive games defined in (Wang et al., 2011b). A detailed security proof for this phase is therefore omitted here.

Security of the new authorisation strategy in our scheme is based on the existential unforgeability of the chosen signature scheme. We first define the behaviour of a malicious third-party auditor.

Definition 3 (Malicious TPA): A malicious TPA is a third party who aims at challenging a user's data stored on CSS for integrity proof without the user's permission. The malicious TPA has access to the entire network.

According to this definition, none of the previous data auditing schemes is resilient against a malicious TPA. Now, in our scheme, we have the following theorem:

Theorem 2: Through the authorisation process, no malicious TPA can cause the CSS to respond with an integrity proof P over an arbitrary subset of file F, namely $m_i, i \in I$, unless there is a negligible probability.

Proof: According to our scheme design, TPA must submit $sig_{AUTH} = \text{Sig}_{ssk}(AUTH||t||VID)$ computed by the client as a part of the challenge message $chal$ to CSS. For a malicious TPA M, all three of the following secret materials are needed to forge a valid authorisation message: client's secret key α, secretly negotiated message $AUTH$, and TPA's ID VID. According to the scheme design, the following two statements are true:

1) Given that the signature scheme we use is existentially unforgeable, M cannot forge any signature for an arbitrary message m without ssk.

2) VID is encrypted with the recipient's public key before being sent to the client or CSS. Therefore, if $|VID| = k_v$, the probability for M to get to know VID is only 2^{-k_v}. Similarly, as $AUTH$ is shared securely between client and CSS, the chance for M to know another's $AUTH$ is also negligible.

Based on the facts above, we know that a malicious TPA M cannot compute a valid sig_{AUTH} on any cloud user's behalf. Even when M intercepts some valid sig_{AUTH} sent by other user, M is still required to send the corresponding VID along with this sig_{AUTH} in order to convince CSS, which also exceeds M's capability. Thus, we can say that our newly added authorisation process is secure against any malicious TPA defined above. □

From this theorem, we can see that the security of a public auditing scheme is strengthened by adding the authorisation process. In fact, the scheme is now resilient against malicious or pretended auditing requests, as well as potential DDOS attacks launched by malicious auditors.

For even higher security, the client may mix in a nonce to the authorisation message to make every auditing message distinct, so that no one can utilise a previous authorisation message. However, this setting may not be appropriate for many scenarios, as the client must stay online when each auditing happens.

Verifiable Data Updating:

In the verifiable updating process, the main adversary is the untrustworthy CSS who did not carry out the data update successfully, but still manages to return a satisfactory response to the client thereafter. We now illustrate the security of this phase of our scheme in the following theorem:

Theorem 3: In the verifiable update process in both our basic scheme and the modification, CSS cannot provide the client with the satisfactory result, i.e., R' cannot match the R_{new} computed by the client with $\{H(m_i'), \Omega_i\}$, if CSS did not update the data as requested.

Proof: 1) In our scheme, the block indices are stored locally and 'findBlock()' are executed also locally by the client. Therefore, there is no added risk in the phase of finding the right file block.

2) In our basic scheme, according to *Assumption 1*, the returned m_i is the correct data block and Ω_i will be the correct AAI metadata associated to m_i. Therefore m_i', computed at client-side with m_i and m_{new}, will be the correct new block stored at CSS after this update. Note that Ω_i contains only unaffected WMHT nodes, which stays unchanged during the update. Therefore, During the updating of m_i, any intentional or unintentional mistake made by CSS will cause R' be different from R_{new}, thereby leading to a failure of client-side authentication.

3) In our modified scheme, CSS will not cheat the client on the query response $\{m_{ij}\}_{j\in M}$ according to *Assumption 1*. Therefore, the new block m_i' will also be the correct block after the update. In this regard, just as in our basic scheme, any mistake made in updating the data will result in $R' \neq R_{new}$. This concludes our proof for the theorem. □

Note that in the verifiable update process, data retrieval is a part of the verifiable update process. According to *Assumption 1*, CSS will respond to this query with the

correct m_{ij}. If not with *Assumption 1*, it is recommended to independently retrieve $\{m_{ij}\}_{j\in M}$ before the update so that CSS cannot cheat the client intentionally, as it cannot distinguish whether the following update is based on this retrieval.

If CSS can be trusted even more, the client may let CSS compute $\left(u_j^{m_{ij}}\right)^{\alpha}$ (where m_{ij} are the sectors that did not change) and send it back to the client, then the client will be able to compute σ' using it along with m_{new} and $H(m_i')$. This will keep the communication cost of this phase on a constantly low level. However, as the CSS is only considered semi-trustworthy and it is difficult for the client to verify $\left(u_j^{m_{ij}}\right)^{\alpha}$ without m_{ij}, this assumption is unfortunately too strong for the majority of scenarios.

5.5.2 Efficiency Analysis

We analyse the efficiency of our scheme from computation, storage and communication perspectives. The efficiency of each phase is analysed separately as follows.

Setup:

Compared to previous schemes, the client will have to keep some additional metadata of F, i.e., the block indices and weight of each node in the WMHT, to decompose a fine-grained update request to several verifiable block-level operations for CSS to perform one by one with $PerformUpdate()$. This data will be kept in the form of a binary tree, structured identically as the WMHT stored along with the original dataset on CSS. CSS has the capability of taking over this task to analyse fine-grained update requests for the client without requiring additional storage, but it is hard for the client to verify whether CSS did it incorrectly either intentionally or by mistake. The data to store one client will not take much storage anyway. In implementation, in consideration of the proof size, $s_{\max}$ cannot be too large. For

example, let the dataset size be 1GB and $s_{\max} = 100$, then there are 500,000 blocks in total. A 2-byte unsigned integer will be sufficient in practice to store the weight for each leaf nodes, and a 4-byte integer is sufficient to store the weight of non-leaf nodes. In addition, for a typical binary tree implementation, 2 pointers with 4 bytes each are also stored on non-leaf nodes. Therefore, even for a full binary tree, only around 0.7% data is stored at the client side for each 1GB of cloud data, which is only 7MB. Considering the computing capabilities of the latest client machines such as PCs, laptops, or even smart phones, this is a small requirement. Note that the client in the dynamic PDP model should have some computing capability anyway, because it should be able to play its role in verifying the update operations and computing HLAs. The amount of data stored on CSS will also increase because the weight of each node is needed to be stored along with every node. However, this increase can be considered negligible compared to the data itself, especially with the storage advantage of our scheme during updates.

TPA Authorization:

In the authorisation process, TPA is the one who needs to be authorised by the client. 3 main steps are included:

1) The client needs to compute sig_{AUTH} and send to TPA. The computational overhead is trivial, because sig_{AUTH} is only computed once, no matter how large the actual data storage is.

2) TPA needs to send sig_{AUTH} along with its ID: VID, which takes no computation.

3) CSS needs to verify sig_{AUTH}. The total number of verifications only depends on the number of users who are requiring the auditing service at a time, not the size of the total data storage. In addition, the overheads of several verifications can be negligible for the powerful hardware of CSS. Therefore, the total overhead for this verification on CSS is also negligible.

From the analysis above, it can be observed that our newly added authorisation process can indeed be considered efficient and practical.

Challenge and Verification:

In existing schemes, a global variable s is applied on all file blocks. The proof size therefore stays constant. As our scheme supports variable-sized blocks, the proof size may fluctuate, because the size of $\{\mu_k\}$ (where $\mu_k = \sum_{i \in I} v_i m_{ik}$) depends on k which further depends on $w = \max\{s_i\}_{i \in I}$. If managed correctly, this difference will not affect the total communication. First, we can restrict the size of file blocks to be chosen for auditing. When updates are conducted frequently where block sizes vary from time to time, this strategy will not affect the overall randomness. Second, we will show in Section 6 that the size of $\{H(m_i), \Omega_i\}$ is a far more influential factor in the total size of the proof. In this regard, the communication overheads of the old and new auditing schemes can be considered the same as in this phase.

Dynamic Data Update:

There are additional overheads at the client-side in splitting a fine-grained update request to block-level operations. However, as discussed before, these can be computed very efficiently using findBlock() (see Fig. 5-4) with $\log(l)$ complexity along with the split method in *Lemma 1*.

Both our schemes have advantages over small updates, which we demonstrated quantitatively in Section 6. In addition to less storage and communications, our WMHT structure will also be more balanced compared to the tree in [6] because small insertions will invoke less block insertions. A more balanced tree structure means less fluctuation and more predictability in the overheads overall.

Our modified scheme introduced in Section 4.5 will always invoke smaller communication overheads in $\mathcal{PM}$ operations, as it only requires retrieval of a part

of the block instead of a whole one. We will test its actual advantage in Chapter 7. Specifically, a detailed quantitative analysis with experimental results for the proposed FU-DPA scheme are provided in Section 7.4.

Chapter 6

MuR-DPA: Secure Public Auditing for Dynamic Multi-replica Big Data Storage on Cloud

This chapter discusses our research presented in (Liu et al., 2014c) where a novel public auditing scheme named MuR-DPA is presented. Theoretical analysis and experimental results show that the proposed MuR-DPA scheme will not only incur less communication overheads for both the update verification and integrity verification of cloud datasets with multiple replicas, but will also provide enhanced security against dishonest cloud service providers. This chapter is organised as follows. Section 6.1 provides an introduction and states the main research contributions of this work. Section 6.2 introduces preliminaries, mainly the novel authenticated data structure RMR-MHT as it is essential in the MuR-DPA scheme. Section 6.3 provides a detailed description for verification of all replicas at once in the proposed MuR-DPA scheme. Section 6.4 presents the process for verifiable updates in the MuR-DPA scheme. Section 6.5 provides related extensions and discussions. Section 6.6 provides a security and efficiency analysis.

6.1 Introduction

In order to improve data reliability and availability, storing multiple replicas along with original datasets is a common strategy for cloud service providers. Public

data auditing schemes allow users to verify their outsourced data storage without having to retrieve the whole dataset. Existing public auditing schemes can already support verification over data which can be subjected to dynamic updates. Such an auditing approach is supported by verifying the auxiliary authentication information (AAI) managed by authenticated data structures (ADS) such as Merkle hash trees (Wang et al., 2011b, Liu et al., 2014b). However, there still exists a number of research gaps in the above mentioned approach. Addressing these gaps is the aim of this thesis. First, existing research lacks investigation of efficient public auditing for dynamic datasets that maintain multiple replicas. Storing multiple replicas is a common strategy for reliability and availability of datasets stored over remote cloud storage. For highly dynamic data, each update will lead to updates of every replica. Given the fact that update verifications in current auditing schemes are of O(log$n$) communication complexity, verifying these replicas one by one will be very costly in terms of communication. Second, current schemes for dynamic public auditing are susceptible to attacks from dishonest servers because of a lack of block index authentication. Although there is an integrity verification scheme for a dataset with replicas (Curtmola et al., 2008) and schemes with index verification such as (Erway et al., 2009), there will be security and/or efficiency problems if these schemes are extended directly to support public verifiability.

In this chapter, we present a multi-replica dynamic public auditing (MuR-DPA) scheme that can bridge the gaps mentioned above through a newly designed authenticated data structure. Research contributions of this work can be summarised as follows:

1. To address the efficiency problem in verifiable updates for cloud storage with multiple replicas, we propose a multi-replica public auditing (MuR-DPA) scheme. The new scheme is based on a novel rank-based multi-replica Merkle hash tree (RMR-MHT). To support full dynamic data updates and authentication of block indices, we include rank and level values in computation of MHT nodes. In contrast to existing schemes, level values of nodes in RMR-MHT are assigned in a top-down

order, and all replica blocks for each data block are organised into the same replica sub-tree. Experimental results show that our scheme can drastically reduce communication overheads for update verification of cloud data storage with multiple replicas.

2. As the previous usage of the Merkle hash tree (MHT) in public auditing of dynamic data did not involve authentication of node indices, such schemes are susceptible to cheating behaviours from a dishonest server. In this work, with the support of RMR-MHT, we propose the first MHT-based dynamic public auditing scheme with authentication of index information that is secure against dishonest servers. The main strategy is top-down levelling and verification of indices from both sides.

3. With RMR-MHT, we have also designed a novel public auditing protocol for verification of all replicas at once. Experimental results show that our scheme not only provides efficient verification for multiple replicas but also incurs less extra storage overhead at the server side.

6.2 Preliminaries

6.2.1 Bilinear Pairing

Like the FU-DPA scheme, bilinear pairing is the foundation for the MuR-DPA scheme and it has already been introduced in Chapter 3. Therefore the details are omitted here to avoid duplication. Please refer to Section 3.1.3 for a detailed introduction of bilinear pairing.

6.2.2 Rank-based Multi-Replica Merkle Hash Tree

A Rank-based multi-replica Merkle hash tree (RMR-MHT) is a novel authenticated data structure designed for efficient verification of data updates, as well

as authentication for block indices. Each RMR-MHT is constructed based not only on a logically segmented file, but also on all of its replicas, as well as a pre-defined cryptographic hash function H. An example of RMR-MHT, constructed based on a file with 4 blocks and 3 replicas, is shown in Fig. 6-1. Similar to an RASL, the rank value $r(v)$ of a node v is defined as the maximum number of nodes in the leaf (bottom) level that can be reached from v. The differences from RMR-MHT and MHT are as follows:

1. Value stored in the leaf nodes are hash values of stored replica blocks. In RMR-MHT, leaf nodes represent replica blocks $b_{i,j}$, namely the jth replica of the ith file block.

2. Value stored in a node v from a none-leaf level is computed from the hash values of its child nodes and two other indices $l(v)$ and $r(v)$. $l(v)$ is the level of node v and $r(v)$ is the rank of v, i.e. the maximum number of leaf nodes that can be reached from *v*. Different to RASL in (Erway et al., 2009), the levels are defined in an top-down order, i.e., the level of root node R is defined as 0, and levels of its child nodes are defined as 1, etc.. The values stored in leaf nodes are $H(1||l(b_{i,j})||H(b_{i,j})$; the value in each none-leaf node is computed as $H(r||l||h_{left}||h_{right})$ where h_{left} and h_{right} denotes the values stored in its left child node and right child node, respectively. In Fig. 6-1, under our definition, $l(b_{i,j})$ (and for all leaf nodes) is 4, $r(b_{i,j}) = 1$. For example, the value h_d is computed as:

$$h_d = H\left(r(d)||l(d)||h(b_{1,1})||h(b_{1,2})\right) = H\left(2||3||h(b_{1,1})||h(b_{1,2})\right)$$

and $h(b_{1,2}) = H\left(1||4||H(b_{1,2})\right)$, $h_b = H(6||1||h_1||h_2)$, etc..

3. The AAI $\Omega_{i,j}$ is different from the MHT in (Wang et al., 2011b) as follows. They now contain not only hash values of the intermediate nodes, but tuples in the format of $\{h, l, q, d\}$, one tuple for each node. h is the hash value stored on this node,

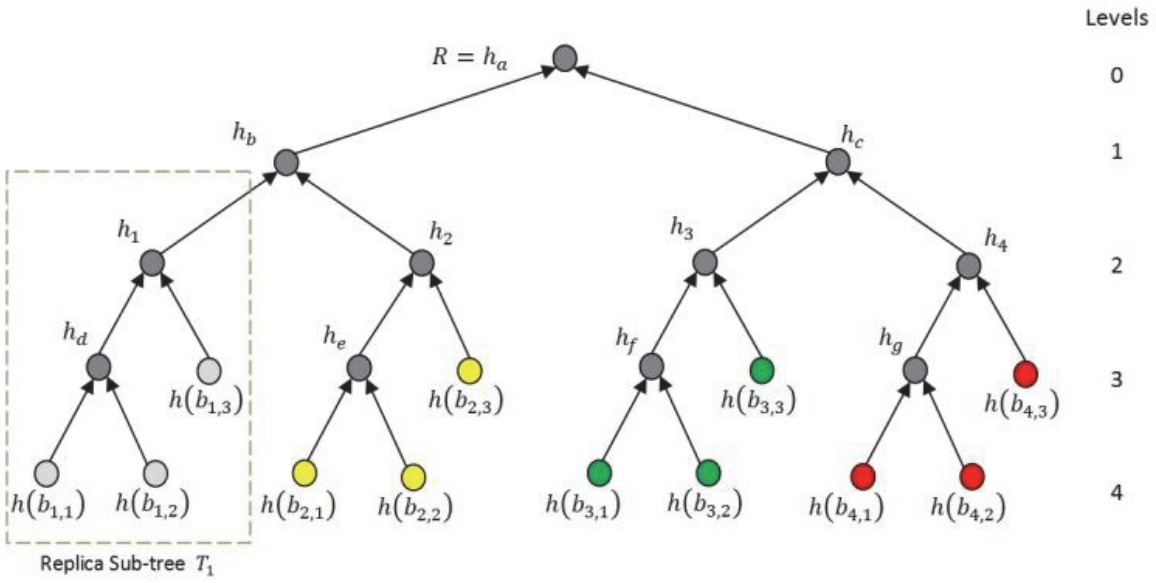


Figure 6-1 An example of RMR-MHT

l is the level of this node, q is the maximum number of leaf nodes reachable from this node, and d is a Boolean value that indicates this node is to the right (0) or left (1) of the node on the verification path, i.e. the nodes from leaf node to the root R. For example, in Fig. 6-1, $\Omega_{2,1}$ for replica block $b_{2,1}$ is defined as $\{(h(b_{2,1}), 4,1,0), (h(b_{2,2}), 4,1,0), (h(b_{2,3}), 3,1,0), (h_1, 2,3,1),$

$(h_c, 1,6,0)\}$, and its verification path is $\{h(b_{2,1}), h(e), h(2), h(b), R\}$.

