Towards Efficient Proofs of Storage and Verifiable Outsourced Database in Cloud Computing

Jia Xu

B.Comp.(Hons.), NUS

A THESIS SUBMITTED
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN DEPARTMENT OF COMPUTER SCIENCE

NATIONAL UNIVERSITY OF SINGAPORE

May 2012

Acknowledgement

I would like to thank everyone who has helped me through my PhD study.

First of all, I express my most sincere appreciation to my PhD advisor Dr Ee-Chien Chang. Dr Chang is very kind and provide me a research environment which is full of freedom. He is greatly sensitive in capturing the essential ideas behind a complicate appearance. He is always pursuing simplest and elegant algorithms in solving a wide range of problems. His research methodology and academic personality will benefit me for a long time. I also express my deep appreciation to the thesis committee members Dr Haifeng Yu and Dr Stephanie Wehner.

I thank all of my co-authors and my lab fellows for all of great ideas, hard work, discussions and arguments. They are Chengfang Fang, CheeLiang Lim, Jie Yu, Dr Liming Lu, Dr Sourav Mukhopadhyay, Yongzheng Wu, Chunwang Zhang, Xuejiao Liu. I also thank Dr Aldar Chun-fai Chan and Dr Zachary Peterson for their helpful suggestions. I thank my friends Dr Tao Shao and Jiqin Wang, who helped me in academic or non-academic aspects.

I express my great thanks to my family—my parents who always love me unconditionally, my two sisters and my little niece. I express my most special thanks to my girl friend Zhu Chen, who gives me a lot of delighted hours and always companies me in my bright and dark time.

Thank you all very much! Without your support, this dissertation may not be possible.

Contents

Summary

Cloud computing is becoming an important topic in both industry and academic communities. While cloud computing provides many benefits, it also brings in new challenges in research, especially in information security. One of the main challenges is how to achieve a pair of apparently conflicting requirements simultaneously: *efficiency* in communication, storage and computation on both client and server sides, and *security* against outside and internal attackers. Security concerns consist of data confidentiality and data integrity.

This dissertation is devoted to *efficiently verify integrity in cloud storage and outsourced database*. The main strategy is to *devise new homomorphic cryptographic methods*.

For cloud storage, we propose three efficient methods that allow users to remotely check the integrity of their files stored in a potentially dishonest cloud storage server, without downloading their files. These three methods rely on three underlying homomorphic authentication methods, which we design with different techniques. All of these three underlying homomorphic authentication methods support linear homomorphism: Given a public key and a sequence of message-tag pairs, any third party can compute a valid authentication tag for a linear combination of these messages. Furthermore, the second and third authentication methods support an additional homomorphism: Given a public key and an authentication tag of a long message, any third party can compute a valid authentication tag for a short message, as long as the short message and the long message satisfy a predetermined predicate. We prove security properties of the proposed schemes under various cryptographic hard problem assumptions.

For outsourced database, we propose an efficient authentication method that allows users to query their database which is maintained by a potentially dishonest server, and verify the correctness and completeness of the query results returned by the server. Supported database queries include aggregate count/min/max/median query conditional on multidimensional rectangular range selection, and non-aggregate multidimensional rectangular range selection query. The proposed method relies on our newly constructed functional encryption scheme. This functional encryption scheme allows a third party, with a delegation key that is generated on the fly, to compute a designated function (the function is specified in the delegation key) value of the plaintext from the corresponding ciphertext, yet without knowing the value of the plaintext. We prove security properties of the proposed schemes under various cryptographic hard problem assumptions.

List of Tables

List of Figures

Chapter 1

Introduction

Software as a Service (SaaS), among other forms of cloud computing, is becoming a trend in industry. By outsourcing IT services (e.g. database management, backup services) to a professional service provider, users (e.g. companies, organizations or individuals) can reduce expensive operation cost to maintain IT services, and are relieved to focus on their core business.

Outsourcing computation tasks and IT services to a potentially dishonest cloud service provider, bring in many research challenges, especially in information security. Indeed, any third party cloud service provider could be considered as potentially dishonest to a prudent user. One of the main challenges is how to achieve a pair of apparently conflicting requirements simultaneously: *efficiency* in communication, storage and computation on both client and server sides, and *security* against outside and internal attackers. Security concerns consist of two main aspects among others:

- Computation confidentiality and privacy: The server is able to compute a result upon a query provided by a client, yet both query and result are in some encrypted form and hidden from both the server and outside attackers.

- Computation authentication and integrity: The computation results requested by clients and generated by the server from clients' data should be correct. Clients should be able to efficiently verify the correctness of the returned computation results. Such correctness verification should be more efficient than direct computation of these results from scratch.

In general, privacy-preserving and/or verifiable delegated computation for any polynomial time computable function can be implemented in polynomial time [GGP10, CKV10], due to Gentry's recent breakthrough in constructing fully homomorphic encryption [Gen09]. A natural and meaningful question is that: Can we design more efficient solutions than the generic solution for a smaller class of functions? As Whitfield Diffie said, "*The whole point of cloud computing is economy*" [Dif09].

Indeed, before the generic polynomial time solution appears in 2010, the research community has studied privacy preserving and/or verifiable delegation of computation for smaller class of functions for almost a decade. As one of the first few examples of outsourced IT services, outsourced database and its security [HILM02,MNT06] became a hot research topic in database and security communities, since 2002. Recently, there is growing interests in remote verification of integrity of data stored in a cloud storage server [JK07,ABC$^+$07,CX08], which is another example of secure outsourced IT services.

1.1 Our Results and Contributions

In this dissertation, we focus on only integrity aspect of delegation of two sorts of computation tasks: cloud storage and outsourced database. The goal of this dissertation is to *construct efficient and reliable delegation schemes for proofs of storage and verifiable outsourced database.* Our main strategy is to *devise efficient homomorphic cryptography methods*, which has been proved to be an effective and powerful tools for such task. This dissertation is divided into two parts. Our results and contributions can be summarized as below.

1.1.1 Part I: Proofs of Storage

Problem.

In Part I of this dissertation, we are interested in this problem: Alice, with a small and reliable storage, stores her file `F` together with some authentication information at a potentially dishonest cloud storage server Bob, who has a large storage. Later, Alice will periodically and remotely verify whether Bob indeed keeps the file `F` intact, in an efficient manner.

Results.

In Part I, we propose three methods that allow Alice to verify the integrity of her file stored in the untrusted cloud storage efficiently and reliably, without downloading her file during a verification and without keeping a local copy of the file:

- In Chapter 4, we propose a homomorphic Message Authentication Code scheme named $\mathsf{S1}$ and apply $\mathsf{S1}$ to construct a proofs of storage scheme named $\mathsf{POS1}$. The resulting proofs of storage scheme $\mathsf{POS1}$ is very efficient in communication and computation.

- In Chapter 5, we propose a homomorphic Message Authentication Code scheme named $\mathsf{S2}$ which supports two sorts of homomorphic properties, and apply $\mathsf{S2}$ to construct a proofs of storage scheme named $\mathsf{POS2}$. The constructed scheme $\mathsf{POS2}$ is very efficient in communication and storage, and is practical in computation.

- In Chapter 6, we propose a proofs of storage scheme named $\mathsf{POS3}$ that achieves similar complexity as the second scheme, using different techniques. The construction of of $\mathsf{POS3}$ is conceptually simpler than $\mathsf{POS2}$.

We prove the security properties of the above three proofs of storage methods conditional on various cryptographic hard problem assumptions (i.e. Strong Diffie-Hellman Assumption, Large Integer Factorization Assumption, assumption that secure pseudorandom function exists), under Proofs of Retrievability ($\mathcal{POR}$) security formulation given by Juels and Kaliski [JK07] or Provable Data Possession ($\mathcal{PDP}$) security formulation given by Ateniese *et al.* [ABC$^+$07]. A brief summary of complexities of these three schemes is given in Table 1.1.

Table 1.1: Complexities of the three proofs of storage schemes POS1, POS2, and POS3, proposed in Part I of this dissertation. All of these three solutions require only $\mathcal{O}(\lambda)$ communication, storage and computation cost on client's (Alice's) side, independent on the file size. More detailed complexity analysis will be given later in Chapter 4, Chapter 5, and Chapter 6, respectively.

Scheme	Server Storage Overhead	Computation (Preprocess)	Computation (Prove)	Public Key Size	Security Model
POS1	$\lvert\mathtt{F}\rvert$	$\lvert\mathtt{F}\rvert/\lambda$	$\mathcal{O}(\lambda)$	$\mathcal{O}(\lambda)$	$\mathcal{POR}$
POS2	$\lvert\mathtt{F}\rvert/m$	$\lvert\mathtt{F}\rvert/\lambda$	$\mathcal{O}(\lambda m)$	$\mathcal{O}(\lambda m)$	$\mathcal{POR}$
POS3	$\lvert\mathtt{F}\rvert/m$	$\lvert\mathtt{F}\rvert/\lambda$	$\mathcal{O}(\lambda m)$	$\mathcal{O}(\lambda)$	$\mathcal{PDP}$

Notation: λ is the security parameter, m is the size of a block in POS2 and POS3, and $\lvert\mathtt{F}\rvert$ represents the file size (after error erasure encoding).

1.1.2 Part II: Verifiable Outsourced Database

Problem.

In Part II of this dissertation, we are interested in the integrity of the query results from an outsourced database service provider. Alice has a set $\mathbf{D}$ of d-dimensional integer points. Alice chooses her private key and generates authentication tag $\mathbf{T}$ for data set $\mathbf{D}$ under her private key. Alice passes the data set $\mathbf{D}$ together with the authentication tag $\mathbf{T}$, to an untrusted service provider Bob, and removes local copies of $\mathbf{D}$ and $\mathbf{T}$. Later, Alice issues some query over $\mathbf{D}$ to Bob, and Bob should produce the query result and a proof based on $\mathbf{D}$ and $\mathbf{T}$. Alice wants to verify the integrity of the query result against the proof, using only her private key. In Part II of this dissertation, we consider aggregate query conditional on multidimensional rectangular range selection. In its basic form, a query asks for the total number of data points within a d-dimensional range.

Authenticating Aggregate Count Query.

We are concerned about the number of communication bits required and the size of the tag $\mathbf{T}$. We propose a scheme that requires $O(d^2 \log^2 N)$ communication bits per query and linear size authentication tag $\mathbf{T}$ w.r.t. the size of dataset $\mathbf{D}$, to authenticate an aggregate count query conditional on d-dimensional rectangular range selection, where N is the number of points in the dataset. The security of our scheme relies on Generalized Knowledge of Exponent Assumption proposed by Wu and Stinson [WS07].

Authenticating Aggregate Min/Max/Median Queries and Non-aggregate RangeSelect Query.

Besides counting, our scheme can be extended to support finding of the minimum, maximum or median, and usual (non-aggregate) range selection with similar complexity: $O(d^2 \log^2 N)$ communication bits per query and linear size authentication tag $\mathbf{T}$ w.r.t. the size of dataset $\mathbf{D}$.

Functional Encryption.

The low communication bandwidth is achieved due to a new functional encryption scheme, which we specially design by exploiting a property of BBG HIBE scheme [BBG05]. This functional encryption scheme allows a third party, with a delegation key that is generated on the fly, to compute a designated function value of the plaintext from the corresponding ciphertext, yet without knowing the value of the plaintext. This designated function is a two-input one-way [Gol06] function, where one input is the plaintext and the other input is secretly embedded in the delegation key. This new functional encryption scheme may have independent interests.

1.2 Organization

Except Chapter 13 at the end which concludes the whole dissertation, the rest of this dissertation consists of two parts. Part I includes Chapter 2 to Chapter 6, and is devoted to proofs of storage problem. Part II includes Chapter 7 to Chapter 12, and is devoted to the verifiable outsourced database problem.

1.2.1 Organization of Part I

In the first part, Chapter 2 introduces the background on proofs of storage problem and Chapter 3 gives the formulation. In the subsequent three chapters, we will propose three proofs of storage schemes. In Chapter 4, we propose a homomorphic MAC scheme $\mathsf{S1}$ and apply it to construct a proofs of storage scheme $\mathsf{POS1}$. We prove the security property of $\mathsf{POS1}$ under the $\mathcal{POR}$ model. In Chapter 5, we propose a homomorphic MAC scheme $\mathsf{S2}$ and apply it to construct a proofs of storage scheme $\mathsf{POS2}$. We prove the security property of $\mathsf{POS2}$ under the $\mathcal{POR}$ model. In Chapter 6, we propose the third proofs of storage scheme $\mathsf{POS3}$ and prove its security under $\mathcal{PDP}$ model.

1.2.2 Organization of Part II

The second part of this dissertation is organized as follows: Chapter 7 gives an introduction on the verifiable outsourced database problem and reviews related works. Chapter 8 gives an overview of our main scheme. Chapter 9 presents the problem formulation and security definition. The new functional encryption scheme is constructed in Chapter 10. Our main scheme for count query is described and analyzed in Chapter 11 and its extensions for min/max/median and range selection queries are given in Chapter 12. The full proof of security properties of the functional encryption scheme and the authentication scheme is in Appendix A.

Part I

Proofs of Storage:

Are our files really in the cloud?

Chapter 2

Background

Storing data in a cloud storage, for example Amazon Cloud Drive, Microsoft Skydrive, or Dropbox, is gaining popularity recently. We are considering scenarios where users may have concerns of the integrity of their data stored in the cloud storage. Such prudent users may not be simply satisfied with the cloud storage server's promise on maintaining the data integrity. Instead, they desire a technical way to verify that whether the cloud storage server is keeping his promise and following the service level agreement (SLA). That is, these users want to base their data integrity on the *incapability* of cloud storage server to break SLA without being caught. Threat to integrity of data stored in cloud is indeed realistic. It is reported that Dropbox keeps all user accounts unlocked for almost 4 hours [wir] and allows adversaries to read and modify users' data files, due to a software bug. Very recently, a similar incident occurs to Twitter [Twi]: A software bug in twitter's official client allows adversaries to access (read and modify) user accounts. Several events about massive data loss in cloud have been reported, e.g. Microsoft Sidekick [Wik11], Amazon Cloud Service [Bus, Ama], Gmail [Goo11] and Hotmail [Mic11]. There are also plenty of data loss cases that are claimed by individuals but neither confirmed nor denied officially by the cloud server, e.g. data loss cases in Dropbox [Dro11].

2.1 Problem Description

We are interested in this problem: Suppose a user Alice with a small and reliable storage, stores her file with a potentially dishonest cloud storage server Bob who has a large storage. How can Alice remotely, periodically and efficiently verifies the integrity of her file that is stored in Bob's storage?

2.1.1 Remote Integrity Verification

"Remote Integrity Verification" is a counterpart of local data integrity verification, which is a historic and well-studied problem. Existing solutions to local data integrity verification include collision resistant hash [NIS02] and message authentication code [BCK96] for adversarial errors, and cyclic redundancy check [PB61] (CRC) for random errors, among others. Unlike local data integrity verification, the verifier in remote data integrity verification does not possess (even a small portion of) the data file at the time of verification.

2.1.2 Periodical Integrity Verification

It is desired that such remote data integrity verification could be done by a practically unlimited number of times. For conditional secure remote data integrity verification methods, it is required that, with respect to any fixed polynomial $poly(\cdot)$, for any security parameter λ, the verification method can reliably run for at least $poly(\lambda)$ times, where the polynomial $poly(\cdot)$ is fixed before the value of λ is chosen. The implication of this requirement is that, the communication cost per each remote verification should be as small as possible.