4. All replicas of one file block are organised into a same sub-tree which we call replica sub-tree (RST), see Fig. 6-1. Note that each RST has the same structure. Each block has exactly c replicas because there are c replica files for the original data file. The total number of leaf nodes for every RST is the total replica number c. Different from (Curtmola et al., 2008), replica blocks are treated independently and each replica block has its own authenticator. The root of each RST, which we denote as h_i, will play a vital role in the newly proposed multi-replica verification and update verification in the following sections. We use Ω_i to denote the AAI for h_i, i.e., one can verify the content and index of h_i with Ω_i and R, similar to $\Omega_{i,j}$ as discussed earlier but this has less hash values. Although roots of RSTs are non-leaf nodes, they can still be authenticated in the same way as leaf nodes. In addition, we

define Γ_i as the set of tuples $\{h, l, q, t\}$ for all intermediate nodes in each RST T_i, where h, l, q are defined the same as above, and t is the sequence number for the nodes, ordered from top to bottom and left to right in T_i. For example, in Fig. 6-1, Γ_2 contains only one node h_e where $\Gamma_2 = \{(h_e, 3,2,1)\}$. As the number of replicas is only a small number (less than 10), for simplicity of description, we assume the structure of T_i is stored at the client (and TPA) side, which applies to every RST and takes only a negligible amount of storage. In this case, the client can compute Γ_i, therefore h_i, based on $h(b_{i,j})$ and $l(b_{i,j})$ without requesting Γ_i from the server. For less client-side storage, the client may also request Γ_i from the server and verify them via $R, h(b_{i,j})$ and Ω_i.

Based on this new ADS, the MuR-DPA scheme will now be described in detail.

6.3 Verification of All Replicas at Once

Setup:

The user and cloud server will first establish common parameters, including a bilinear map $e: G \times G \rightarrow G_T$, and a cryptographic hash function H.

$KeyGen(1^k)$: The client generates a secret value $\alpha \in \mathbb{Z}_p$ and a generator g of G, then computes $v = g^\alpha$ where v, g are the public key and α is the secret key. Another secret signing key pair $\{spk, ssk\}$ is chosen with respect to a designated provably secure signature scheme whose signing algorithm is denoted as $Sig()$. This algorithm outputs $\{ssk, \alpha\}$ as the secret key sk and $\{spk, v, g\}$ as the public key pk.

$FilePreProc(F, sk, c)$:

1) For a dataset to be stored on cloud server, the client will first make c replicas based on the original files. In order to enable the verifiability of these

replicas, they should be different from one another; otherwise, the server may cheat the client by responding to challenges with the correct proofs but actually storing only one replica. From an original file $F = \{m_1, m_2, \dots, m_n\}$, we denote its *j*th replica file as $\tilde{F}_j = \{b_{1,j}, b_{2,j}, \dots, b_{n,j}\}, j \in [1, c]$. The replica blocks $b_{i,j}$ are transformed from m_i, and the transformation is reversible, i.e., the client can recover the original file F through retrieval and the reversed transformation of any replica $\tilde{F}_j$. Therefore, the client does not have to upload F; she can recover F with any intact replica if needed. For example, a method described in (Curtmola et al., 2008) is to choose n pseudo-random functions ψ to compute random values $r_{i,j} = \psi(j||i)$ then output $b_{i,j}$ as $b_{i,j} = m_i + r_{i,j}$; the replicas may also be computed with other methods such as public-key techniques.

2) The client constructs an RMR-MHT based on $b_{i,j}$, computes the root value R, and computes its signature sig with ssk.

3) The client will compute an authenticator $\sigma_{i,j} = \left(H(b_{i,j})u^{b_{i,j}}\right)^{\alpha}$ for every replica block $b_{i,j}$.

Finally, this algorithm outputs $\{b_{i,j}, \sigma_{i,j}, sig\}$, and then uploads them all to the cloud server.

Challenge and Verification:

Within our top-down levelled setting, the verifier will need $H(b_{i,j})$ to verify the auditing equation as it is not stored in the RMR-MHT. Here we discuss how to conduct verification on all replica blocks for a given set of indices in one go.

$GenChallenge(Acc,\ pk,\ sig_{AUTH})$: The third-party auditor TPA generates a challenge message with the given accuracy *Acc*, and sends an authorisation. For example, just as before, for a 99% accuracy, the verifier needs to verify 460 blocks out of a 1GB file. The challenge message is $\{sig_{AUTH}, i, v_{i,j}\}_{i \in I}$ where sig_{AUTH} is for

authorisation, I is the random set of indices chosen for verification, and $v_{i,j}$ are random numbers for integration of $b_{i,j}$.

$GenProof(pk, F, \Phi, chal)$*:* The cloud server will first verify sig_{AUTH}, same as in (Liu et al., 2014b). Then, it will compute $\sigma_j = \prod_{i \in I} \sigma_{i,j}^{v_{i,j}}$ and $\mu_j = \sum_j v_{i,j} b_{i,j}$ for every replica, and send $\left\{\mu_j, \sigma_j, \left\{H(b_{i,j}), h(b_{i,j}), \Omega_i\right\}_{i \in I}, sig\right\}$ back to TPA.

$Verify(pk, P)$*:* Since the verifier knows the structure of RSTs, it will compute R with $\left\{h(b_{i,j}), \Omega_i\right\}$ and verify sig for each ith chosen block. The verification process is similar as in section 4.2.3, with iterative triples and verification of $\xi_\pi = i - 1$ of . Also, it needs to verify the authenticity of $H(b_{i,j})$ by verifying if $h(b_{i,j}) = H\left(1||l(b_{i,j})||H(b_{i,j})\right)$, where $l(b_{i,j})$ can be inferred from $l(h_i)$ which equals level of the first node in Ω_i. For example, in Fig. 6-1, $c = 3$. When we know that $l(h_2) = 2$ from $l(h_1) = 2$ ($h_1$is the first node in Ω_2), we can easily derive $l(b_{2,1}) = 4, l(b_{2,2}) = 4, l(b_{2,3}) = 3$. If these verifications are passed, TPA will trust the retrieved $H(b_{i,j})$ are genuine, then it can verify *c* replicas one by one by verifying the following *c* equations:

$$e(\sigma_j, g) = e\left(\prod_i H(b_{i,j})^{v_{i,j}} u^{\mu_j}, v\right), j \in [1, c]$$

If these equations hold then the verification will output '*ACCEPT*', otherwise it will output '*REJECT*'. The process is demonstrated in Fig. 6-2.

6.4 Efficient Verifiable Updates on Multi-replica Cloud Data

In this chapter, the types of updates considered are whole-block insertion *I*, deletion *D* and modification *M*. These are the minimum requirements for support of

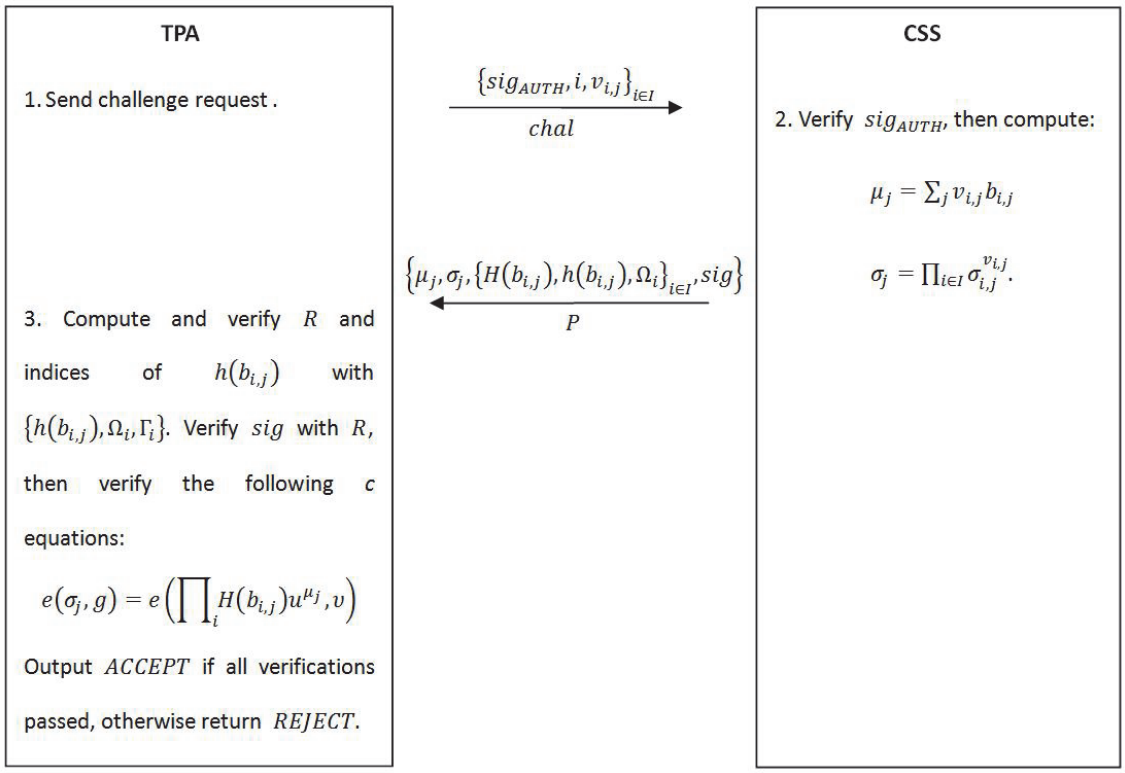


Figure 6-2 Public auditing of all replicas at once

full data dynamics (Erway et al., 2009). In multi-replica scenario, when a block m_i needs to be updated, all its corresponding replica blocks $b_{i,j}$ are also needed to be updated in the same way to maintain consistency. For insertion and modification, the client needs to upload a new data block. As the only one that has the capability to compute replica blocks $b_{i,j}$ based on the original file block m_i, the client will compute the new replica blocks $b'_{i,j}$ then send them to the server along with the update type *I*, *D* or *M*.

$PerformUpdate(UpdateReq)$: The server will parse $UpdateReq$ into $\{Type, i, \{b'_{i,j}\}\}$ and perform the update to file blocks, indices and ADS according to the update request. Specifically, the server will need to update the l value for nodes in insertions and deletions. Note that values in none-leaf nodes in Ω_i stays the same after the update process.

For insertions and deletions, the situations are more complex than in the past

schemes (Erway et al., 2009, Wang et al., 2011b, Liu et al., 2014b). In a traditional MHT, level or rank information is not contained in the nodes; in an RASL, all leaf nodes stay constantly on level 0. Therefore, there is no need to change the hash value in other nodes. In this top-down levelled MHT however, the levels of all leaf nodes in the adjacent RST have also changed by +1 with insertion/-1 with deletion, as the level value is a part in computation of a node value. For example, in Fig. 6-3 (a), with the insertion of $\{b'_{3,j}\}$, the levels of $\{b_{3,j}\}$ have increased by 1, which will cause changes to all $\{h(b_{3,j})\}$; while in Fig. 6-3 (b), with the deletion of $\{b_{3,j}\}$, levels of the old $\{b'_{3,j}\}$ (i.e., old $\{b_{4,j}\}$) have decreased by 1. To output the correct R', these updates are needed to be performed in the hash tree as well. For insertions and modifications, The server will then output $P_{update} = \{\{h(b_{i,j})\}, \Omega_i, R', sig\}$ and return it to the client. For deletions, the server will need to additionally transfer $h'(b_{i,j})$.

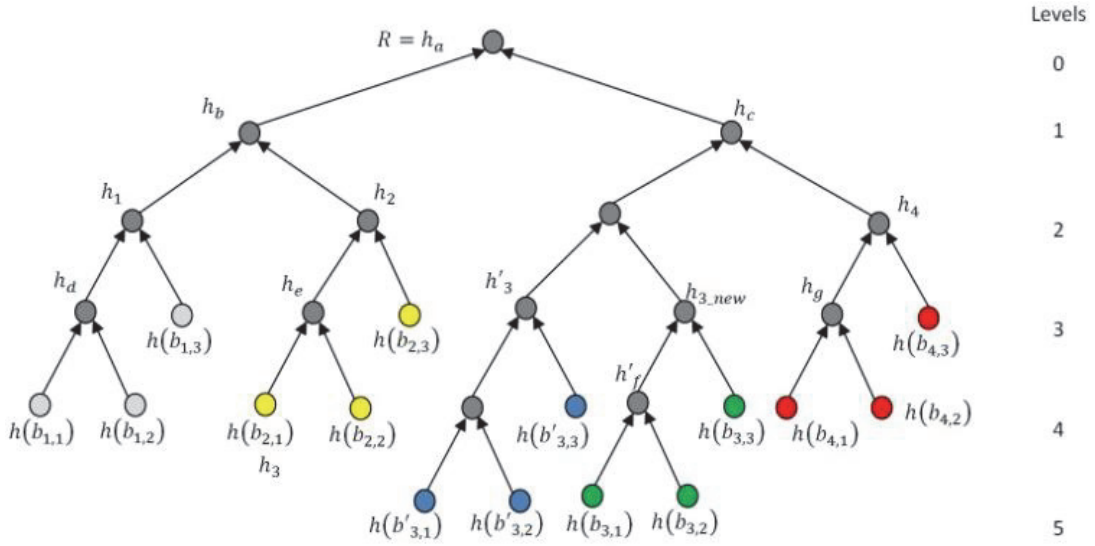
$VerifyUpdate(pk, P_{update})$: In order to verify this update, the client first needs to parse P_{update}. Let the π tuples in Ω_i be (f_k, l_k, q_k, d_k) for each node N_k in an decreasing order of levels, i.e., $l_1 = l(h_i), \ldots, l_{\pi-1} = 2, l_\pi = 1$. A little different from the definition, q_k is the max number of RST roots, instead of leaf nodes, that can be reached from N_k. Since the structure of RST T_i is known to the client, she will be able to compute h_i and h'_i, the old and new roots of T_i, with $h(b_{i,j})$ (got from the server) and $h(b'_{i,j})$ alone respectively.

1. The client will first iteratively compute tuples $(\lambda_k, \eta_k, \xi_k, \zeta_k)$ for nodes on the verification path with nodes N_k in Ω_i as follows, $k = 1, \ldots, \pi$:

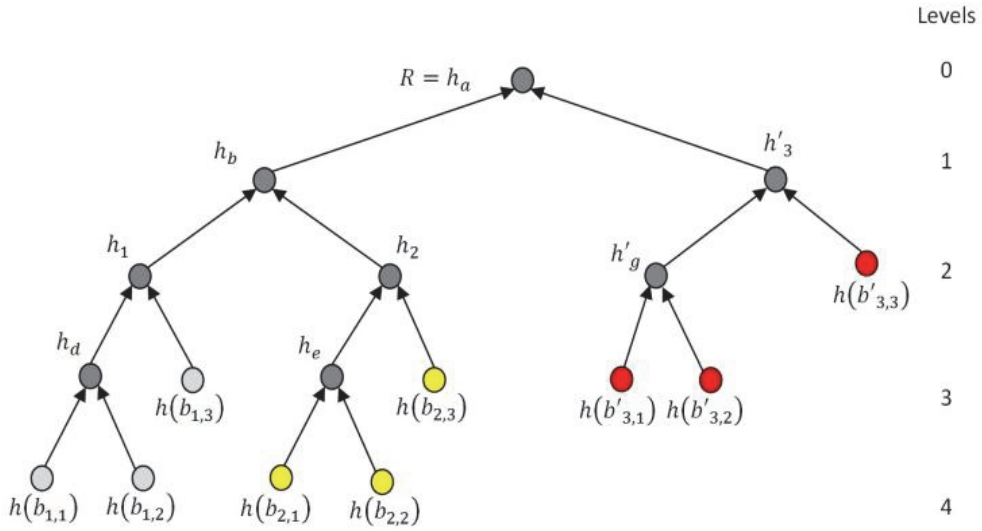
if $d_k = 1$:$\lambda_k = q_{k-1} + \lambda_{k-1}$, $\eta_k = H(\lambda_k||l_k||f_k||\eta_{k-1})$, $\xi_k = \xi_{k-1} + q_k$ and $\zeta_k = \zeta_{k-1}$;

or:

if $d_k = 0$: $\lambda_k = q_{k-1} + \lambda_{k-1}$, $\eta_k = H(\lambda_k||l_k||\eta_{k-1}||f_k)$, $\xi_k = \xi_{k-1}$ and $\zeta_k = \zeta_{k-1} + q_k$



(a)



(b)

Figure 6-3 Update examples to RMR-MHT

Insertion before the 3rd block and deletion of the 3rd block for the RMR-MHT in Fig. 6-1

where $\eta_0 = h(b_{i,j}), \lambda_1 = 1, \xi_0 = 0, \zeta_0 = 0.$

After $\{\lambda_\pi, \eta_\pi, \xi_\pi, \zeta_\pi\}$ is obtained, the client will verify $R = \eta_\pi$ with sig, and verify if $\xi_\pi = i - 1$ and $\zeta_\pi = n - i$ hold at the same time. If the three values pass this authentication, the authenticity of Ω_i (also $b_{i,j}$) and its index i can be confirmed.