2.1.3 Efficient Integrity Verification

We concern about the efficiency of such verification methods in communication[1], storage, and computation, on both client side (i.e. Alice's side) and server side (i.e. Bob's side). Furthermore, efficiency (computation and storage and communication)

[1]Alice's communication cost is always equal to Bob's communication cost.

on client side takes priority. Ideally, all of computation cost, storage overhead and communication cost on client side should be $\mathcal{O}(\lambda)$, independent on the file size, where λ is the security parameter. Indeed, all of three proofs of storage schemes $\mathsf{POS1}$, $\mathsf{POS2}$ and $\mathsf{POS3}$ satisfy this efficiency requirement.

2.1.4 Simple but Undesirable Methods

The above requirements exclude the following straightforward approaches, which are either efficient or robust, but not both.

Keeping a hash value.

Alice keeps a hash value of her file in local storage. In a verification, Alice asks Bob to compute the hash value over her file stored in Bob's storage. The hash value returned by Bob will be compared to the one kept in Alice's local storage. This method can support verification for only one time, since dishonest Bob may cache the hash value and delete Alice's file. This method can be generalized in this way: Alice precomputes a number of t keyed-hash values under t different random hash keys, and in each out of t verifications, Alice consumes one random key out of t hash keys.

Downloading the file.

Alice keeps a hash value of her file in local storage. In each verification, Alice downloads her file from Bob and verifies file integrity locally. This method suffers from a large communication cost per verification.

Keeping a local copy of the file.

Alice persistently keeps a copy of her file in local storage. In each verification, Alice sends a random key to Bob and asks Bob to compute a key-ed hash value over her file. Alice computes the key-ed hash value over her local copy of the file w.r.t. the same key and compare the result with the hash value returned from Bob. This method suffers from large storage cost on client side.

2.2 Two Early Approaches

Now we brief two early approaches for proofs of storage: One based on RSA method, and the other based on Message Authentication Code. These two approaches have influence in many subsequent solutions to proofs of storage, including Ateniese *et al.* [ABC+07], Chang and Xu [CX08], Shacham and Waters [SW08a], and all of three solutions proposed in the Part I of this dissertation.

2.2.1 RSA based method

This scheme appears in [DQS03, FB06]: Alice chooses two primes p and q, and compute a RSA modulus $N = pq$, where N is made public and p, q are kept secretly. Alice also chooses a random integer $g < N$ which is co-prime to N. Suppose Alice wants to backup a file $\mathtt{F}$ to Bob's storage. Alice treats $\mathtt{F}$ as a single large integer, and computes $\big(\mathtt{F} \mod (p-1)(q-1)\big)$ and $\pi = \big(g^{\mathtt{F} \mod (p-1)(q-1)} \mod N\big)$. At the end of the setup between Alice and Bob, Alice has only (N, g, π) in her storage and Bob has $(N, \mathtt{F})$ in his storage. In each verification, Alice chooses a random number d and sends $\big(g^d \mod N\big)$ to Bob. Bob should compute and return the value $\psi = \big(g^d\big)^{\mathtt{F}} = g^{d\mathtt{F}} \mod N$. After receiving the response ψ, Alice checks whether ψ is equal to $\big(\pi^d \mod N\big)$.

2.2.2 MAC based method

This scheme appears in Naor and Rothblum [NR05, NR09]: Alice chooses a Message Authentication Code scheme (for example, HMAC [BCK96]) and generates a random private key k for the MAC scheme. To backup her file $\mathtt{F}$ to Bob, Alice encodes file $\mathtt{F}$ using some error erasure code (e.g. Reed-Solomon code [RS60]) to obtain file blocks $\mathtt{F}_i, i = 0, 1, 2, \ldots, n-1$, such that only a fraction of blocks $\mathtt{F}_i$'s can recover the original file $\mathtt{F}$ using the decoding algorithm of the error erasure code. For each i, Alice produces a MAC value σ_i for the combination of block $\mathtt{F}_i$ and index i, under private key k. Then Alice interacts with Bob to carry out a setup. At the end of setup, Alice has only the private key k and file size n (in term of number of blocks) in her storage and Bob has all blocks and MAC values $\{(i, \mathtt{F}_i, \sigma_i) : i = 0, 1, 2, \ldots, n-1\}$. In each verification, Alice sends a random subset $C \subset \{0, 1, 2, \ldots, n-1\}$ to Bob. Bob is supposed to return $\{(i, \mathtt{F}_i, \sigma_i) : i \in C\}$. After receiving the response, Alice will check each tuple $(i, \mathtt{F}_i, \sigma_i)$ using the MAC scheme, with private key k.

2.2.3 Advantages and Disadvantages

The above two methods can reliably verify integrity of Alice's data stored in Bob's storage for practically unlimited times. The RSA based scheme is very efficient in communication and storage: $\mathcal{O}(\lambda)$ communication cost and $\mathcal{O}(\lambda)$ storage overhead on both client (Alice) and server (Bob) side, where λ is the bit-length of the RSA modulus N. The MAC based scheme requires $\mathcal{O}(\lambda)$ storage overhead on client side and accesses only a portion of the file of interest per verification, where λ is the bit-length of the private key k.

However, the RSA based scheme has to access every single bit of the file of interest during each verification, and the MAC based scheme has large communication cost and large storage overhead on server side. Subsequent solutions improve these two approaches in efficiency from various aspects.

Particularly, the first part of this dissertation will propose three solutions to the problem described above. In all of these three methods, communication cost, storage overhead and computation cost on client side are in $\mathcal{O}(\lambda)$. In each verification, only a small portion of files are accessed on the server side, independent on file size. Furthermore, in the second and third methods, the storage overhead is just a fraction[2] of the original file size.

2.3 Tools and Building blocks

Many constructions of proofs of storage consist of three components: (1) Chunking and Indexing; (2) Error Erasure Coding and Random sampling; (3) Homomorphic cryptography. Each of them is described as below.

[2]This fraction is a configurable system parameter.

2.3.1 Chunking and Indexing

An efficient proofs of storage scheme only requires a prover to access a sublinear fraction of data file for one verification request. Thus the data file has to be broken into many small units and each unit gets a unique identifier, otherwise the verifier has no way to tell the prover to access which part of the file and to ensure the prover's response is indeed computed from those selected data units.

We call such small unit as *block*. Typically, the identifier of a data block in a data file $\mathtt{F}$ consists of two parts: (1) a sequence number to distinguish it from other blocks in the same file; (2) a unique file identifier for $\mathtt{F}$ to distinguish it from other files. A file $\mathtt{F}$ with file identifier id and consisting of n blocks is represented as $(\mathtt{F}_0, \ldots, \mathtt{F}_{n-1})$, where for each $i \in [0, n-1]$, the i-th block $\mathtt{F}_i$ has the unique identifier $\mathsf{id}\|i$ across all data blocks and all files. Here the binary operator $\|$ denotes the unambiguous string concatenation which allows unique unambiguous decomposition. Readers will find that in all proofs of storage schemes proposed later in this dissertation, the authentication tag for the i-th block $\mathtt{F}_i$ in file $\mathtt{F}$, which has identifier id, involves $\mathsf{id}\|i$. Thus, the verifier can distinguish different files and different blocks during a verification. Notice that for proofs of storage schemes [WWL+09, EKPT09] built on authenticated data structure (e.g. Merkle Hash Tree [Mer80] or authenticated skip list [EKPT09]), the index information may be *implicitly* embedded into the authentication meta-data.

2.3.2 Random Sampling and Error Erasure Code

We just discussed that a file is broken into many blocks, and in a verification the prover only accesses a small subset of blocks specified by the verifier. A natural question is that: How should the verifier sample the subset of blocks? If no extra knowledge about error distribution among blocks is present, uniformly random sampling could be the best strategy for the verifier to maximize the error detection probability.