2. For deletion, the client needs to verify $h'(b_{i,j})$. Note that $h'(b_{i,j})$ represents the same block and replicas whose root of RST was stored as the first tuple in Ω_i, e.g., in Fig. 6-3 (b), $h(b'_{3,j})$ and $h(b_{4,j})$ represented in the same set of data; the only difference is that $l(b'_{3,j}) = l(b_{4,j}) - 1$. Therefore, the client has enough information to verify $h(b'_{i,j})$ with $h(b_{i,j})$, Ω_i and R. The verification processes are similar to those above. As for insertion, $h(b_{i,j})$ has already been verified along with Ω_i; the client can safely compute the new h_{i_new} without additional verifications, see Fig. 6-3 (a).

3. With RST structure, the client will then compute h'_i with $h'(b_{i,j})$, then compute R_{new} with Ω_i and h'_i and compare R_{new} with R'.

If all 3 verifications are passed, it means that the server has performed the update to all replicas honestly. The client will update the total block number n, then compute $\sigma'_{i,j}$ (the authenticators for $b'_{i,j}$) and store them on the server.

The protocol for verification of updates is demonstrated in Fig. 6-4.

6.5 Discussions and Extensions

Since each replica block $b_{i,j}$ has its own authenticator $\sigma_{i,j}$, our scheme also supports single replica verification. The process will be similar to the verification in (Wang et al., 2011b) with additional verification of $H(b_{i,j})$ and the index of $h(b_{i,j})$. Except for the rank verifications of ξ, ζ are now $\xi_\pi = (i-1)\cdot c + j - 1$ and $\zeta_\pi = (n-i+1)\cdot c - j$. other details will be similar as the verifications described above.

In (Shacham and Waters, 2008), the authors proposed a value s for trade-off of storage and communication overheads. In this strategy, every file block m_i is segmented into *s* segments m_{ik} (length of each segment equals the length of a block without *s*, typically 20bytes), and the authenticators are computed as $\sigma_i =$

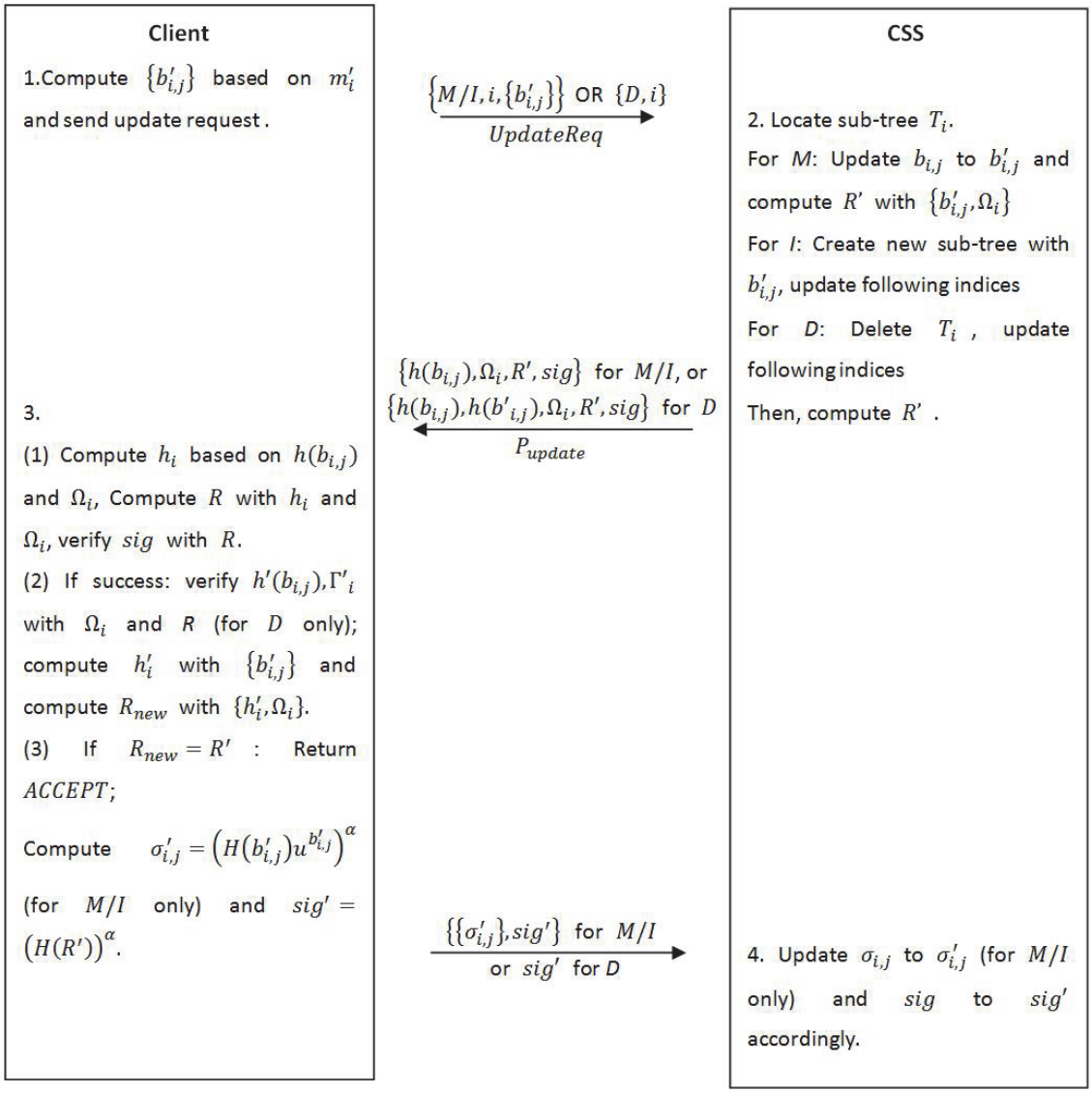


Figure 6-4 Dynamic data update and verification

$\left(H(m_i)\prod_{k=1}^{s} u_k^{m_{ik}}\right)^{\alpha}$. In this case, the proof size has increased by $s\times$ because there will be multiple $\mu_k = \Sigma_i v_i m_{ik}$, instead of one, to be included in the proof. However, the storage overhead has decreased to $1/s$ as there is only one authenticator stored along with s sectors. As our scheme is also based on the BLS signature, with the same block segmentation strategy, the trade-off can easily be applied to our scheme to support dynamic data with multiple replicas. We will show our experimental results under different s values in Section 6.

Based on the segmented blocks, we have investigated fine-grained updates for variable-sized file blocks with different segmentations and WMHT in Chapter 5. If we extend RMR-MHT to let the nodes store the 'rank' information computed from different sizes of blocks, MuR-MHT can also support fine-grained updates and enhance the FU-DPA scheme with efficient support for update of multiple replicas.

Wang et, al. have proposed a random masking technology for privacy protection against the third-party auditor (Wang et al., 2010). In their scheme, the server will mask the proof μ (integrated blocks) with a random r and generate a new $\mu' = r + \gamma\mu$ so that TPA will not learn the users' data from multiple challenging of the same set of blocks. In the multi-replica setting, the proof μ is computed based on replica blocks $b_{i,j}$ instead of the message blocks m_i. Therefore, in most scenarios it is not necessary to apply another masking from the server. Even TPA can infer $b_{i,j}$ from multiple challenges, it will not get any information of the user data m_i without knowing the transformation method, which is known only by the client, from $b_{i,j}$ to m_i. If there is any need to protect replica blocks against the TPA, our scheme can be extended with the same server-side padding strategy.

6.6 Security and Efficiency Analysis

As before, the security of our scheme is based on:

1. Collision-resistance of the hash function,

2. Difficulty of the gap Diffie-Hellman problem, and

3. Unforgeability of the chosen signature scheme.

6.6.1 Verifiable Multi-Replica Updates

Lemma 1. With sig, RST structure, total number of blocks n and a given block index $i \leq n$, if a returned block-AAI combination $\{h_i, \Omega_i\}$ for an RST root passed

the authentication, then either it is computed with the actual replica blocks, or the server has found a way to find collisions in the hash function H.

Proof. The client will first infer $\mathrm{l}(\mathrm{h_i})$, the level of h_i, from Ω_i. Let π be the number of tuples in Ω_i, then $l(h_i) = \pi$. If a dishonest server does not have the ability to find arbitrary collisions of hash functions, it must select an existing node *N* and its corresponding AAI Ω_N in the RMR-MHT in order to let the client compute *R*, thereby verify sig, through iterative hashing. When *N* is not the queried node, i.e., when the server is acting dishonestly, the situation can be covered by the following 3 cases:

1. If *N* is not located on the verification path of h_i, then either the server provides the wrong level or rank values, which will lead to failure in computing the right *R*; or the verification of both values of ξ_π and ζ_π will fail.

2. When the queried node is a left child node, choosing any other hash value and the corresponding AAI from the verification path will let the verification process output the correct ξ_π (the number of file blocks, i.e., leaf nodes, left of this node), but not the correct ζ_π (the number of file blocks, i.e., leaf nodes, right of this node). Therefore, the verification of R will fail.

3. When the queried node is a right child node, choosing any hash value and the corresponding AAI from the verification path will let the verification process output the correct ζ_π, but not the correct ξ_π. The reason is similar to the second case.

Therefore, except for finding hash collisions, the server must return the exact h_i in order to let all three values pass the verification. □

With this Lemma, we can now describe the soundness and security of the update verification process in MuR-DPA through the following theorems.

Theorem 1. If there is any fault to the new data content or index in the server execution of an update request $\left\{Type, i, \left\{b'_{i,j}\right\}\right\}$, the client verification will fail.

Proof. According to Lemma 1, the RST root h_i and its AAI Ω_i returned by the server are the correct representatives for the RST where $\{b_{i,j}\}$ has resided, otherwise the verification of R will fail.

1. For insertions and modifications, if $b'_{i,j}$ was updated incorrectly, then h'_i, therefore R', will be computed incorrectly due to the collision resistance of hash function H. According to the property of MHT, Ω_i stays the same throughout the update. As the client has the right $b'_{i,j}$ and Ω_i, the values h'_i and R' at client side will be correct. Therefore, the verification will fail.

2. For deletions, the returned $h(b'_{i,j})$ will be incorrect once there is any fault in this update. As $h(b'_{i,j})$ is included in the $\Omega(b_{i,j})$, the client will identify the abnormality if $h(b'_{i,j})$ is incorrect.

Therefore, through the verification, the client will be able to detect any fault caused by accidental or dishonest behaviours in the update. □

This concludes the proof that the MuR-DPA scheme can support public auditing of dynamic data without being cheated by a dishonest server. As for efficiency, the AAI Ω_i will take the majority of data transfer because it is composed of log(n) hash values and rank/level information for each update. For updating of multiple replicas (which is a must for cloud storage with multiple replicas), only one, instead of c AAIs, is needed to be transferred for verification of c replica blocks. Therefore, the more replicas there are, the more efficiency advantages our scheme will have.

6.6.2 All-at-once Multi-Replica Verification

Just as in the verification of updates, there is need for verification of Ω_i.

Theorem 3. In the MuR-DPA scheme, if integrity of any replica $b_{i,j}$ of the i-th block was breached, the server cannot build a response

$\left\{\mu_j, \sigma_j, \left\{H\left(b_{i,j}\right), h\left(b_{i,j}\right), \Omega_i\right\}_{i\in I}, sig\right\}$ that can successfully pass the verification, unless any of the 3 assumptions at the beginning of this section fails to hold.

Proof. As the structure of RST is known by the verifier, the verifier will be able to re-build the RST under $h(i)$, and thereby compute $h(i)$ based on $h\left(b_{i,j}\right)$. With Lemma 1, the authenticity of $\{h_i, \Omega_i\}$ can be verified via sig, i and n. Therefore, if $h\left(b_{i,j}\right), l\left(b_{i,j}\right), r\left(b_{i,j}\right)$ are not all correct, then $h(i)$ will be incorrect; with Ω_i, the verification for R will fail. Because $h\left(b_{i,j}\right)$ was computed with $r\left(b_{i,j}\right), l\left(b_{i,j}\right)$ and $H\left(b_{i,j}\right)$, if all these 3 values are correct, then the returned $H\left(b_{i,j}\right)$ must be correct, otherwise the client will fail to verify the equation $h\left(b_{i,j}\right) = H\left(r\left(b_{i,j}\right)||l\left(b_{i,j}\right)||H\left(b_{i,j}\right)\right)$. Therefore, our design can make sure the returned $H\left(b_{i,j}\right)$ are indeed the hash values of the designated replicas for the ith block. On the other hand, the soundness and security of the verification equation $e\left(\sigma_j, g\right) = e\left(\prod_i H\left(b_{i,j}\right)^{v_{i,j}} u^{\mu_j}, v\right)$ itself has already been proven in (Shacham and Waters, 2008) and (Wang et al., 2011b). Therefore, any integrity breach will be identified with MuR-DPA. □

The proof above is based on the assumption that the verifier knows the structure of RST. In fact, even when the RST structure was unknown to the verifier, the verification for all replicas may still be resilient to dishonest servers as exchanging the orders of replicas under an RST does not affect the verification. We leave this problem for future work.

Our scheme is also based on the Merkle hash tree. Therefore, just as in past schemes, the proof size is also dependent of the data size and number of data blocks. As a drawback, RMR-MHT introduced more levels (depth of RSTs) than each MHT in SiR-DPA to store replica blocks. Therefore, the verification cost for one replica in MuR-DPA will be slightly larger than in SiR-DPA. However, as the replica number is small (usually less than 10), the depth of RSTs is constant (usually only less than 4 levels). Therefore, there is no significant additional overhead for the client to verify a

single replica. Details will be discussed in the Chapter 7. Specifically, a detailed quantitative analysis with experimental results for the proposed MuR-DPA scheme are provided in Section 7.5.

Chapter 7

Experimental Results and Evaluations

In this chapter, experimental results and evaluations of our schemes presented above are provided. It can be inferred from these results and analyses that our schemes are significantly more efficient than existing schemes. The chapter is organised as follows. Section 7.1 provides a qualitative comparison of our public auditing schemes against existing representative schemes. Section 7.2 introduces our experimental environment. Section 7.3 provides results on key exchange schemes to demonstrate the importance of research on key exchange in cloud and public data auditing, as well as the efficiency improvement of our KE schemes CCBKE and HKE-BC which were introduced in Chapter 4. Section 7.4 presents experimental results of our FU-DPA scheme which was introduced in Chapter 5. Section 7.5 presents experimental results of our MuR-DPA scheme which was introduced in Chapter 6.

7.1 Qualitative Comparison of Public Auditing Schemes

We first provide a brief comparison between our schemes and existing schemes regarding certain properties in public auditing and verification of outsourced data. These properties include not only existing ones such as blockless and stateless verification, public verifiability etc., but also new properties introduced in this thesis such as authorised auditing, fine-grained updates and multi-replica public auditing. Please refer to Table 7-1 and Table 7-2 for details, where the improvements of our new schemes are demonstrated.

7.2 Experimental Environment

	POR (Juels and B. S. Kaliski, 2007)	PDP (Ateniese et al., 2007)	Scalable PDP (Ateniese et al., 2008)	Compact POR (Shacham and Waters, 2008)	MR-PDP (Curtmola et al., 2008)
Blockless Verification	No	Yes	Yes	Yes	Yes
Stateless Verification	No	Yes	Yes	Yes	Yes
Infinite Verifications	No	Yes	No	Yes	Yes
Public Verifiability/Auditability	No	Yes	No	Yes	No
Coarse-grained Verifiable Data Updating	No	No	Partly	No	No
Fine-grained Verifiable Data Updating	No	No	No	No	No
Variable-sized Data Blocks	No	Yes	Yes	No	Yes
Authorised Auditing	No	No	No	No	No
Authentication of Block Indices (for schemes with ADS)	N/A	N/A	N/A	N/A	N/A
One Interaction for Updating	No	No	No	No	No

We conducted all our experiments on U-Cloud -- a cloud computing environment located in the University of Technology, Sydney (UTS). The computing facilities of this system are located in several labs in the Faculty of Engineering and IT, UTS. On top of hardware and Linux OS, We installed KVM Hypervisor which virtualises the infrastructure and allows it to provide unified computing and storage resources. Upon virtualised data centers, Hadoop is installed to facilitate the MapReduce programming model and distributed file system. Moreover, we installed the OpenStack open source cloud platform which is responsible for global management, resource scheduling, task distribution and interaction with users (For experiments with CCBKE, the platform was Eucalyptus). The structure of U-Cloud is demonstrated in Fig. 7-1.