Error erasure encoding (e.g. Reed-Solomon code [RS60]) allows the verifier to achieve high error detection rate when randomly sampling only a small number (possibly constant) of blocks during one verification, at the cost of file size expansion.

Suppose an error erasure encoded file consists of n blocks $\mathtt{F}_0, \ldots, \mathtt{F}_{n-1}$, such that any ρn number of blocks can recover the original file, where $\rho n \in \{1, 2, 3, 4, \ldots, n\}$. If the original file is unable to be recovered using the error erasure decoding algorithm, then at most $\rho n - 1$ number of data blocks remain intact. The probability that a randomly chosen block is intact is at most $\rho - \frac{1}{n}$, and the probability (False Acceptance Rate) that ℓ number of independently and randomly chosen blocks are all intact is at most $\left(\rho - \frac{1}{n}\right)^{\ell} < \rho^{\ell}$. Thus a random sample of size ℓ will hit at least one corrupted block with probability at least $1 - \left(\rho - \frac{1}{n}\right)^{\ell} > 1 - \rho^{\ell}$. This lower bound is a decreasing function of $\rho \in [\frac{1}{n}, 1]$, indicating that the more redundancy introduced in the error erasure encoding (i.e. the smaller ρ), the higher the hitting probability is. Notice that if these ℓ blocks are chosen at random such that all ℓ blocks have distinct indices, i.e. random sampling without replacement, the above lower bound on error detection probability still holds.

Table 2.1 lists the "False Acceptance Rate" that ℓ random samples do not hit any corrupted data blocks with $\rho \in \{0.98, 0.99\}$ and $\ell \in \{100, 300, 500, 700\}$. Note that the the storage overhead due to erasure encoding is $1/0.99 - 1 \approx 0.0101$ of original file size, if $\rho = 0.99$; $1/0.98 - 1 \approx 0.0204$, if $\rho = 0.98$.

Table 2.1: The False Acceptance Rate Versus Challenge Size ℓ and Erasure Code Rate ρ. The challenge size ℓ represents the number of data blocks accessed in a verification.

Challenge Size	**False Acceptance Rate** ρ^{ℓ} with $\rho = 0.99$	**False Acceptance Rate** ρ^{ℓ} with $\rho = 0.98$
$\ell = 100$	0.366032341	0.132619556
$\ell = 300$	0.049040894	0.002332506
$\ell = 500$	0.006570483	0.000041024
$\ell = 700$	0.000880311	0.000000722

Remark 1 *We remark two points:*

- *In a verification of proofs of storage scheme, the verifier chooses a subset C of ℓ indices, and asks the prover to check those data blocks with index within the set C. To reduce communication cost, a natural thought [ABC$^+$07] is to* *represent the set C compactly with a short seed s of some secure pseudorandom function* PRF*:* $C = \{\mathsf{PRF}_s(i) : i \in [0, \ell - 1]\}$*. However, Shacham and Waters [SW08a] points out that this intuitive method actually requires rigorous security proof, since the dishonest prover knows the values of seeds and the typical indistinguishability argument of pseudorandom function does not apply here. This issue influences works [ABC+07, ABC+11b, CX08, SW08a] and all of three schemes in Part I of this dissertation. The consequence is that, if we adopt this compact representation of set C, our proposed schemes are only provable secure in random oracle model, instead of standard model.*

- *Some proofs of storage schemes built on Merkle Hash Tree choose consecutive blocks, in order to reduce proof size, at the cost of sacrificing error detection probability. In comparison, in all of proofs of storage schemes proposed in this dissertation, the proof size is independent on the choices of the subset of blocks to be checked in a verification.*

2.3.3 Homomorphic Cryptography

Previously, we analyzed the probability that a random sample of size ℓ will hit at least one corrupted block, when the encoded file is so corrupted such that the original file cannot be recovered. Once the hit event occurs, the cryptographic authentication method should detect errors with overwhelming high probability—this is a requirement in security aspect. In the efficiency aspect, ℓ selected blocks and authentication tags are too large as a proof. Ideally, a homomorphic authentication method may allow the prover to aggregate those ℓ block-tag pairs into a single block-tag pair as proof, and the verifier can detect any error among these ℓ blocks caused by a computationally bounded adversary with overwhelming high probability, by checking only the single aggregated block-tag pair.

Linearly homomorphic cryptography allows the prover to produce an authentication tag for a linear combination of the ℓ selected data blocks with only a public key. Among various homomorphic cryptography, linearly homomorphic cryptography could be a good choice in constructing efficient proofs of storage scheme, since very efficient linearly homomorphic authentication scheme exists, and more importantly, the original data blocks can be recovered efficiently from a number of authenticated aggregated blocks by solving a linear equation system.

2.3.4 Framework

A typical framework of proofs of storage scheme is as below: A data file is error erasure encoded and the encoded file consists of many small blocks. Each file is distinguished from other files with a unique identifier and each data block is distinguished from the other blocks within the same file with a unique sequence number. For each data block, an authentication tag is generated w.r.t. the corresponding file identifier and block sequence number, using some homomorphic authentication method. During a verification, the verifier samples a random subset of ℓ data blocks, and the prover produces an aggregated block-tag pair from these selected ℓ block-tag pairs by applying the homomorphic property. Only the aggregated block-tag pair is sent back as proof to the verifier. Due to the security property of the homomorphic authentication method, if verifier accepts the proof, then with overwhelming high probability, those ℓ selected data blocks are intact. The integrity of these ℓ blocks implies the integrity of the original data file with high probability, due to the error erasure encoding.

All of Ateniese *et al.* [ABC+07,ABC+11b,AKK09], Chang and Xu [CX08], Shacham and Waters [SW08a], and the three schemes POS1, POS2 and POS3 proposed in Part I of this dissertation, fit in the above framework.

2.4 Related Work

2.4.1 Early Approaches

Our research is motivated by applications in remote-backup and peer-to-peer backup [ATS04, BBST02, LD06]. Peer-to-peer backup system requires a mechanism to maintain the availability and integrity of data stored in peer nodes. Li and Dabek [LD06] proposed to choose neighboring nodes based on the social relationships and relies on the heuristic assumption that people are more likely cooperative with friends.

2.4.2 Online Memory Checker and Sublinear Authenticator

Remote integrity verification has a close relationship with memory integrity verification [$\text{BEG}^+91, \text{SCG}^+03, \text{NR05}, \text{DNRV09}$]. The notion of authenticator proposed by Naor and Rothblum [NR05] is formulated for memory integrity checker. There is an essential difference between memory checker and proofs of storage problem studied in this dissertation: in the memory checker problem, an honest prover will follow the specified protocol to verify its storage, where the storage is untrusted and could be altered by outside attackers or random hardware failure; in the proofs of storage problem, both the prover and its storage are untrusted, such that the prover could do *anything*[3] during a verification and the storage could be altered carefully by the dishonest prover. Consequently, any solution to a proofs of storage problem is also a solution to the memory checker problem. Thus, the lower bound on complexity of memory checker discovered by Naor and Rothblum [NR05] also applies to proofs of storage. Additionally, the idea of introducing redundancy to tradeoff resources is useful in proofs of storage.

2.4.3 Proofs of Retrievability and Provable Data Possession

Recently, there is a growing interest in the cryptographic aspects of cloud storage problem. Perhaps Filho and Barreto [FB06] first studied the scenario where the verifier does not have the original. They described two potential applications: *uncheatable data transfer* and *demonstrating data possession*, and proposed the RSA-based scheme. Juels and Kaliski [JK07] proposed a formulation called Proofs of Retrievability $\mathcal{POR}$ for the proofs of storage problem. Essentially, in a $\mathcal{POR}$ scheme, if the cloud storage server can pass verification with a noticeable probability, then the verifier can *retrieve* the original data from messages collected during polynomially many verification interactions between the verifier and the cloud storage server. So $\mathcal{POR}$ formulation allows a user to ensure whether his/her file is indeed in the cloud storage in an intact form without actually downloading the file. However, the $\mathcal{POR}$ construction in Juels and Kaliski [JK07] can support only a predefined constant number

[3]The only limitation is that the prover's computation resource is polynomially bounded.

of verifications. A refined security formulation is given in [BJO09b] .

Ateniese *et al.* [ABC$^+$07] gave an alternative formulation called *Provable Data Possession* for proofs of storage problem, and proposed an efficient construction. Their method can be viewed as an extension of the RSA-based scheme. Similarly, the scheme named RSAh given in our publication [CX08] exploits similar idea, and the third scheme POS3 proposed in this dissertation is a refined version of RSAh, which is more efficient than RSAh and Ateniese *et al.* [ABC$^+$07].

Shacham and Waters [SW08a] proposed two efficient constructions of $\mathcal{POR}$, where one scheme supports private key verification and the other supports public key verification.

Ateniese and Kamara and Katz [AKK09] studied how to utilize homomorphic linear identification scheme to construct proofs of storage scheme. Dodis and Vadhan and Wichs [DVW09] studied how to construct proofs of retrievability scheme through hardness application. All of schemes in [ABC$^+$07, CX08, SW08a, AKK09] utilize some underlying linear homomorphic authentication methods, which also has applications in network coding [AB09, BFKW09]. Several proofs of storage schemes with pre-defined number of verifications have been proposed in works [JK07, ADPMT08, DVW09]. A survey of proofs of storage is given by Yang and Jia [YJ11].

In this dissertation, we will compare our second method POS2 and third method POS3 to Ateniese *et al.* [ABC$^+$07, ABC$^+$11b] and/or Shacham and Waters [SW08a].

2.4.4 Proofs of Storage with More Features

Very recently, several works [CKBA08, BJO09a, EKPT09, WWL$^+$09, WWRL10] have devoted to extend proofs of storage to support more features. In [CKBA08], verifier checks whether the cloud storage server indeed keeps multiple intact copies of a user's file. Dynamic-$\mathcal{PDP}$ [EKPT09] allows insertion and deletion of data blocks on the fly after setup. Proofs of storage schemes supporting public verifiability are proposed in Shacham and Waters [SW08a] and Wang [WWL$^+$09] and the privacy issue in public verification is studied in Wang [WWRL10].

2.4.5 More General Delegated Computation and Proofs of Storage

Kate and Zaverucha and Goldberg [KZG10] proposed an efficient commitment scheme for polynomial and Benabbas and Gennaro and Vahlis [BGV11] proposed a secure delegation scheme for polynomial evaluation. Both schemes can be extended to support $\mathcal{POR}$ easily but with limitations: the $\mathcal{POR}$ scheme implied in Kate and Zaverucha and Goldberg [KZG10] has large storage cost on client side and the $\mathcal{POR}$ scheme implied in Benabbas and Gennaro and Vahlis [BGV11] has large storage and computation cost on the server side. We will elaborate more on Kate and Zaverucha and Goldberg [KZG10]'s polynomial commitment scheme in Section 5.3.1 in Chapter 5.

The two solutions [GGP10,CKV10] to verifiable delegation of generic computation task based on fully homomorphic encryption [Gen09], also imply a secure proofs of storage scheme. However, the efficiency overheads in communication, storage and computation on the server side are too large, rendering the resulting proofs of storage schemes impractical.

Chapter 3

Definitions and Formulation

In this chapter, we provide preliminary definitions, and give security formulation for proofs of storage problem.

3.1 Preliminaries

3.1.1 Terminologies

Definition 1 (Negligible [Gol06]) *A non-negative function $\epsilon(\lambda)$ is* negligible *in λ, if for any positive integer c, for all sufficiently large integer λ, $0 \leq \epsilon(\lambda) \leq \lambda^{-c}$.*

Definition 2 (Overwhelming High Probability) *Let $\mu(\cdot)$ be a function which measures the probability of some event. We say $\mu(\lambda)$ is* overwhelming high probability*, if $1 - \mu(\lambda)$ is negligible in λ.*

Definition 3 (Noticeable [Gol06]) *A function $\epsilon(\lambda)$ is* noticeable *in λ, if there exists a positive integer c, for all sufficiently large integer λ, $\epsilon(\lambda) \geq \lambda^{-c}$.*

Note that (1) any noticeable function is non-negligible; (2) any negligible function is non-noticeable; (3) there exists function which is both non-noticeable and non-negligible (See the below Example 1).

Example 1 *The function $\epsilon(\cdot)$ defined as below is neither noticeable nor negligible.*

$$\epsilon(\lambda) = \begin{cases} \frac{1}{2} & (\textit{ if } \lambda \textit{ is odd}) \\ 2^{-\lambda} & (\textit{ if } \lambda \textit{ is even}) \end{cases}$$

3.1.2 Conventions

We use expression $\mathsf{S} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ to represent a scheme named S, which consists of three algorithms KeyGen, Sign and Verify. If necessary, we will use notation $\mathsf{S.KeyGen}$ to represent the algorithm KeyGen in the scheme S, to distinguish it from key generating algorithms from other schemes.

In this dissertation, the word "random" refers to "uniform random", if there is no distribution specified.

We also clarify two distinct concepts *valid proof* and *genuine proof.*

Valid Proof : A proof is valid, if it is accepted by the verifier.

Genuine Proof : A proof is genuine, if it is the same as the one generated by an honest (deterministic[1]) prover on the same query.

We give an example to distinguish valid proof and genuine proof.

Example 2 *Take as example the straightforward approach where Alice keeps a hash value and downloads her file from Bob to perform a local data integrity check during each verification. If Bob* somehow *finds another file* $\mathtt{F}' \neq \mathtt{F}$*, such that* $\mathsf{hash}(\mathtt{F}) = \mathsf{hash}(\mathtt{F}')$*, and returns* $\mathtt{F}'$ *back to Alice. Alice will accept* $\mathtt{F}'$ *as a valid proof. In this case, both* $\mathtt{F}$ *and* $\mathtt{F}'$ *are valid proofs, but only* $\mathtt{F}$ *is the genuine proof.*

3.1.3 Summary of Notations

We summarize the key notations used in Part I of this dissertation in Table 3.1.

[1]The provers in all of three schemes in Part I are deterministic.

Table 3.1: Summary of Key Notations in Part I of this dissertation

Notation	**Semantics**
$x := a$	Assign the value a to the variable x.
$x \overset{\text{def}}{=} A$	The statement A defines the semantics of x.
$x \overset{\$}{\leftarrow} S$	Uniformly randomly choose x from a finite set S.
$\|$	Binary operator $\|$ denotes the unambiguous string concatenation which allows unique unambiguous decomposition.
$[a, b]$	The set $\{a, a+1, \ldots, b\}$ where both a and b are integers and $a \leq b$.
λ	Security parameter. Group element size in bits.
n	The number of data blocks in a data file.
m	The number of sectors in a data block. Typically each sector is a group element.
ℓ	The number of data blocks accessed during a verification.
$\vec{\boldsymbol{u}}$	A vector of form $(u_0, u_1, u_2, \ldots, u_{d-1})$, where d is the dimension of vector $\vec{\boldsymbol{u}}$.
$f_{\vec{u}}(x)$	A polynomial $u_0 + u_1 x + u_2 x^2 + \ldots + u_{d-1} x^{d-1}$ of degree $d-1$ with vector $\vec{\boldsymbol{u}}$ as coefficient, where d is the dimension of vector $\vec{\boldsymbol{u}}$.
PPT	Probabilistic Polynomial Time.
negl	Some negligible function [Gol06].
MAC	Message Authentication Code [Gol06].
PRF	Pseudorandom function [Gol06].
$\mathcal{POR}$	Proofs of Retrievability [JK07].
$\mathcal{PDP}$	Provable Data Possession [ABC$^+$07].
S1	The name of the homomorphic MAC scheme proposed in Chapter 4.
S2	The name of the homomorphic MAC scheme proposed in Chapter 5.
POS1	The name of proofs of storage scheme proposed in Chapter 4.
POS2	The name of proofs of storage scheme proposed in Chapter 5.
POS3	The name of proofs of storage scheme proposed in Chapter 6.
S1.KeyGen	The key generating algorithm KeyGen of scheme S1.