	DPDP (Erway et al., 2009)	SR-DPA (Wang et al., 2011b)	FU-DPA (Liu et al., 2014b), Ch. 5	MuR-DPA (Liu et al., 2014c), Ch. 6
Blockless Verification	Yes	Yes	Yes	**Yes**
Stateless Verification	Yes	Yes	Yes	**Yes**
Infinite Verifications	Yes	Yes	Yes	**Yes**
Public Verifiability/Auditability	No	Yes	Yes	**Yes**
Coarse-grained Verifiable Data Updating	Yes	Yes	Yes	**Yes**
Fine-grained Verifiable Data Updating	No	No	Yes	**Capable**
Variable-sized Data Blocks	Yes	No	Yes	**Yes**
Authorised Auditing	No	No	Yes	**Yes**
Authentication of Block Indices (for schemes with ADS)	Yes	No	No	**Yes**
One Interaction for Updating All Replicas	No	No	No	**Yes**

7.3 Experimental Results for Key Exchange Schemes

7.3.1 Comparison of Key Exchange schemes

In this section, we will compare the time consumption of the IKE key exchange scheme to the encryption time and public auditing time (specifically, proof generation time) through a series of experimental results. Through this comparison, the necessity of research on efficient key exchange schemes is demonstrated.

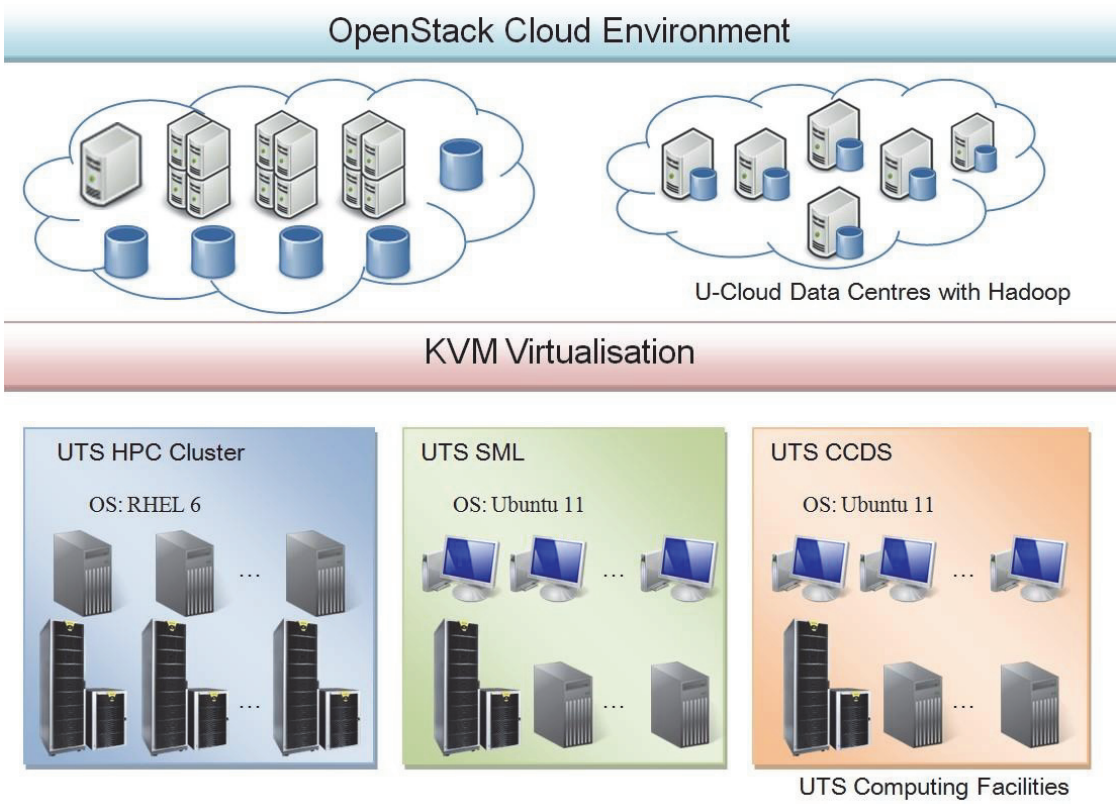


Figure 7-1 U-Cloud environment

Key exchange vs. encryption

Key exchange schemes are accompanied by symmetric encryptions. We first show that key exchange schemes take a large percentage of run time when running under cloud computing, which indicates the significance of research on efficient authenticated key exchange schemes. When applying hybrid encryption to traditional data-intensive applications, key exchange schemes are always being utilised in combination with symmetric-key encryption to ensure data security. In these scenarios, time consumption of key exchange schemes can be neglected compared to the heavy time consumption on encryption. However, the situation is different in cloud computing, and we have demonstrated the difference. A cloud computing infrastructure often employs thousands of server instances. For time-critical data-intensive applications such as scientific applications, datasets in gigabytes are split into blocks in megabytes and then distributed and executed on server instances through MapReduce. We use IKE time consumption data from our experiment to

represent the efficiency of the key exchange scheme. For symmetric-key encryption algorithms, we include two algorithms used for dataset encryption: the most widely-recognized block cipher, *AES*, in Galois Counter Mode (GCM) with 64K tables, and *Salsa20/12*, a stream cipher which is a more popular kind in encrypting large datasets because of its high efficiency. Both of the two algorithms are proven to be secure against various kinds of cryptanalysis. For the efficiency of encryption algorithms, we refer to the performance result from Crypto++ benchmarks which indicates the speed of *AES/GCM* with 64K tables is 108MB/s, and the speed of *Salsa20/12* is 643MB/s for data encryption. Experiments are conducted on several datasets taken from astrophysics research; and results are listed in Tables 7-3 and 7-4.

From the results in the tables we can see that with increased dataset size and a number of involved server instances, the time consumption of key negotiation increases more rapidly than that of data encryption itself. We can also infer from the results that, in a hybrid cloud computing environment, key exchange operations in a hybrid security scheme do indeed take a large percent of time consumption[4]. This means that the cost of key exchange will indeed take a considerable percentage in terms of security-aware large-scale cloud computing applications. In other words, the overall performance of such applications will be significantly improved if a key exchange scheme with better efficiency is used.

Key exchange vs. public auditing

For public auditing schemes, as indicated in Chapter 3, distributed AKE schemes must be applied before proof computation if data is encrypted for security. For evaluation, we take the FU-DPA scheme introduced in Chapter 5 as an example. To ensure 99% assurance, 460 blocks are needed to be challenged. When all blocks are stored on separate instances, KE operations will take 7,187ms when the IKE

[4] This percentage will be even higher in real-world scenario since KE efficiency depends heavily on scheduling algorithm and network status while encryption time stays relatively constant.

Dataset Size (GB)	2	8	12	15	32
Server Instances Involved	100	500	1000	1500	4000
Data Block Size (MB)	20	16	12	10	8
AES/GCM Encryption Time (s)	18.52	74.07	111.11	138.89	296.31
IKE Key Exchange Time (s)	4.04	20.48	41.77	61.92	163.10
Key Exchange Take Percentage of (%)	**17.91**	**21.66**	**27.32**	**30.84**	**35.50**

Table 7-3 Time consumption comparisons of IKE and AES encryption on CLC.

Dataset Size (GB)	2	8	12	15	32
Server Instances Involved	100	500	1000	1500	4000
Data Block Size (MB)	20	16	12	10	8
Salsa20/12 Encryption Time (s)	3.11	12.44	18.66	23.33	49.77
IKE Key Exchange Time (s)	4.04	20.48	41.77	61.92	163.10
Key Exchange Take Percentage of (%)	**56.50**	**62.21**	**69.12**	**72.63**	**76.62**

Table 7-4 Time consumption comparisons of IKE and Salsa encryption on CLC.

scheme is applied, whereas proof computation at CLC afterwards takes only 520ms. Therefore, it can be inferred from these experiments that an efficient KE scheme is also of great importance to the efficiency of a security-aware public auditing scheme, as long as data encryption is needed for data transfer inside the cloud.

7.3.2 Efficiency improvements of CCBKE and HKE-BC

We implemented HKE-BC, CCBKE and IKE schemes using C++ with

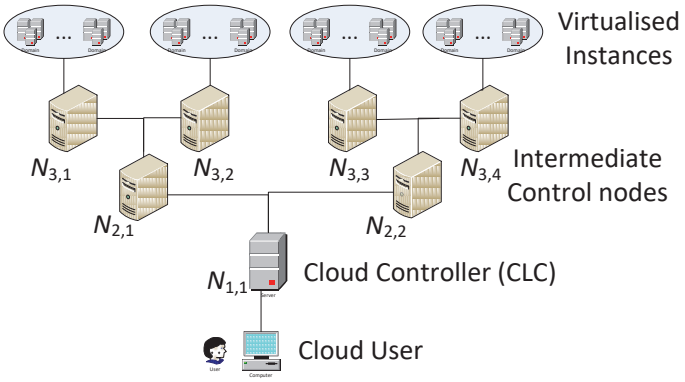
MIRACL cryptography library, and tested them on our U-Cloud environment. On each cloud instantiation, we repeatedly ran each of the key exchange scheme 20 times to simulate a large-scale computation task with 20 rounds of CLC-VM interactions, each encrypted with different keys. The time consumptions of key exchange are recorded and demonstrated in Figures 7-3 and 7-4.

As the HKE-BC scheme performs differently on different cloud layouts, we tested our scheme under several differently structured cloud instantiations of U-Cloud. The layouts of the two experimental cloud scenarios are shown in Figure 7-2.

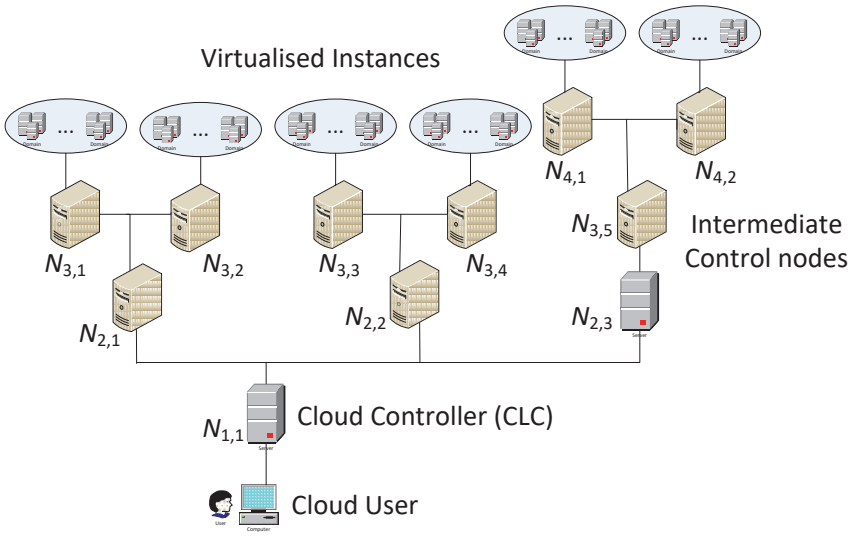
Experimental results are shown in Figure 7-3 where we demonstrate time consumption of key exchange operations in the two different cloud scenarios shown in Figure 7-2 with an increasing number of instances launched. Cloud instantiation A have 3 control layers and 4 NCs where all control nodes are evenly distributed, while the instantiation B simulates a hybrid cloud with an uneven structure, a total of 4 control layers and 6 NCs. The numbers of instances launched by each NC are $5, 10, 15, \ldots, 50$, consecutively.

We can see from these results that in both cloud scenarios, CCBKE has a significant efficiency improvement against the widely-adopted IKE scheme. Further, CBHKE outperforms CCBKE and IKE in terms of time efficiency. Compared to IKE in U-Cloud, the total average time consumption of KE in CCBKE is decreased to 52.9% and 51.5% in scenarios A and B, respectively. For HKE BC, the total average time consumption in KE is decreased by an average of 85.9% and 89.8% in scenarios A and B, respectively. This efficiency advantage of HKE-BC when compared to CCBKE in the two scenarios is 70.96% (max: 75.9%; min: 58.9%) and 77.85% (max: 82.4%; min: 61.3%), respectively. This is a significant improvement in efficiency without compromising the level of security. The results match our efficiency analysis in section 4.

More results are shown in Figure 7-4. We tested the key exchange schemes under cloud environments with different numbers of control layers: 1, 2, 3 and 4, with



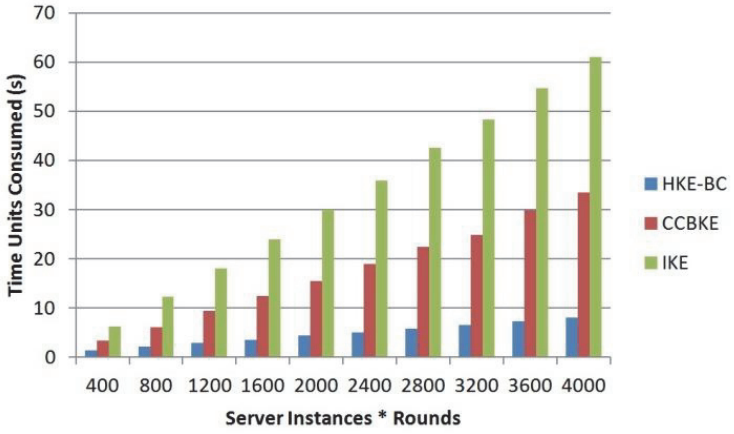
Cloud instantiation A: $l = 3$



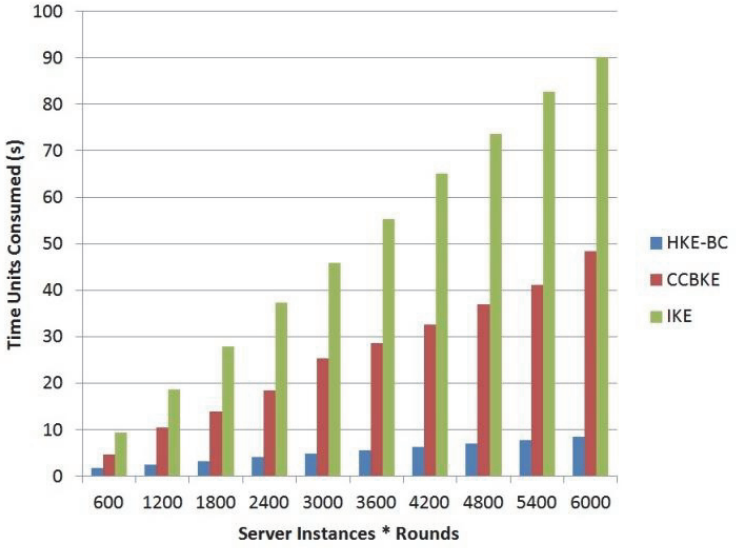
Cloud instantiation B: $l = 4$

Figure 7-2 Structures of two cloud instantiations of U-Cloud.

a fixed total number of instances (100) launched. The number of sub-nodes for each control level (except for end node controller) is chosen as $n = 2$, because for a larger n value the number of servers needed grows at a speed of a geometrical series, which is far from practical scenario. All instances are evenly distributed on every node controller. U-Cloud instantiation A (see Figure 7-2) is an example of such cloud layouts where there are 3 control levels and each NC launches 25 instances. We can see that when compared to its predecessors IKE and CCBKE, the HKE-BC scheme can drastically reduce time consumption in key exchange for most cloud layouts. The



Results in scenario A



Results in scenario B

Figure 7-3 Time efficiency of HKE-BC, CCBKE and IKE in the two cloud instantiations.

more layers the cloud environment has, the more efficient the new scheme is. The time consumption can even be decreased down to only 9.2% of the time when $l = 5$. However, note that an increase in the layer number will also cause an increase in the total network throughput, along with difficulty in terms of scheduling and total robustness. This is why practical clouds do not usually employ too many control layers, and also why we did not test a cloud with more control layers. Nevertheless,

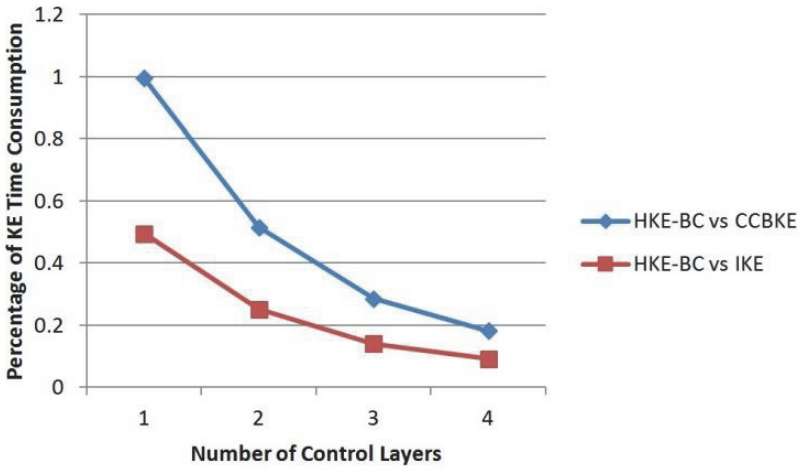


Figure 7-4. Efficiency advantage of HKE-BC

Efficiency advantage of HKE-BC with different numbers of control layers (l) where $n = 2$

results show that the new scheme always has a vast efficiency advantage over the existing schemes in most hierarchical cloud environments.

7.4 Experimental Results for FU-DPA

We implemented both our scheme and its modification on U-Cloud, using a virtual machine with 36 CPU cores, 32GB RAM and 1TB storage in total. As in previous work (Wang et al., 2011b, Erway et al., 2009), we also used a 1GB randomly generated dataset for testing. The scheme is implemented under 80-bit security, i.e., $\eta = |p| = 160$bits. As the number of sectors s (per block) is one of the most influential metrics to overall performance, we will use it as our primary metric. For saving of the first wave of allocated storage, we used $s_i = s_{\max}$ in the initial data splitting and uploading. Note that $s_{\max}$ decides the total number of blocks for an arbitrary $|F|$. However, according to (Ateniese et al., 2007), the number of authenticated blocks is a constant with respect to a certain percentage of file tampered and a certain success rate of detection, therefore we will not take the number of audited blocks as our primary variable of measurement. All experimental results are an average of 20 runs.

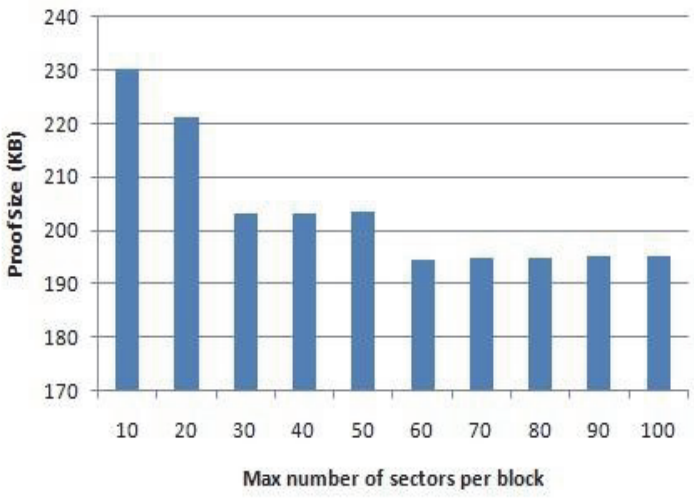


Figure 7-5 Auditing communication overhead in FU-DPA for different block size.

Communication overhead invoked by an integrity proof with 80-bit security under different s_{max} for verifying a 1GB dataset.

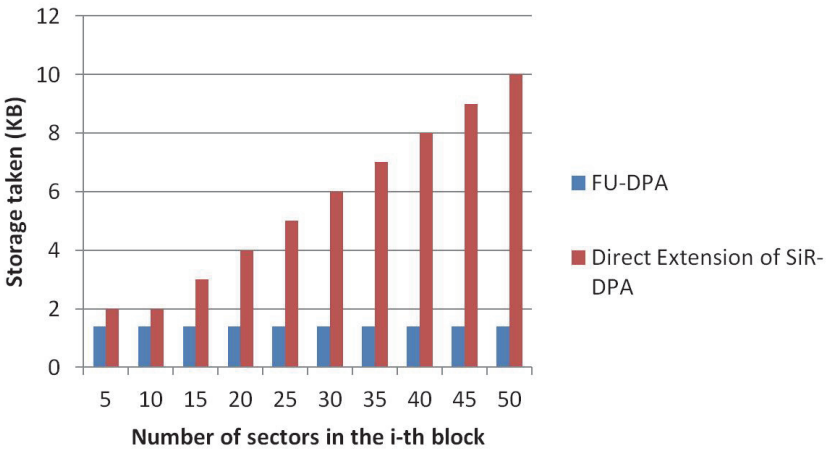


Figure 7-6 FU-DPA: Comparison of storage overhead.

Comparison of the total storage overhead invoked by 10* 140-byte insertions to the *i*-th block in FU-DPA, as opposed to the direct extension of SiR-DPA.

We first tested how $s_{\max}$ can influence the size of proof P, which is missing in former schemes (Wang et al., 2011b, Wang et al., 2010). From Fig. 7-5, we can see that generally the proof size decreases when $s_{\max}$ increases, because the average depth of leaf nodes m_i of T decreases when $s_{\max}$ increases to a certain

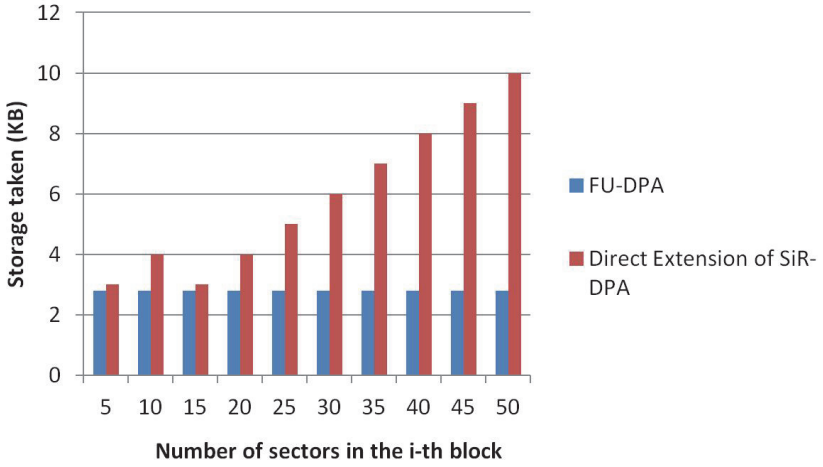


Figure 7-7 FU-DPA: Comparison of storage overhead (continued).

Comparison of the total storage overhead invoked by 10* 280-byte insertions to the *i*-th block in FU-DPA, as opposed to the direct extension of SiR-DPA scheme

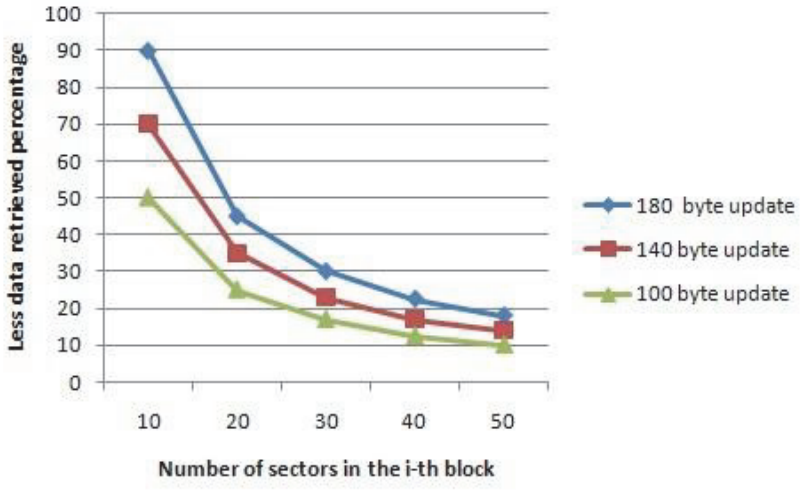


Figure 7-8 FU-DPA: Reduction of communication overhead.

The percentage in saving of communication overhead in data retrieval in the modified / final FU-DPA, compared to the first proposal.

level, especially when right after the initial uploading of $\mathcal{F}$. Note that the storage of HLA and WMHT at the CSS side will also decrease with the increase of the average number of blocks. Therefore, a relatively large $s_{\max}$ (but not too large, which we will discuss along with the third experiment) is recommended in our dynamic

setting.

Second, we tested the storage overhead for small insertions. Without support for fine-grained updates, every small insertion will cause creation of a whole new block and an update of related MHT nodes, which is why our scheme has an efficiency advantage. We compared our scheme against a representative (and also recent) public auditing scheme (Wang et al., 2011b). For comparison, we extended the older scheme a bit to let it support the communication-storage trade-off introduced in (Shacham and Waters, 2008) so that it can support larger file blocks with multiple (but only a predefined constant number of) sectors each. The updates chosen for experiments are 10*140 Bytes and 10*280 Bytes, filled with random data. Results are shown in Fig. 7-6 and Fig. 7-7. For updates of the same total size, the increased storage on CSS for our scheme stays constant, while in the extended old scheme (Wang et al., 2011b) (see Section 3.2.2) the storage increases linearly with the increase in size of the affected block. These results demonstrated that our scheme with fine-grained data update support can incur significantly lower storage overhead (down to $0.14\times$ in our test scenarios) for small insertions when compared to existing scheme.

Third, we investigated the performance improvement of the modification introduced in Section 4.5. We used 3 pieces of random data with sizes of 100 bytes, 140 bytes and 180 bytes, respectively, to update several blocks that contain 10 to 50 standard 20-byte sectors each. Data retrieval is a key factor of communication overheads in the verifiable update phase. For each update, we recorded the total amount of data retrieval for both our modified scheme and our basic scheme. The results in comparison are shown in Fig. 7-8. We can see that our modified scheme always has better efficiency with respect to data-retrieval-invoked communication overheads, and the advantage is more significant for larger updates. However, for an update of the same size, the advantage will decrease with the increase of $|s_i|$ where a larger number of sectors in the original file are needed to be retrieved. Therefore, the block size needs to be kept low if less communication in verifiable updates is

demanded.

From the experimental results on small updates, we can see that our scheme can incur significantly lower storage overhead while our modified scheme can dramatically reduce communication overheads compared to the existing scheme. In practice, the important parameter $s_{\max}$ should be carefully chosen according to different data size and different efficiency demands in storage or communications. For example, for general applications with a similar scale (1GB per dataset and frequent 140-byte updates), a choice of $s_{\max} = 30$ will allow the scheme to incur significantly lower overheads in both storage and communications during updates.

7.5 Experimental Results for MuR-DPA

For quantitative evaluations of MuR-DPA introduced in Chapter 6, we provide experimental results to demonstrate the improved efficiency of MuR-DPA when deployed on cloud data storage. We compare our new scheme, MuR-DPA, against the direct extension of the existing scheme in (Wang et al., 2011b) with tags of each replica indexed in separate MHTs and the MHTs have levels and ranks for index authentication. We name this scheme SiR-DPA - Dynamic Public Auditing with Separately-indexed Replicas. We implemented both schemes on U-Cloud, using a virtual machine with 36 CPU cores, 32GB RAM and 1TB storage in total. The design of public auditing schemes does not take into account the content of data. Therefore, as in previous work, we used a 1GB randomly generated dataset for each testing, with the replicas computed as $b_{i,j} = m_{i,j} + r_{i,j}$. BLS parameters are chosen with 80-bit security, i.e., the length of order of G is 160 bits. All experimental results are an average of 20 runs.

As in previous studies, the computation time is not the primary concern in our new scheme, because the challenged blocks are a constant value regardless of the file size, and the time consumption in proof computation or proof verification only takes less than 1 second. Therefore, we will mainly focus on measuring the communication

s (number of sectors per block)	Data Updated (MB)	Total Server Response for Verification (MB)
1	512/1024	19.507
5	512/1024	3,625
10	512/1024	1,743
20	512/1024	837
50	512/1024	321
100	512/1024	154

Table 7-5 Price of dynamism.

Communication Overhead for Verifying Updates of Half Blocks in a 1GB File.

and storage costs, especially those incurred in the verification of updates.

We first measured the communication overhead for the verification of updates. Table 7-5 shows the total communication overhead for update verification of only one replica, where overheads of SiR-DPA and MuR-DPA are the same. The testing dataset is 1 GB and we update half of the blocks with 512MB new content in total; with adjusting parameter s. Communication overhead for update verification in the protocol in (Erway et al., 2009) and the MHT-based scheme in (Wang et al., 2011b) will be similar to our SiR-DPA setting, as the communication complexities in MHT and RASL are both $O(\log n)$ with high probability (whp). Note that in this experiment, there is only one update for each block for all modifications. Under this setting, we can see that this overhead is always a heavy burden. Even for a large $s = 100$, there is still 154MB verification data which is needed to be transferred from the server for update of the size 512MB. Although the communication overhead will decrease for a larger block size (because the number of blocks will be smaller), it may take several update processes to update half of its content, where the communication overhead will increase beyond the amount in Table 7-5. To make things worse, with multiple replicas, the SiR-DPA scheme will multiply this communication overhead,

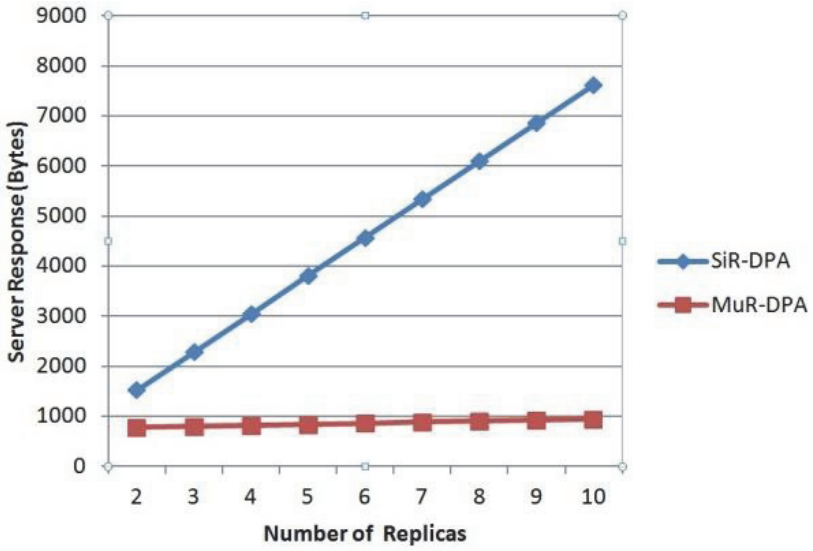
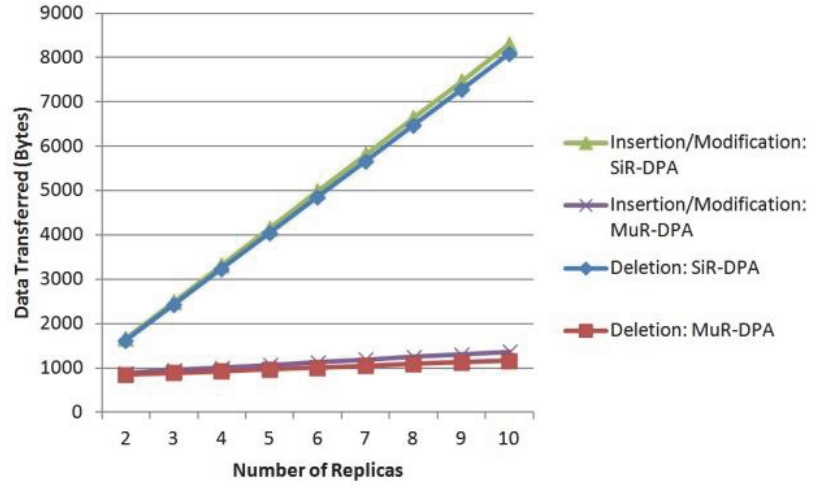


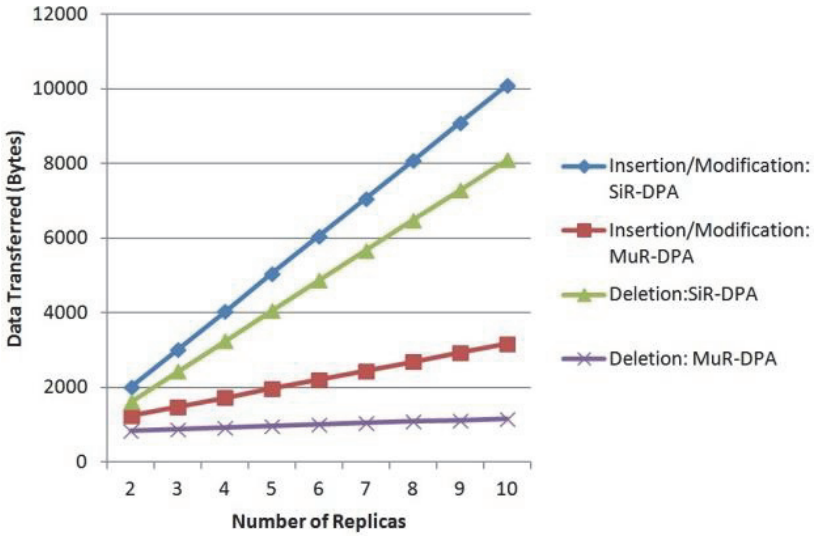
Figure 7-9 MuR-DPA: Length of server response for one verifiable modification/insertion of one block.

which has to be avoided if possible, given the fact that cloud service providers always keep multiple replicas for storage services.

Second, we tested the communication overhead for updates with different numbers of replicas and different sizes of blocks. Results are depicted in Figs. 7-9 and 7-10. From Fig. 7-9, we can see that the length of server response for modification and insertion has been greatly reduced when there are multiple replicas, which means the load and utilisation server's crucial downlink bandwidth will be comparatively less. It is clear that MuR-DPA will scale gracefully with increases in number of replicas of the dataset. We can also safely conclude that overheads for deletions will be similar as there is only one more hash value to be included in the server response. Therefore, evaluation for the deletion operation is omitted here. The total communication overheads for verification of updates to datasets with multiple replicas are also tested. For block insertion and modification, the new data block needs to be uploaded. Therefore, for a larger s, (i.e. a larger block size), the total communication cost will rise. For block deletion, nothing needs to be uploaded since there is no new data block. Therefore, the total communication overhead for a single deletion stays unchanged with different s values. Either way, for $s = 1$ and $s = 10$, our results show that communication overheads of verification of updates in MuR-DPA always have



(a)



(b)

Figure 7-10 MuR-DPA: Total communication for one verifiable update.