3.2 Formulation: Proofs of Retrievability

Proofs of storage requires to periodically, remotely and reliably verify the integrity of data stored in a cloud storage, without retrieving the data file. *Proofs of Retrievability* ($\mathcal{POR}$) model proposed by Juels and Kaliski [JK07] is among the first few attempts to formulize the notion of "remotely and reliably verifying the data integrity".

In this section, we review the $\mathcal{POR}$ model, which is proposed by Juels and Kaliski [JK07] and revisited by Shacham and Waters [SW08a].

3.2.1 System Model

We restate the $\mathcal{POR}$ [JK07, SW08a] model as below, with slight modifications on notations. We adopt the 1-round prove-verify version in Juels and Kaliski [JK07] for simplicity.

Definition 4 ($\mathcal{POR}$ [JK07, SW08a]) *A* Proofs Of Retrievability *($\mathcal{POR}$) scheme consists of four algorithms* $(\mathsf{KeyGen}, \mathsf{DEncode}, \mathsf{Prove}, \mathsf{Verify})$*:*

- $\mathsf{KeyGen}(1^\lambda) \to (pk, sk)$*: Given security parameter λ, the probabilistic key generating algorithm, run by the data owner Alice, outputs a public-private key pair (pk, sk).*

- $\mathsf{DEncode}(sk, \mathtt{F}) \to (\mathsf{id}_\mathtt{F}, \hat{\mathtt{F}}, n)$*: Given the private key sk and a data file $\mathtt{F}$, the encoding algorithm* $\mathsf{DEncode}$*, run by Alice, produces a unique identifier* $\mathsf{id}_\mathtt{F}$ *and the encoded file $\hat{\mathtt{F}}$ with size n (in term of number of blocks), where* (id, n) *will be kept by the data owner Alice and* $(\mathsf{id}, \hat{\mathtt{F}})$ *will be kept by the cloud storage server Bob.*

- $\mathsf{Prove}(pk, \mathsf{id}_\mathtt{F}, \hat{\mathtt{F}}, \mathsf{Chall}) \to \psi$*: Given the public key pk, an identifier* $\mathsf{id}_\mathtt{F}$*, an encoded file $\hat{\mathtt{F}}$, and a challenge query* Chall *as input, the prover algorithm* Prove*, run by cloud storage server Bob, produces a proof ψ.*

- $\mathsf{Verify}(sk, \mathsf{id}_\mathtt{F}, \mathsf{Chall}, \psi) \to \mathtt{accept}$ *or* $\mathtt{reject}$*: Given the private key sk, an identifier* $\mathsf{id}_\mathtt{F}$*, a challenge query* Chall*, and a proof ψ as input, the deterministic* *verifier algorithm* Verify*, run by the data owner Alice, will output either* `accept` *or* `reject`*.*

A proofs of storage system between data owner Alice and cloud storage server Bob can be implemented using a $\mathcal{POR}$ scheme (KeyGen, $\mathsf{DEncode}$, Prove, Verify): *At the very beginning, Alice chooses a security parameter* λ*, and generates a pair of public-private keys* $(pk, sk) := \mathsf{KeyGen}(1^\lambda)$ *where* pk *is made public and* sk *is kept private. Then Alice will interact with Bob in the following way.*

Setup Phase: *Alice and Bob will carry out the setup phase for one time per each file.*

- *Alice preprocesses her file* $\mathtt{F}$ *to produce* $(\mathsf{id}, \hat{\mathtt{F}}, n) := \mathsf{DEncode}(sk, \mathtt{F})$*. Alice sends* $(\mathsf{id}, \hat{\mathtt{F}})$ *to Bob and removes* $\hat{\mathtt{F}}$ *from local storage.*

At the end of setup phase, Alice only has (sk, id, n) *in her local storage, and Bob has* $(pk, \mathsf{id}, \hat{\mathtt{F}})$ *in his storage.*

Verification Phase: *The verification phase consists of multiple verification sessions. In each session, Alie and Bob interact as below.*

- *To check the file with identifier* id*, Alice chooses a random challenge* Chall *and sends* Chall *together with the identifier* id *to Bob.*
  Note: Typically, the challenge Chall includes as a part a subset $C \subset [0, n-1]$, which indicates those blocks that Bob should access.
- *Bob is supposed to run algorithm* Prove *upon the encoded file* $\hat{\mathtt{F}}$ *corresponding to the identifier* id *to generate a proof* $\psi := \mathsf{Prove}(pk, \mathsf{id}, \hat{\mathtt{F}}, \mathsf{Chall})$*, and send* ψ *to Alice.*
- *Alice runs the algorithm* Verify *with the private key sk to check the validity of the received proof* ψ*. Alice computes* $b := \mathsf{Verify}(sk, \mathsf{id}, \mathsf{Chall}, \psi) \in \{\mathtt{accept}, \mathtt{reject}\}$ *and outputs* b*.*

Definition 5 (Completeness of $\mathcal{POR}$) *A* $\mathcal{POR}$ *scheme (*KeyGen*,* $\mathsf{DEncode}$*,* Prove*,* Verify*) is* complete*, if an honest prover (who ensures the integrity of his storage and* *executes the procedure* Prove *to compute a proof) will always be accepted by the verifier. More precisely, for any key pair* (pk, sk) *generated by* KeyGen*, and any data file* $\mathtt{F}$*, any challenge query* Chall*, if* $\psi \leftarrow \mathsf{Prove}(pk, \mathsf{id}_{\mathtt{F}}, \hat{\mathtt{F}}, \mathsf{Chall})$*, then* $\mathsf{Verify}(sk, \mathsf{id}_{\mathtt{F}}, \mathsf{Chall}, \psi)$ *outputs* `accept` *with probability 1, where* $(\mathsf{id}_{\mathtt{F}}, \hat{\mathtt{F}}, n) \leftarrow \mathsf{DEncode}(sk, \mathtt{F})$.

3.2.2 Security Model

3.2.2.1 Trust Model and Scope of Topic

In a proofs of storage system, only the data owner Alice is trusted and the cloud storage server Bob is treated as untrusted and potentially malicious.

We clarify that, the following topics are out of the scope of this dissertation: (1) Confidentiality of Alice's data against Bob; (2) Support of dynamic operations like insertion and deletion of data blocks; (3) Denial of Service Attack; (4) Frame attack where dishonest Alice claims honest Bob was cheating.

3.2.2.2 $\mathcal{POR}$ Security Game

We rephrase the $\mathcal{POR}$ security game, which is proposed by Juels and Kaliski [JK07] and revisited by Shacham and Waters [SW08a], in a standard way. The $\mathcal{POR}$ security game between a *probabilistic polynomial time* (PPT) adversary $\mathcal{A}$ and a PPT challenger $\mathcal{C}$ w.r.t. a $\mathcal{POR}$ scheme $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{DEncode}, \mathsf{Prove}, \mathsf{Verify})$ is as below.
Setup: The challenger $\mathcal{C}$ runs the key generating algorithm KeyGen to obtain public-private key pair (pk, sk). The challenger $\mathcal{C}$ gives the public key pk to the adversary $\mathcal{A}$ and keeps the private key sk securely.
Learning: The adversary $\mathcal{A}$ adaptively makes queries, where each query is one of the following:

- Store query ($\mathtt{F}$): Given a data file $\mathtt{F}$ chosen by $\mathcal{A}$, the challenger $\mathcal{C}$ responses by running data encoding algorithm $(\mathsf{id}, \hat{\mathtt{F}}, n) \leftarrow \mathsf{DEncode}(sk, \mathtt{F})$ and sending the encoded data file $\hat{\mathtt{F}}$ together with its identifier id to $\mathcal{A}$. The challenger $\mathcal{C}$ will keep (id, n).

- Verification query (id): Given a file identifier id chosen by $\mathcal{A}$, if id is the (partial) output of some previous store query that $\mathcal{A}$ has made, then the challenger $\mathcal{C}$ initiates a $\mathcal{POR}$ verification with $\mathcal{A}$ w.r.t. the data file $\mathtt{F}$ associated to the identifier id in this way:
  - $\mathcal{C}$ chooses a random challenge Chall using the meta-data n;
  - $\mathcal{A}$ produces a proof ψ w.r.t. the challenge Chall;
    Note: adversary $\mathcal{A}$ may generate the proof in an arbitrary method rather than applying the algorithm Prove.
  - $\mathcal{C}$ verifies the proof ψ by running algorithm $\mathsf{Verify}(sk, \mathsf{id}, \mathsf{Chall}, \psi)$. Denote the output as $b \in \{\mathtt{accept}, \mathtt{reject}\}$.

  $\mathcal{C}$ sends the decision bit b to $\mathcal{A}$ as feedback. Otherwise, if id is not the (partial) output of any previous store query that $\mathcal{A}$ has made, $\mathcal{C}$ does nothing.

Commit: Adversary $\mathcal{A}$ chooses a file identifier id^* among all file identifiers she obtains from $\mathcal{C}$ by making store queries in **Learning** phase, and commits id^* to $\mathcal{C}$. Let $\mathtt{F}^*$ denote the data file associated to identifier id^*.
Retrieve: The challenger $\mathcal{C}$ initiates ζ number of $\mathcal{POR}$ verifications with $\mathcal{A}$ w.r.t. the data file $\mathtt{F}^*$, where $\mathcal{C}$ plays the role of verifier and $\mathcal{A}$ plays the role of prover, as in the **Learning** phase. From messages collected in these ζ interactions with $\mathcal{A}$, $\mathcal{C}$ extracts a data file $\mathtt{F}'$ using some PPT extractor algorithm. The adversary $\mathcal{A}$ wins this game, if and only if $\mathtt{F}' \neq \mathtt{F}^*$.

The adversary $\mathcal{A}$ is ϵ-*admissible* [SW08a], if the probability that $\mathcal{A}$ convinces $\mathcal{C}$ to accept in a verification in the **Retrieve** phase, is at least $\epsilon \in (0, 1)$. We denote the above game as $\mathsf{Game}^{\mathcal{E}}_{\mathcal{A}}(\zeta)$.

Definition 6 ([JK07, SW08a]) *A $\mathcal{POR}$ scheme $\mathcal{E}$ is sound, if for any PPT ϵ-admissible adversary $\mathcal{A}$ with ϵ being a noticeable function in the security parameter λ, there exists a polynomial function ζ in λ, such that the advantage $\mathsf{Adv}^{\mathcal{E}}_{\mathcal{A}}(\zeta)$ defined* *as below is negligible in λ.*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathcal{E}}(\zeta) \overset{\text{def}}{=} \mathsf{Pr}\left[\mathcal{A} \textit{ wins } \mathsf{Game}_{\mathcal{A}}^{\mathcal{E}}(\zeta)\right]. \tag{3.1}$$

Notice that the above definition is slightly different from [JK07, SW08a], in which ϵ is non-negligible and the extractor algorithm runs in time $\Omega(\epsilon^{-1})$. When ϵ is non-negligible and not noticeable, $\Omega(\epsilon^{-1})$ is not upper-bounded by any fixed polynomial.

3.2.2.3 Clarification of Security Model

There should be no confusion between the security formulation and the real world application of a $\mathcal{POR}$ scheme. We remark that the security games $\mathsf{Game}_{\mathcal{A}}^{\mathcal{E}}$, especially the $\mathsf{Retrieve}$ phase, are only for security formulation, and applications of a $\mathcal{POR}$ scheme do not necessarily follow the description of the security game exactly. For example, in real world applications, the data owner will be the one who chooses the data file, instead of the cloud storage server, and the data owner can retrieve her file by simply requesting the cloud storage server to send it back.

The $\mathsf{Retrieve}$ phase in the security games just ensures that, in theory, user's file *can* be recovered from multiple verifications with the cloud storage server efficiently (using some PPT extractor algorithm), as long as the cloud storage server can pass a noticeable fraction of challenge queries. Essentially, a secure $\mathcal{POR}$ scheme provides a mechanism, in which the data owner will be guaranteed that her data file can be efficiently recovered from the server's storage at the moment that a verification is accepted, without *actually* downloading the file from the server. Furthermore, this guarantee is based on the assumption that the cloud storage server is not able to solve some cryptographic hard problems[2], without trusting in the cloud storage server.

3.2.3 Alternative Formulation: Provable Data Possession

An alternative formulation *Provable Data Possession* ($\mathcal{PDP}$) proposed by Ateniese *et al.* [ABC$^+$07] will be reviewed later in Chapter 6, since the third proofs of storage

[2]For information-theoretical secure $\mathcal{POR}$ schemes (e.g. [DVW09]), such assumption is not necessary.

scheme proposed in Chapter 6 will be proved under the $\mathcal{PDP}$ model, while the first two proofs of storage schemes proposed in Part I of this dissertation will be proved under $\mathcal{POR}$ model. It is well-known that $\mathcal{PDP}$ is a weaker formulation than $\mathcal{POR}$, in the sense that any secure $\mathcal{POR}$ scheme is a secure $\mathcal{PDP}$ scheme, but not vice versa. Our third proofs of storage scheme is such an example: It is provably secure under $\mathcal{PDP}$, and insecure under $\mathcal{POR}$.

Chapter 4

Proofs of Retrievability from Linearly Homomorphic Message Authentication Code

A *Linearly Homomorphic Message Authentication Code (MAC)* scheme allows any third party to compute a valid MAC value for a linear combination of messages $\mathtt{M}_i$'s from the MAC values of these messages $\mathtt{M}_i$'s, using the public key. This chapter will propose a linearly homomorphic MAC scheme, which we refer to as $\mathsf{S1}$, and apply it to construct a proofs of storage scheme, which we refer to as $\mathsf{POS1}$ in this dissertation. The proposed scheme $\mathsf{POS1}$ is very efficient in communication and computation, and will be proved under the Proofs of Retrievability formulation. The result in this chapter is published in Chang and Xu [CX08] (under name "$\mathtt{HTAG}$"). We remark that Chang and Xu [CX08] declared the $\mathtt{HTAG}$ scheme as an independent work of Shacham and Waters [SW08a].

4.1 Overview

Section 2.2.2 (on page 11) described a proofs of storage scheme constructed from a non-homomorphic MAC scheme. In each verification of this MAC based method, the verifier selects a random subset of ℓ indices, and the prover finds all data blocks corresponding to the selected indices and sends these data blocks together with their MAC values to the verifier. Thus the communication cost is dominated by the ℓ message-MAC pairs. A wishful but natural thought is that: Can we somehow *aggregate* all of these ℓ message-MAC pairs into a single message-MAC pair, such that any errors in these ℓ message-MAC pairs introduced by a computationally bounded adversary can be detected by checking only the single aggregated message-MAC pair? Linearly homomorphic MAC is a good answer. Since such homomorphic MAC allows the prover to compute a valid MAC value for a linear combination of these ℓ selected data blocks from only these ℓ message-MAC pairs and a public key. Thus only one message, which is the linear combination of previous ℓ data blocks, together with its MAC value is sent back to the verifier, yet the verifier is able to detect any possible errors within these ℓ data blocks with overwhelming high probability by checking the received message-MAC pair.