Total communication overhead for one verifiable update of one block when (a) $s = 1$; (b) $s = 10$

significant advantages compared to SiR-DPA.

Third, we evaluate the storage overhead for dynamic public auditability, as well as communication overheads for auditing multiple replicas simultaneously. Although the total number of authenticators stayed the same, now there is only one MHT (although with more levels) as opposed to c MHTs in SiR-DPA. We can infer from Fig. 7-11 that the extra storage cost is reduced by a significant percentage when there are

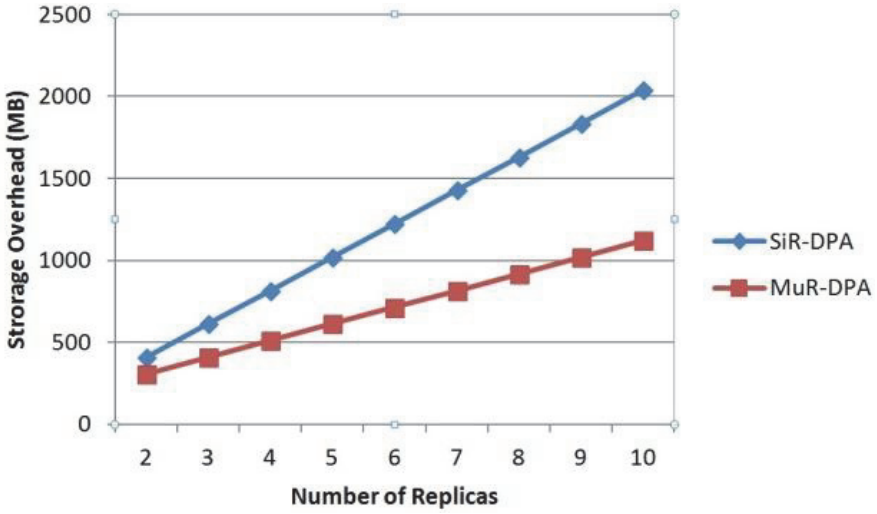
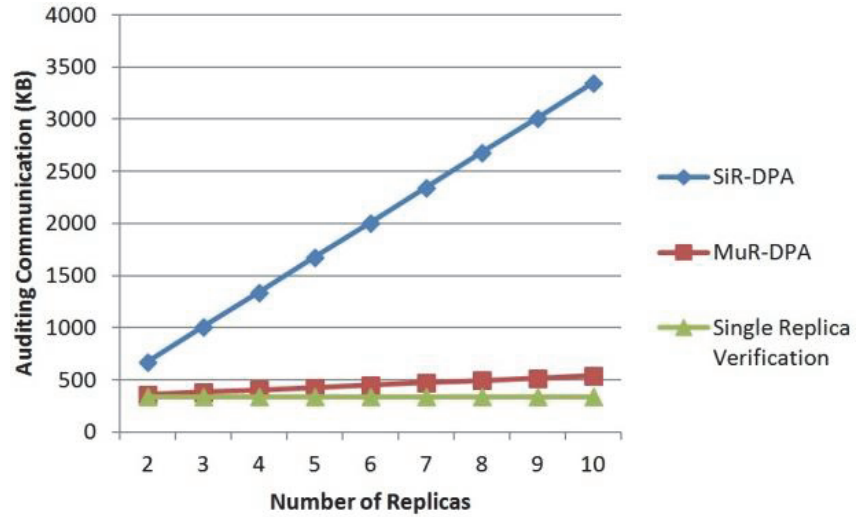


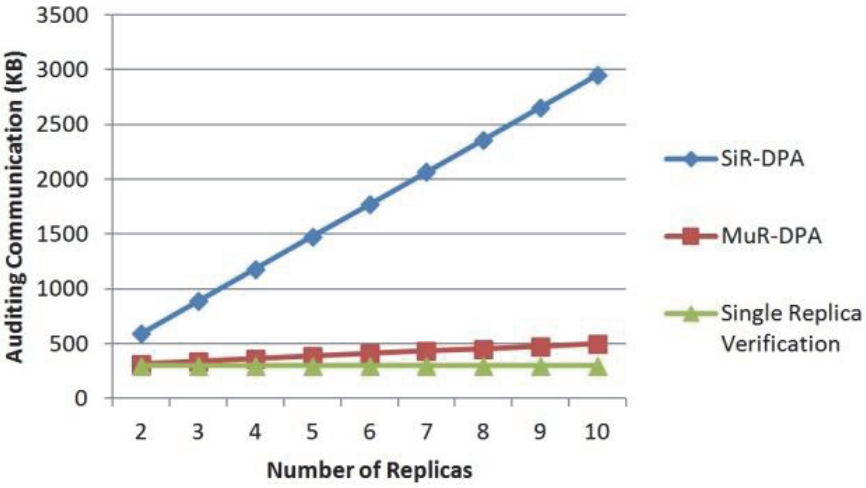
Figure 7-11 MuR-DPA: Extra storage overhead at server side for support of public auditability and data dynamics

multiple replicas stored in cloud. Communication overheads for simultaneously verifying multiple replicas are depicted in Fig 7-12. We can see that the with increases in the number of replicas that a server stores, the MuR-DPA scheme seems to outperform SiR-DPA more significantly in terms of communication overheads. We also note that with the growth of number of replicas, the communication overheads for verifying all replicas with the MuR-DPA scheme is comparable to verifying a single replica, while the overhead of SiR-DPA grows at a much faster pace. For example, when $s = 1, c = 5$, verifying all 5 replicas with MuR-DPA takes 26.8% more communication than verifying only 1 replica, while this percentage for SiR-DPA is 398.8%. Therefore, the MuR-DPA scheme is not only useful for verification of dynamic data, but also seems to scale much better when subjected to multiple replica updates.

We also tested the communication cost for one replica, under a different *s* value. As analysed in section 5, our scheme will constantly incur more communication overheads because of the extended RSTs. However, as can be seen from Fig. 7-13, the extra communication overheads are small and can be considered negligible. Even for an exaggerated case where $s = 1{,}000$ and $c = 8$, the extra communication for verification of one replica in MuR-DPA scheme is only 15.3% compared to the



(a)



(b)

Figure 7-12 MuR-DPA: Total communication overhead for auditing of all replicas.

Total communication overhead for public auditing of all replicas when (a) $s = 1$ and (b) $s = 10$.

SiR-DPA scheme. For a more common choice of 4 replicas and $s = 10$, this percentage is only 8.1%. Given that the MuR-DPA scheme has much less communication costs for verification of all replicas at once as well as verification of updates, it is always an advantageous trade-off.

From these analyses and experimental results, we can see that the MuR-DPA

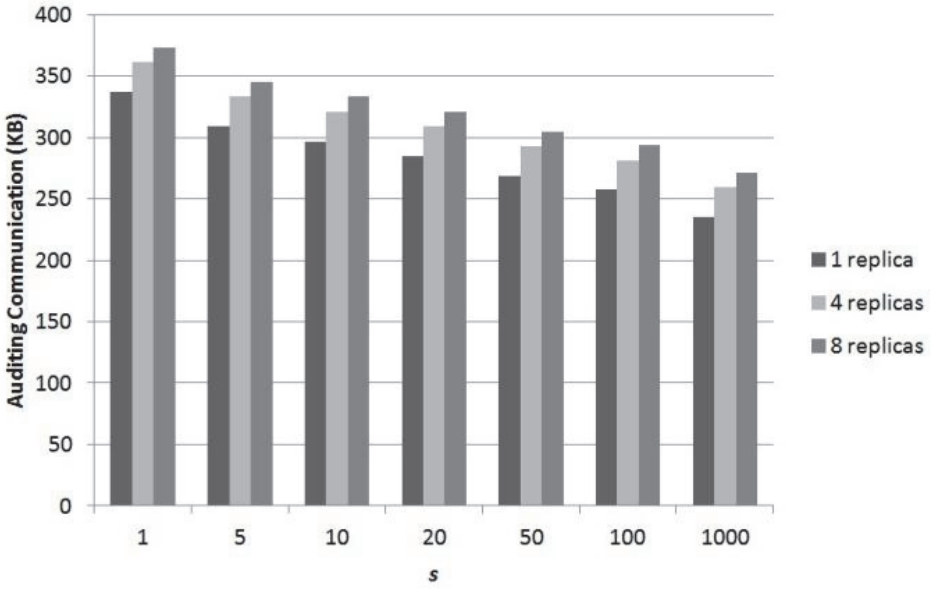


Figure 7-13 MuR-DPA: Communication for auditing of 1 chosen replica for a dataset with 1, 4 and 8 total replicas with different *s* value.

scheme has a significant advantage in auditing cloud storage with multiple replicas. The performance of public auditing schemes are not affected by the contents of data. Therefore, size of file blocks, *s* value and the number of replicas are main impact factors for the overall performance. As our experiments are based on these metrics, we believe the experimental results demonstrated here can accurately present the advantage our scheme has when deployed in practice.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

In this thesis, we have analysed the research problems of public data auditing in the cloud and big data, and we proposed a framework to address the security and efficiency problems in public auditing of dynamic big data in the cloud. Within the framework, we have developed, tested and published a series of security schemes and algorithms for secure and efficient public auditing of dynamic big data storage on the cloud. Specifically, our work focused on the following aspects: cloud internal authenticated key exchange, authorisation on third-party auditor, fine-grained update support, index verification, and efficient multi-replica public auditing of dynamic data. To the best of our knowledge, this thesis is the first sustained work to systematically analyse and address this research problem. Experimental results and analyses show that our research presented in this thesis is suitable for auditing dynamic big data storage on the cloud and they represent significant improvements in terms of both efficiency and security.

8.2 Future Work

8.2.1 Aspects for Measurements and Improvements

Aspects and examples of future work discussed in this section are briefly summarised in table 8-1. As can be seen from all the discussions above, the topic of integrity verification of big data in cloud computing is a flourishing area that is

Auditing for streaming data	**Data auditing within distributed data processing**	**Auditing of shared data**
Pre-processing	Data distribution	Levels of privileges
Application-specific data evaluation and selection	Replication strategy	Data consolidation

Table 8-1 Future Work.

attracting more and more research interest and there is still a lot of research which is currently being conducted in this area. Cloud and big data are fast-developing topics. Therefore, even though existing research has already achieved some amazing goals, we are confident that integrity verification mechanisms will continue to evolve along with the development of the cloud and big data applications to meet emerging new requirements and address new security challenges. For future developments, the following aspects are particularly interesting to look at.

Efficiency: Due to high efficiency demands in big data processing overall, efficiency is one of the most important factors in designing new techniques related to big data and cloud. In integrity verification/data auditing, the main costs can come from many aspects, including storage, computation and communication, and they can all affect the total cost-efficiency due to the pay-as-you-go model in cloud computing. We now analyse these three aspects one by one for a scheme with public auditability and support of full dynamic verifiable data updates.

a) Communication and storage: These two are the main efficiency concerns of public auditing schemes. One of the most challenging problems is that due to usage of ADS, the size of proofs depends logarithmically on the total size of the dataset, which constitutes the main communication overhead for verification of updates. Similarly, the authenticators take extra storage overhead at the server side, which also grows with the growth of the total size of datasets. Although there are works for

their optimisations, the ideal case is that the proof size and storage overhead remains constant. To the best of our knowledge, these desirable properties have never been achieved by any dynamic public auditing scheme.

b) Computation time: it is not the primary concern but it is important. The computation time for proof generation can be considered negligible in most cases, but the pre-processing time can sometimes be considerable for incremental datasets.

Security: Security is always a problem between spear and shield; that is, attack and defense. Although the current formalisations and security model seems very rigorous and potent, new exploits can always develop, especially with dynamic data streams and varying user groups. Finding the security holes and fixing them can be a long-lasting game. The security focus of existing work can be summarised in terms of different adversaries: dishonest cloud servers (Erway et al., 2009) (Liu et al., 2014c), malicious TPA (Liu et al., 2014b), other malicious users (Wang et al., 2013b), and other general-sense attackers (Liu et al., 2013b, Liu et al., 2013c). With the proposed authentication mechanisms in (Erway et al., 2009) and (Liu et al., 2014c), exploits from dishonest servers can be effectively detected in data updates. Based on existing research, a most attractive future research topic will be letting the TPA get minimal information on client data during auditing. There may also be big potential in addressing security threats from other malicious users. Multi-tenancy is one of the cloud's main characteristics, and there is currently not much work focusing on investigating this area.

Scalability/elasticity: As the cloud is a parallel distributed computing system in nature, scalability is one of the key factors as well. Programming models for parallel and distributed systems, such as MapReduce, are attracting attention from a great number of cloud computing researchers. Some of the latest work in integrity verification is already considering how to work well with MapReduce for better parallel processing (Zhu et al., 2012). On the other hand, elasticity is one of a biggest reasons why big companies are moving their business, especially service-related businesses, to the cloud (2012). User demands vary all the time, and

it would be a waste of money to purchase hardware that can handle the demands at peak times. The advent of the cloud solved this problem -- cloud allows their clients to deploy their applications on a highly elastic platform whose capabilities can be scaled up and down on-the-fly, and the cost is based solely on usage. Therefore, an integrity verification mechanism that has the same level of scalability and elasticity will be highly resourceful for big data applications in cloud environments.

8.2.2 Future Research Problems

Auditing for streaming data. Streaming data is one of the most important types of dynamic big data. Examples of streaming data including: 1) sensor data from the gathering of geographical data, temperature, humidity, etc.); 2) image data, e.g., satellites, video surveillance, etc. 3) Internet data such as video streams and social networks. A great proportion of this kind of data needs to be stored or archived for future use or further analyses. As the cloud is now the backbone for storing and processing of such data, it is essential to maintain the auditability of streaming data for cloud users (data owners) to audit their data. To date, not much work has been done in this area.

Streaming data are dynamic and real-time in nature. Like other research problems related to streaming data, the main problem in maintaining auditability of streaming data is to perform data processing on-the-fly. In other words, time complexity becomes the biggest concern here. To address this concern, future research work needs to focus on the following aspects.

1) In public auditing, pre-processing is very time-consuming due to the expensive pairing and exponentiation operations. As new data are constantly being produced in data streams, the time-consuming pre-processing contradicts the aim of efficient on-the-fly processing of streaming data. Therefore, aiming at developing a solution that can efficiently provide public auditability for streaming data while maintaining security is essential.

2) Not all the data are of the same importance. Therefore, it will be important to develop an application-specific data evaluation and selection strategy for public auditing of critical data from streaming cloud data. Through this approach, the user will be confident in terms of the integrity of the streaming data, or specifically, the useful knowledge in the coming data streams.

Data auditing within distributed data processing. The problems discussed above did not take into account the internal data storage strategy. For improved efficiency and scalability, big data applications in large-scale data centre clouds are always processed in a parallel fashion, which is achieved by distributed programming models such as MapReduce. Therefore, cloud systems always employ distributed file systems such as the Hadoop Distributed File System (HDFS) for data management and storage. In such file systems, data are at first globally partitioned for optimal performance in terms of total throughput, latency and efficiency in data processing. Then, these partitioned data are indexed and distributed to store in storage nodes. This index is stored in the name node, and the storage nodes are located in different storage servers and/or data centres. The research challenges in this area are mainly in data locality and resource availability.

First, we look at data distribution strategy. In public auditing schemes, data are also segmented into blocks and each block is accompanied by an authenticator used for auditing. However, this segmentation is only logical with another index, and it is separated from the data partitioning. Current schemes do not take into account data locality, i.e., the physical distribution of data in distributed file systems. For example, a logical block may contain data from several different storage servers. This will lead to excessive communication overheads and disk read/writes.

As mentioned earlier, a replication strategy is essential for cloud data located in separated disk blocks. For availability, replicas are located in different physical locations. This has not been considered in existing multi-replica auditing schemes, which could potentially lead to many problems. For example, when there is only a need to audit one replica, choosing replicas stored on the same server (or the least

number of servers possible) will significantly lower the total communication overheads.

According to the analysis, it will be good to develop a public auditing scheme with optimised mechanisms (e.g. block segmentation, replica selection) for user verification of data stored in distributed file systems. As different applications require different data partitioning strategies, our solution will also depend on a specific big data application and its partitioning strategy in the distributed file system.

Auditing of shared data. In public clouds, data from different users are consolidated in the same cloud service provider. Data ownership is a big problem. The problem can be mainly analysed in terms of two aspects: 1) From the data users' perspective, different users will have different levels of privileges to a certain shared data pool, and the users' identities and privileges may change from time to time; this problem needs to be carefully addressed. 2) Some data to be shared, such as medical records or police records, needs to be carefully controlled. These data are only allowed to be shared or audited by certain parties. For example, datasets to be shared by different parties may not be appropriate to be audited by the same auditor, unless they figure out useful logical connections between these datasets.

These problems have only just begun to be studied. The aim of our future work is going to be twofold.

(1) For privately shared data, the aim is at efficient and secure auditing of the dynamic variations of data users' identities and privileges while maintaining different levels of auditability of cloud data. To achieve this, efficient key management is required as different users and/or user groups will use different sets of public/private key pairs.

(2) For publicly shared data, the aim is at developing not only mechanisms for effective user and auditor management, but also strategies for controlling data sharing to maintain secure auditability of shared data. This may be achieved by analysis of

relationships between data, data users and auditors.

Bibliography

Australia Telescope, Parkes Observatory [Online]. Available: http://www.parkes.atnf.csiro.au/ [Accessed 20 December, 2014].

Crypto++ Benchmarks. [Online]. Available: http://www.cryptopp.com/benchmarks.html [Accessed 20 December, 2014].

Eucalyptus Open Source Cloud Platform [Online]. Available: http://www.eucalyptus.com/ [Accessed 20 December, 2014].

The Four V's of Big Data [Online]. Available: IBM Big Data and Analytics Hub, http://www.ibmbigdatahub.com/infographic/four-vs-big-data [Accessed 20 December, 2014].

Hadoop MapReduce [Online]. Available: http://hadoop.apache.org [Accessed 20 December, 2014].

KVM Hypervisor [Online]. Available: www.linux-kvm.org/ [Accessed 20 December, 2014].

MIRACL Cryptography Library [Online]. Available: http://certivox.com/index.php/solutions/miracl-crypto-sdk/ [Accessed 20 December, 2014].

OpenStack Open Source Cloud Software [Online]. Available: http://openstack.org/ [Accessed 20 December, 2014].

2012. Available: http://aws.amazon.com/apac/awssummit-au/ [Accessed 20 December, 2014].

2013. Department of Finance and Deregulation, Australian Government, Big Data Strategy – Issues Paper. Available: http://agimo.gov.au/files/2013/03/Big-Data-Strategy-Issues-Paper1.pdf [Accessed 5 June, 2014].

AGRAWAL, D., DAS, S. & ABBADI, A. E. Year. Big data and cloud computing: current state and future opportunities. *In:* Proceedings of the 14th International Conference on Extending Database Technology (EDBT/ICDT '11), 2011 Uppsala, Sweden. 530-533.

ARMBRUST, M., FOX, A., GRIFFITH, R., JOSEPH, A. D., KATZ, R., KONWINSKI, A., LEE, G., PATTERSON, D., RABKIN, A., STOICA, I. & ZAHARIA, M. 2010. A View of Cloud Computing. *Communications of the ACM,* 53**,** 50-58.

ARMBRUST, M., FOX, A., GRIFFITH, R., JOSEPH, A. D., KATZ, R. H., KONWINSKI, A., LEE, G., PATTERSON, D. A., RABKIN, A., STOICA, I. & ZAHARIA, M. 2009. Above the Clouds: A Berkeley View of Cloud Computing. *Technical Report No. UCB/EECS-2009-28, University of California at Berkeley.*

ATENIESE, G., BURNS, R., CURTMOLA, R., HERRING, J., KHAN, O., KISSNER, L., PETERSON, Z. & SONG, D. 2011. Remote Data Checking Using Provable Data Possession. *ACM Transactions on Information and System Security,* 14**,** Article 12.

ATENIESE, G., JOHNS, R. B., CURTMOLA, R., HERRING, J., KISSNER, L., PETERSON, Z. & SONG, D. Year. Provable Data Possession at Untrusted Stores. *In:* Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS '07), 2007. 598-609

ATENIESE, G., KAMARA, S. & KATZ, J. Year. Proofs of Storage from Homomorphic Identification Protocols. *In:* Proceedings of the 15th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT '09), 2009 Tokyo, Japan. 319 - 333.

ATENIESE, G., PIETRO, R. D., MANCINI, L. V. & TSUDIK, G. Year. Scalable and Efficient Provable Data Possession. *In:* Proceedings of the 4th International Conference on Security and Privacy in Communication Netowrks (SecureComm '08), 2008 İstanbul, Turkey. 1-10.

BARSOUM, A. F. & HASAN, M. A. 2011. On Verifying Dynamic Multiple Data Copies over Cloud Servers. *IACR Cryptology ePrint Archive, Report 2011/447*.

BARSOUM, A. F. & HASAN, M. A. Year. Integrity Verification of Multiple Data Copies over Untrusted Cloud Servers. *In:* Proceedings of the12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID '12), 2012 Ottawa, Canada. 829-834.

BELLARE, M., BOLDYREVA, A. & STADDON, J. Year. Randomness Re-use in Multi-recipient Encryption Schemeas *In:* Proceedings of the 6th International Workshop on Theory and Practice in Public Key Cryptography (PKC '03), 2003 Miami, USA. Springer-Verlag

BONEH, D., SHACHAM, H. & LYNN, B. 2004. Short Signatures from the Weil Pairing. *Journal of Cryptology,* 17**,** 297-319.

BRESSON, E., CHEVASSUT, O. & POINTCHEVAL, D. Year. Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions. *In:* Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology (EUROCRYPT '02), 2002 Amsterdam, Holland.

BUYYA, R., YEO, C. S., VENUGOPAL, S., BROBERG, J. & BRANDIC, I. 2009. Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility. *Future Generation Computer Systems,* 25**,** 599-616.

CAMENISCH, J., LEHMANN, A., NEVEN, G. & RIAL, A. Year. Privacy-Preserving Auditing for Attribute-Based Credentials. *In:* Proceedings of the 19th European Symposium on Research in Computer Security (ESORICS '14), 2014 Wroclaw, Poland. 109-127.

CANETTI, R. & KRAWCZYK, H. Year. Security Analysis of IKE's Signature-Based Key-Exchange Protocol. *In:* Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology (CRYPTO '02), 2002 Santa Barbara, USA. 143-161.

CAO, N., YANG, Z., WANG, C., REN, K. & LOU, W. Year. Privacy-Preserving Query over Encrypted Graph-Structured Data in Cloud Computing. *In:* Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS '11), 2011. 393 - 402.

CASH, D., KÜPÇÜ, A. & WICHS, D. Year. Dynamic Proofs of Retrievability via Oblivious RAM. *In:* Proceedings of the 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT '13), 2013 Athens, Greece. 279-295.

CHAUDHURI, S. Year. What next?: a half-dozen data management research goals for big data and the cloud. *In:* Proceedings of the 31st symposium on Principles of Database Systems (PODS '12), 2012 Scottsdale, Arizona, USA. 1-4.

CLIFTON, C. & TASSA, T. Year. On Syntactic Anonymity and Differential Privacy. *In:* Proceedings of the IEEE 29th International Conference on Data Engineering Workshops (ICDEW), 2013. 88-93.

CURTMOLA, R., KHAN, O., BURNS, R. C. & ATENIESE:, G. Year. MR-PDP: Multiple-Replica Provable Data Possession. . *In:* Proceedings of the 28th IEEE International Conference on Distributed Computing Systems (ICDCS '08), 2008 Beijing, China. 411-420.

CUZZOCREA, A., FORTINO, G. & RANA, O. Year. Managing Data and Processes in Cloud-Enabled Large-Scale Sensor Networks: State-of-the-Art and Future Research Directions. *In:* Proceedings of the 13th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGRID '13), 2013. 583-588.

DEELMAN, E., SINGH, G., LIVNY, M., BERRIMAN, B. & GOOD, J. Year. The Cost of Doing Science on the Cloud: the Montage Example. *In:* Proceedings of the 2008 ACM/IEEE Conference on Supercomputing (SC '08), 2008 Austin, Texas. 1–12.

DEPARTMENT OF FINANCE AND DEREGULATION, A. G. 2011. Cloud Computing Strategic Direction Paper: Opportunities and Applicability for Use by the Australian Government. Available: http://agimo.gov.au/files/2012/04/final_cloud_computing_strategy_version_1.pdf.

DEPARTMENT OF FINANCE AND DEREGULATION, A. G. 2013. Big Data Strategy – Issues Paper. Available: http://agimo.gov.au/files/2013/03/Big-Data-Strategy-Issues-Paper1.pdf.

DIFFIE, W. & HELLMAN, M. 1976. New Directions in Cryptography. *IEEE Transactions on Information Theory,* 22**,** 644 - 654.

DWORK, C. Year. Differential Privacy: A Survey of Results. *In:* Proceedings of the 5th International Conference on Theory and Applications of Models of Computation (TAMC'08), 2008. 1-19.

ERWAY, C., KÜPÇÜ, A., PAPAMANTHOU, C. & TAMASSIA, R. Year. Dynamic Provable Data Possession. *In:* Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS'09), 2009 Chicago, USA. 213-222.

ETEMAD, M. & KÜPÇÜ, A. Year. Transparent, Distributed, and Replicated Dynamic Provable Data Possession. *In:* Proceedings of the 11th International Conference on Applied Cryptography and Network Security (ACNS '13), 2013a Banff, Canada. 1-18.

ETEMAD, M. & KÜPÇÜ, A. 2013b. Transparent, Distributed, and Replicated Dynamic Provable Data Possession. *IACR Cryptology ePrint Archive, Report 2013/225***,** 1-18.

FAN, L. & XIONG, L. Year. Real-time Aggregate Monitoring with Differential Privacy. *In:* Proceedings of the 21st ACM International Conference on Information and Knowledge Management

(CIKM '12), 2012. 2169-2173.

GARG, S. K., GOPALAIYENGAR, S. K. & BUYYA, A. R. Year. Virtual Machine Provisioning Based on Analytical Performance and QoS in Cloud Computing Environments. *In:* Proceedings of the 40th International Conference on Parallel Processing (ICPP '11), 2011 Taipei, Taiwan.

GENTRY, C. Year. Fully Homomorphic Encryption Using Ideal Lattices. *In:* Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC '09), 2009 Bethesda, USA.

GROCE, A. & KATZ, J. Year. A New Framework for Efficient Password-based Authenticated Key Exchange. *In:* Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS '10), 2010 Chicago, USA. 516-525.

HANSER, C. & SLAMANIG, D. Year. Efficient Simultaneous Privately and Publicly Verifiable Robust Provable Data Possession from Elliptic Curves. *In:* Proceedings of the 10th International Conference on Security and Cryptography (SECRYPT '13), 2013 Reykjavik, Iceland. 15 - 26.

HE, Y., BARMAN, S. & NAUGHTON, J. F. Year. Preventing Equivalence Attacks in Updated, Anonymized Data. *In:* Proceedings of the 27th IEEE International Conference on Data Engineering (ICDE '11), 2011. 529-540.

HEATH, N. 2012. *Cern: Cloud Computing Joins Hunt for Origins of the Universe* [Online]. Available: http://www.techrepublic.com/blog/european-technology/cern-cloud-computing-joins-hunt-for-origins-of-the-universe/262 [Accessed 20 December, 2014].

HWANG, M.-S., LEE, C.-C. & SUN, T.-H. 2014. Data Error Locations Reported by Public Auditing in Cloud Storage Service. *Automated Software Engineering,* 21**,** 373-390.

IOSUP, A., OSTERMANN, S., YIGITBASI, M. N., PRODAN, R., FAHRINGER, T. & EPEMA, D. H. J. 2011. Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing. *IEEE Transactions on Parallel and Distributed Systems,* 22**,** 931-945.

JOHNSON, R., MOLNAR, D., SONG, D. & WAGNER, D. 2002. Homomorphic Signature Schemes. *Topics in Cryptology - CT-RSA 2002, Lecture Notes in Computer Science,* 2271**,** 244-262.

JUELS, A. & B. S. KALISKI, J. Year. PORs: Proofs of Retrievability for Large Files. *In:* Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS '07), 2007 Alexandria, USA. 584-597.

KATZ, J. & SHIN, J. S. Year. Modeling Insider Attacks on Group Key-exchange Protocols. *In:* Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS'05), 2005 Alexandria, USA. 180-189.

KATZ, J. & VAIKUNTANATHAN, V. Year. Round-optimal Password-based Authenticated Key Exchange. *In:* Proceedings of the 8th conference on Theory of cryptography (TCC'11), 2011 Providence, USA. 293-310.

KATZ, J. & YUNG, M. 2007. Scalable Protocols for Authenticated Group Key Exchange. *Journal of Cryptology,* 20**,** 85 - 113.

KAUFMAN, C., HOFFMAN, P., NIR, Y. & ERONEN, P. September 2010. Internet Key Exchange Protocol

Version 2 (IKEv2). Available: http://tools.ietf.org/html/rfc5996 [Accessed 10 April, 2012].

KAWATA, D., CEN, R. & HO, L. C. 2007. Gravitational Stability of Circumnuclear Disks in Elliptical Galaxies. *The Astrophysical Journal* 669(1)**,** 232-240.

KEAHEY, K., FIGUEIREDO, R., FORTES, J., FREEMAN, T. & TSUGAWA, M. Year. Science Clouds: Early Experiences in Cloud Computing for Scientific Applications. *In:* Proceedings of the First Workshop on Cloud Computing and its Applications (CCA '08), 2008 Chicago, USA. 1 - 6.

KUROSAWA, K. Year. Multi-recipient Public-Key Encryption with Shortened Ciphertext. *In:* Proceedings of the 5th International Workshop on Practice and Theory in Public Key Cryptosystems: Public Key Cryptography (PKC '02), 2002 Paris, France. 321–336.

KÜSTERS, R. & TUENGERTHAL, M. Year. Computational Soundness for Key Exchange Protocols with Symmetric Encryption. *In:* Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS '09), 2009 Chicago, USA. 91-100.

LI, J., OOI, B. C. & WANG, W. Year. Anonymizing Streaming Data for Privacy Protection. *In:* IEEE 24th International Conference on Data Engineering (ICDE '08), 2008.

LIU, C., BEAUGEARD, N., YANG, C., ZHANG, X. & CHEN, J. 2014a. HKE-BC: Hierarchical Key Exchange for Secure Scheduling and Auditing of Big Data in Cloud Computing. *Concurrency and Computation: Practice and Experience*.

LIU, C., CHEN, J., YANG, L. T., ZHANG, X., YANG, C., RANJAN, R. & RAMAMOHANARAO, K. 2014b. Authorized Public Auditing of Dynamic Big Data Storage on Cloud with Efficient Verifiable Fine-grained Updates. *IEEE Transactions on Parallel and Distributed Systems,* 25**,** 2234 - 2244.

LIU, C., RANJAN, R., YANG, C., ZHANG, X., WANG, L. & CHEN, J. 2014c. MuR-DPA: Top-down Levelled Multi-replica Merkle Hash Tree Based Secure Public Auditing for Dynamic Big Data Storage on Cloud. *IACR Cryptology ePrint Archive, Report 2014/391*.

LIU, C., RANJAN, R., ZHANG, X., YANG, C., GEORGAKOPOULOS, D. & CHEN, J. Year. Public Auditing for Big Data Storage in Cloud Computing - A Survey. *In:* Proceedings of the 16th IEEE International Conference on Computational Science and Engineering (CSE '13), 2013a Sydney, Australia. 1128-1135.

LIU, C., YANG, C., ZHANG, X. & CHEN, J. 2014d. External Integrity Verification for Outsourced Big Data in Cloud and IoT: A Big Picture. *Future Generation Computer Systems*.

LIU, C., ZHANG, X., CHEN, J. & YANG, C. Year. An Authenticated Key Exchange Scheme for Efficient Security-Aware Scheduling of Scientific Applications in Cloud Computing. *In:* Proceedings of the 2011 IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC '11), 12-14 Dec. 2011 2011. 372-379.

LIU, C., ZHANG, X., LIU, C., YANG, Y., RANJAN, R., GEORGAKOPOULOS, D. & CHEN, J. Year. An Iterative Hierarchical Key Exchange Scheme for Secure Scheduling of Big Data Applications in Cloud Computing. *In:* Proceedings of the 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom '13), 2013b. 9-16.

LIU, C., ZHANG, X., YANG, C. & CHEN, J. 2012. CCBKE - Session Key Negotiation for Fast and Secure Scheduling of Scientific Applications in Cloud Computing. *Future Generation Computer Systems*.

LIU, C., ZHANG, X., YANG, C. & CHEN, J. 2013c. CCBKE - Session Key Negotiation for Fast and Secure Scheduling of Scientific Applications in Cloud Computing. *Future Generation Computer Systems,* 29**,** 1300-1308.

LU, W., MIKLAU, G. & IMMERMAN, N. 2013. Auditing A Database under Retention Policies. *The VLDB Journal,* 22**,** 203-228.

MA, Y., RAO, J., HU, W., MENG, X., HAN, X., ZHANG, Y., CHAI, Y. & LIU, C. Year. An Efficient Index for Massive IOT Data in Cloud Environment. *In:* Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM '12), 2012 Hawaii, USA. 2129-2133.

MATHER, T., KUMARASWAMY, S. & LATIF, S. 2009. *Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance,* Sebastopol, O'Reilly Media.

MERKLE, R. C. Year. A Digital Signature Based on a Conventional Encryption Function. *In:* Proceedings of A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology (CRYPTO '87), 1987. 369-378.

MO, Z., ZHOU, Y. & CHEN, S. Year. A Dynamic Proof of Retrievability (PoR) Scheme with O(logn) Complexity. *In:* Proceedings of the 2012 IEEE International Conference on Communications (ICC '12), 2012. 912-916.

NAONE, E. 28 September, 2010. *What Twitter Learns from All Those Tweets* [Online]. MIT. Available: http://www.technologyreview.com/view/420968/what-twitter-learns-from-all-those-tweets/ [Accessed 20 December, 2014].

NI, J., YU, Y., MU, Y. & XIA, Q. 2014. On the Security of an Efficient Dynamic Auditing Protocol in Cloud Storage. *IEEE Transactions on Parallel and Distributed Systems,* 25**,** 2760-2761.

NURMI, D., WOLSKI, R., GRZEGORCZYK, C., OBERTELLI, G., SOMAN, S., YOUSEFF, L. & ZAGORODNOV, D. Year. The Eucalyptus Open-Source Cloud-Computing System. *In:* Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID '09), 2009. 124-131.

PUTTASWAMY, K. P. N., KRUEGEL, C. & ZHAO, B. Y. 2011. Silverline: Toward Data Confidentiality in Storage-Intensive Cloud Applications. *2nd ACM Symposium on Cloud Computing (SOCC '11).* Cascais, Portugal.

RISTENPART, T., TROMER, E., SHACHAM, H. & SAVAGE, S. Year. Hey, You, Get off of My Cloud: Exploring Information Leakage in Third-party Compute Clouds. *In:* Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS '09), 2009 Chicago, USA. 199-212.

ROY, I., SETTY, S. T. V., KILZER, A., SHMATIKOV, V. & WITCHEL, E. Year. Airavat: Security and Privacy for MapReduce. *In:* Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation (NSDI'10), 2010. 297-312.

SCHMIDT, S. E. 2012. *Security and Privacy in the AWS Cloud* [Online]. Available: http://aws.amazon.com/apac/awssummit-au/ [Accessed 20 December, 2014].

SHACHAM, H. & WATERS, B. Year. Compact Proofs of Retrievability. *In:* Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT '08), 2008. 90 - 107

SHI, E., STEFANOV, E. & PAPAMANTHOU, C. Year. Practical Dynamic Proofs of Retrievability. *In:* Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security (CCS '13), 2013. 325-336.

SRIRAMA, S. N., JAKOVITS, P. & VAINIKKO, E. 2012. Adapting Scientific Computing Problems to Clouds Using MapReduce. *Future Generation Computer Systems,* 28**,** 184-192.

STEFANOV, E., DIJK, M. V., SHI, E., FLETCHER, C., REN, L., YU, X. & DEVADAS, S. Year. Path ORAM: An Extremely Simple Oblivious RAM Protocol. *In:* Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security (CCS '13), 2013. 299-310.

STEFANOV, E. & SHI, E. Year. ObliviStore: High Performance Oblivious Cloud Storage. *In:* Proceedings of the 2013 IEEE Symposium on Security and Privacy (SP '13), 2013. 253-267.

SUBASHINI, S. & KAVITHA, V. 2010. A Survey on Security Issues in Service Delivery Models of Cloud Computing. *Journal of Network and Computer Applications,* 34**,** 1-11.

TANG, X., KENLI, L., ZENG, Z. & VEERAVALLI, B. 2011. A Novel Security-Driven Scheduling Algorithm for Precedence-Constrained Tasks in Heterogeneous Distributed Systems. *IEEE Transactions on Computers,* 60**,** 1017-1029.

VECCHIOLA, C., CALHEIROS, R. N., KARUNAMOORTHY, D. & BUYYA, R. 2012. Deadline-driven Provisioning of Resources for Scientific Applications in Hybrid Clouds with Aneka. *Future Generation Computer Systems,* 28**,** 58-65.

WANG, B., CHOW, S. S. M., LI, M. & LI, H. Year. Storing Shared Data on the Cloud via Security-Mediator. *In:* 33rd IEEE International Conference on Distributed Computing Systems (ICDCS '13), 2013a Philadelphia, USA.

WANG, B., LI, B. & LI, H. Year. Oruta: Privacy-Preserving Public Auditing for Shared Data in the Cloud. *In:* Proceedings of the 2012 IEEE 5th International Conference on Cloud Computing (CLOUD '12), 2012 Hawaii, USA. 295-302.

WANG, B., LI, B. & LI, H. Year. Public Auditing for Shared Data with Efficient User Revocation in the Cloud. *In:* Proceedings of the 32nd Annual IEEE International Conference on Computer Communications (INFOCOM'13), 2013b Turin, Italy. 2904-2912.

WANG, B., LI, B. & LI, H. 2014. Oruta: Privacy-Preserving Public Auditing for Shared Data in the Cloud. *IEEE Transactions on Cloud Computing,* 2**,** 43-56.

WANG, C., CHOW, S. M., WANG, Q., REN, K. & LOU, W. 2013c. Privacy-Preserving Public Auditing for Secure Cloud Storage. *IEEE Transactions on Computers,* 62**,** 362-375.

WANG, C., WANG, Q., REN, K. & LOU, W. Year. Privacy-Preserving Public Auditing for Data Storage

Security in Cloud Computing. *In:* Proceedings of the 29th Annual IEEE International Conference on Computer Communications (INFOCOM'10), 2010 San Diego, USA. 1 - 9.

WANG, H. & ZHANG, Y. 2014. On the Knowledge Soundness of a Cooperative Provable Data Possession Scheme in Multicloud Storage. *IEEE Transactions on Parallel and Distributed Systems,* 25**,** 264-267.

WANG, L. & FU, C. 2010. Research Advances in Modern Cyberinfrastructure. *New Generation Computing,* 28**,** 111-112.

WANG, L., KUNZE, M., TAO, J. & LASZEWSKI, G. V. 2011a. Towards Building A Cloud for Scientific Applications. *Advances in Engineering Software,* 42**,** 714-722.

WANG, L., TAO, J., KUNZE, M., CASTELLANOS, A. C., KRAMER, D. & KARL, W. Year. Scientific Cloud Computing: Early Definition and Experience. *In:* Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications (HPCC '08) 2008 Dalian, China. 825 - 830.

WANG, Q., WANG, C., REN, K., LOU, W. & LI, J. 2011b. Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing. *IEEE Transactions on Parallel and Distributed Systems,* 22**,** 847 - 859.

WATERS, B. R., BALFANZ, D., DURFEE, G. & SMETTERS, D. K. Year. Building an Encrypted and Searchable Audit Log. *In:* Proceedings of the 11th Annual Network and Distributed System Security Symposium (NDSS '04), 2004 San Diego, USA.

WILLIAMS, P., SION, R. & TOMESCU, A. Year. PrivateFS: A Parallel Oblivious File System. *In:* Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS '12), 2012. 977-988.

WONG, W. K., CHEUNG, D. W.-L., KAO, B. & MAMOULIS, N. Year. Secure kNN Computation on Encrypted Databases. *In:* Proceedings of the 35th SIGMOD international conference on Management of data (SIGMOD '09), 2009 Providence, USA. 139-152.

WORKU, S. G., XU, C. & ZHAO, J. 2014. Cloud Data Auditing with Designated Verifier. *Frontiers of Computer Science,* 8**,** 503-512.

WU, X., ZHU, X., WU, G.-Q. & DING, W. 2014. Data Mining with Big Data. *IEEE Transactions on Knowledge and Data Engineering,* 26**,** 97-107.

YANG, C., LIU, C., ZHANG, X., NEPAL, S. & CHEN, J. 2013a. A Time Efficient Approach for Detecting Errors in Big Sensor Data on Cloud. *IEEE Transactions on Parallel and Distributed Systems*.

YANG, C., ZHANG, X., ZHONG, C., LIU, C., PEI, J., RAMAMOHANARAO, K. & CHEN, J. 2013b. A Spatiotemporal Compression based Approach for Efficient Big Data Processing on Cloud. *Journal of Computer and System Sciences (JCSS)*.

YANG, K. & JIA, X. 2012. Data Storage Auditing Service in Cloud Computing: Challenges, Methods and Opportunities. *World Wide Web,* 15**,** 409-428.

YANG, K. & JIA, X. 2013. An Efficient and Secure Dynamic Auditing Protocol for Data Storage in Cloud

Computing. *IEEE Transactions on Parallel and Distributed Systems,* 24**,** 1717-1726.

YAO, J., CHEN, S., NEPAL, S., LEVY, D. & ZIC, J. Year. TrustStore: Making Amazon S3 Trustworthy with Services Composition. *In:* Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGRID '10), 2010 Melbourne, Australia. 600-605.

YOUNG CHOON LEE, A. Y. Z. 2011. Energy Conscious Scheduling for Distributed Computing Systems under Different Operating Conditions. *IEEE Transactions on Parallel and Distributed Systems,* 22**,** 1374-1381.

YU, Y., MU, Y., NI, J., DENG, J. & HUANG, K. Year. Identity Privacy-Preserving Public Auditing with Dynamic Group for Secure Mobile Cloud Storage. *In:* Proceedings of the 8th International Conference on Network and System Security (NSS 2014), 2014a Xi'an, China. 28-40.

YU, Y., NIU, L., YANG, G., MU, Y. & SUSILO, W. 2014b. On the Security of Auditing Mechanisms for Secure Cloud Storage. *Future Generation Computer Systems,* 30**,** 127-132.

YUAN, D., YANG, Y., LIU, X. & CHEN, J. 2010. A Data Placement Strategy in Scientific Cloud Workflows. *Future Generation Computer Systems,* 26**,** 1200-1214.

YUAN, D., YANG, Y., LIU, X. & CHEN, J. 2011. On-demand Minimum Cost Benchmarking for Intermediate Dataset Storage in Scientific Cloud Workflow Systems. *Journal of Parallel and Distributed Computing,* 71**,** 316-332.

YUAN, J. & YU, S. Year. Secure and Constant Cost Public Cloud Storage Auditing with Deduplication. *In:* Proceedings of the First IEEE Conference on Communications and Network Security (CNS '13), 2013 Washington D.C., USA. 145-153.

ZHANG, K., ZHOU, X., CHEN, Y., WANG, X. & RUAN, Y. Year. Sedic: Privacy-aware Data Intensive Computing on Hybrid Clouds. *In:* Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS '11), 2011. 515-526.

ZHANG, X., LIU, C., NEPAL, S., PANLEY, S. & CHEN, J. 2012. A Privacy Leakage Upper-bound Constraint based Approach for Cost-effective Privacy Preserving of Intermediate Datasets in Cloud. *IEEE Transactions on Parallel and Distributed Systems*.

ZHANG, X., LIU, C., NEPAL, S., PANLEY, S. & CHEN, J. 2013a. A Privacy Leakage Upper-bound Constraint based Approach for Cost-effective Privacy Preserving of Intermediate Datasets in Cloud. *IEEE Transactions on Parallel and Distributed Systems,* 24**,** 1192-1202.

ZHANG, X., LIU, C., NEPAL, S., YANG, C., DOU, W. & CHEN, J. 2013b. A Hybrid Approach for Scalable Sub-Tree Anonymization over Big Data using MapReduce on Cloud. *Journal of Computer and System Sciences (JCSS), in press*.

ZHANG, X., YANG, L. T., LIU, C. & CHEN, J. 2013c. A Scalable Two-Phase Top-Down Specialization Approach for Data Anonymization using MapReduce on Cloud. *IEEE Transactions on Parallel and Distributed Systems*.

ZHANG, Y. & BLANTON, M. 2012. Efficient Dynamic Provable Possession of Remote Data via Update Trees. *IACR Cryptology ePrint Archive, Report 2012/291*.

ZHANG, Y. & BLANTON, M. Year. Efficient dynamic provable possession of remote data via balanced update trees. *In:* Proceedings of the 8th ACM Symposium on Information, Computer and Communications Security (ASIACCS '13), 2013 Hangzhou, China. 183-194.

ZHAO, J. & GU, D. 2010. Provably Secure Authenticated Key Exchange Protocol under the CDH Assumption. *Journal of Systems and Software,* 83**,** 2297-2304.

ZHAO, J., WANG, L., TAO, J., CHEN, J., SUN, W., RANJAN, R., KOLODZIEJ, J., STREIT, A. & GEORGAKOPOULOS, D. 2014. A Security Framework in G-Hadoop for Big Data Computing Across Distributed Cloud Data Centres. *Journal of Computer and System Sciences,* 80**,** 994-1007.

ZHOU, Z. & HUANG, D. Year. An Optimal Key Distribution Scheme for Secure Multicast Group Communication. *In:* Proceedings of the 2010 IEEE Conference on Computer Communications (INFOCOM '10), 2010 San Diego, USA.

ZHU, Y., HU, H., AHN, G.-J. & YU, M. 2012. Cooperative Provable Data Possession for Integrity Verification in Multi-Cloud Storage. *IEEE Transactions on Parallel and Distributed Systems,* 23**,** 2231-2244.

ZISSIS, D. & LEKKAS, D. 2011. Addressing Cloud Computing Security Issues. *Future Generation Computer Systems,* 28**,** 583-592.

Appendix A

Acronyms

For the convenience of readers, acronyms used in this thesis are listed in alphabetical order in this section.

AAI	Auxiliary Authentication Information
ADS	Authenticated Data Structure
AKE	Authenticated Key Exchange
BLS	Boneh-Lynn-Shacham signature scheme
CA	Certificate Authority
CSS	Cloud Storage Server
HLA	Homomorphic Linear Authenticator
HVT	Homomorphic Verifiable Tag
MAC	Message Authentication Code
MHT	Merkle Hash Tree
PDP	Provable Data Possession
PKI	Public-Key Infrastructure
POR	Proof of Retreivability

RASL	Rank-based Authenticated Skip List
RMR-MHT	Rank-based Multi-Replica Merkle Hash Tree
SiR-DPA	Public auditing scheme presented in (Wang et al., 2011b)
TPA	Third-Party Auditor
WMHT	Weighted Merkle Hash Tree

Appendix B

Notation Index

For the convenience of readers, notations used in this thesis are listed in alphabetical order in this section.

$AUTH$	A message shared between the client and CSS, used for authorisation of third-party auditing
$b_{i,j}$	The *i*th block of replica $\tilde{F}_j$.
C	Cloud controller (CLC)
$Cert_i$	Certificate
$CertReq$	Certificate request
$DEC_k(M)$	Decrypt message M with session key k
(f_k, l_k, q_k, d_k)	The *k*th tuple in Ω_i where f_k is the hash value, l_k is the level of node, q_k is the rank value and d_k indicates whether this node is a left or right child node
F	Raw data file to be uploaded by the client to store in CSS

$\mathcal{F}$	Segmented file F, in the form of $\{m_{ij}\}$. Sometimes we will use $\mathcal{F}$ and F interchangeably
$\tilde{F}_j$	The jth replica of file F
$h(v), h_i$	In Chapter 6, hash values stored in node v from RMR-MHT T.
HDR_i	Header, contains security parameter indexes
ID_i	Identity information
KEY_UP_N	Temporary key used by control node N to encrypt the communications with its parent node
$KEY\_DOWN_{N,k}$	Temporary key used by control node N to encrypt the communications with its kth sub-node
l	Number of control layers in cloud
$l(v)$	The level of node v in RMR-MHT
$m_{i,j}$	Number of sub-nodes for jth control node on the ith layer
m_i	The ith file block of F. There are a total of n blocks

$N_{i,j}$	The jth control node (or *node*, for simplicity) on layer i. $N_{1,1}$ is the cloud controller (CLC). See Figure 3-5 for example
$\overline{N}_k$	Control node *N*'s sub-nodes (children nodes)
$\widetilde{N}$	Control node *N*'s parent node
n_i	Number of nodes on the ith control layer
$nonce$	One-time nonce for message freshness
o	An offset in file F, its value equals the bit length in the range from start point of *F* to the checkpoint
prf()	Pseudorandom function
PUB_N:	Node *N*'s public key for KE
$r(v)$	Rank value of node v - the maximum number of nodes in the leaf (bottom) level that can be reached from v
$r_{i,j}$	Padding message used to generate replica block $b_{i,j}$ with the original file block m_i
R	The hash value stored in the root node of T

s	Number of segments per block
$s_{\max}$	The maximum number of segments per block
S	Server instances domain
S_i	The ith server instance in S
SA	Security associations, used in negotiating cryptographic algorithms
Sig_i/ sig	Digital signature that can be verified by anyone with a public key. In KE schemes in Chapter 4, it can be verified using algorithms negotiated in SA and a public key obtained from $Cert_i$
sig_{AUTH}	A signature used for authorisation of TPA
t	File tag of file F, which can be used to uniquely identify F
T	The WMHT (in Chapter 5) or RMR-MHT (in Chapter 6) developed based on $\mathcal{F}$
T_i	Replica-sub Tree of RMR-MHT T
VID	Verifier (TPA)'s ID

Γ_i	The set of tuples $\{h, l, q, t\}$ for all intermediate nodes in the RST T_i
η	Size of each file segment
$(\lambda_k, \eta_k, \xi_k, \zeta_k)$	A tuple of variables used for verification in MuR-DPA in Chapter 6. For a successful verification, after iterative computation with Ω_i, λ will become the number of total file blocks, η will become the root value R, ξ will become the block index and ζ will become the reversed block index, i.e., the block count from right
σ	The homomorphic authenticator (verification tag)
Φ	The ordered set of authenticators for $\mathcal{F}$
Ω_i	A set of hash values (or tuples that include hash values) that are used as m_i's auxiliary authentication information (AAI)
$\{m\}_k$	Encrypt message m with session key k
$\|m\|$	Size of message m