4.1.1 A Brief Description of proofs of storage scheme POS1

Setup. Suppose Alice wants to backup her data file $\mathtt{F}$ to a cloud storage server Bob. Alice first encodes file $\mathtt{F}$ using some error erasure code, and resulting enlarged file consists of n data blocks $\mathtt{F}_0, \ldots, \mathtt{F}_{n-1}$. Alice chooses a prime p, a random secret value τ, and a random secret seed, denoted as seed, of a pseudorandom function PRF [Gol06]. For each data block $\mathtt{F}_i$, Alice generates an authentication tag σ_i using a linearly homomorphic MAC scheme:

$$\sigma_i := \mathsf{PRF}_{\mathsf{seed}}(i) + \tau \mathtt{F}_i \mod p. \tag{4.1}$$

Alice sends all data blocks together with authentication tags, i.e. $\{(i, \mathtt{F}_i, \sigma_i) : i \in [0, n-1]\}$, to Bob.

Verification. Later, Alice may remotely verify the integrity of her data file stored with Bob periodically. In each verification session, Alice randomly selects a subset $C \subset [0, n-1]$ of indices and selects a random weights ν_i for each $i \in C$. Alice sends $\{(i, \nu_i) : i \in C\}$ as challenge to Bob. Bob then finds all data blocks $\mathtt{F}_i$'s and authentication tags σ_i's with index $i \in C$, and apply the linear homomorphism to compute an aggregated message-MAC pair $(\mathtt{M}, \sigma)$ as below:

$$\mathtt{M} := \sum_{i \in C} \nu_i \mathtt{F}_i \mod p; \tag{4.2}$$

$$\sigma := \sum_{i \in C} \nu_i \sigma_i \mod p. \tag{4.3}$$

Upon receiving $(\mathtt{M}, \sigma)$ as response from Bob, Alice checks the following equality with private information (seed, τ):

$$\sigma \overset{?}{=} \sum_{i \in C} \nu_i \mathsf{PRF}_{\mathsf{seed}}(i) + \tau \mathtt{M} \mod p. \tag{4.4}$$

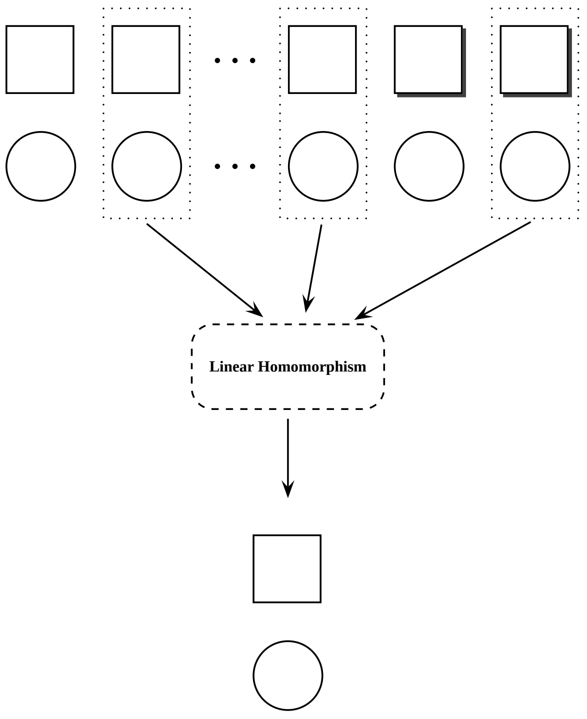


Figure 4.1: $\mathcal{POR}$ scheme $\mathsf{POS1}$ constructed from linearly homomorphic MAC $\mathsf{S1}$. A square represents a data block and a circle represents an authentication tag. Detailed explanation is in the paragraph with title "Illustration Picture".

Illustration Picture. Figure 4.1 illustrates the scheme POS1 that we just briefed. In Figure 4.1, a square represents a data block and a circle represents an authentication tag (i.e. a MAC value). Each square together with the circle which lies just below, represent a valid message-MAC pair. Those shaded squares represent data blocks that are generated by the error erasure code. Figure 4.1 shows that during a verification, a subset of 3 message-MAC pairs are selected and are aggregated into a single message-MAC pair by applying the linear homomorphism property.

4.1.2 Organization

The rest of this chapter is organized as below. Section 4.2 gives the definition of linearly homomorphic MAC scheme and Section 4.3 provides the construction of a linearly homomorphic MAC scheme named S1. Next, we propose the proofs of storage scheme POS1 based on S1 in Section 4.4 and analyze its performance in Section 4.5. We analyze the security of S1 and POS1 in Section 4.6 and Section 4.7, respectively. In the end, Section 4.8 summarizes this chapter.

4.2 Linearly Homomorphic MAC: Definition

A linearly homomorphic MAC scheme consists of four algorithms ($\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Combine}, \mathsf{Verify}$), where each algorithm is described as below.

- $\mathsf{KeyGen}(1^\lambda) \to (spk, ssk)$: The probabilistic key generating algorithm takes the security parameter λ as input, and outputs a pair of public and private key (spk, ssk).

- $\mathsf{Sign}(ssk, i, \mathtt{M}) \to \sigma$: The signing algorithm takes the private key ssk, an index i and a message $\mathtt{M}$ as input, and outputs a signature σ.
  Note: The algorithm Sign *is stateful.*

- $\mathsf{Combine}(spk, \{(i, \mathtt{M}_i, \sigma_i, \nu_i) : i \in C\},) \to (\mathtt{M}, \sigma)$: The algorithm Combine implements the homomorphic property. Taking as input the public key spk, the sequence of tuples $(i, \mathtt{M}_i, \sigma_i, \nu_i), i \in C$, where σ_i is a MAC value of a message $\mathtt{M}_i$

w.r.t. an index i, and ν_i is a weight of $\mathtt{M}_i$ in a linear combination, algorithm $\mathsf{Combine}$ outputs a message-MAC pair $(\mathtt{M} := \sum_{i \in C} \nu_i \mathtt{M}_i,\ \sigma)$.

- $\mathsf{Verify}(ssk, \mathtt{M}, \sigma, \{(i, \nu_i) : i \in C\}) \to \mathtt{accept}$ or $\mathtt{reject}$: The deterministic verification algorithm takes the private key ssk, a message $\mathtt{M}$, a MAC value σ, a sequence of tuples $\{(i, \nu_i) : i \in C\}$ as input, and outputs either $\mathtt{accept}$ or $\mathtt{reject}$.

Definition 7 (Linearly Homomorphic MAC) *A MAC scheme* $\mathsf{S} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Combine}, \mathsf{Verify})$ *is* Linearly Homomorphic*, if for any public-private key pair* $(spk, ssk) := \mathsf{KeyGen}(1^\lambda)$ *generated by the key generating algorithm* KeyGen *and for any sequence of message-MAC pairs* $\{(i, \mathtt{M}_i, \sigma_i), i \in C\}$ *generated by the signing algorithm* Sign *under the private key ssk, the output* $(\sum_{i \in C} \nu_i \mathtt{M}_i,\ \sigma) := \mathsf{Combine}(spk, \{(i, \mathtt{M}_i, \sigma_i, \nu_i) : i \in C\})$ *is accepted as a valid message-MAC pair by the verification algorithm* Verify *under the private ssk w.r.t.* $\{(i, \nu_i) : i \in C\}$*, i.e.* $\mathsf{Verify}(ssk, \sum_{i \in C} \nu_i \mathtt{M}_i, \sigma, \{(i, \nu_i) : i \in C\}) = \mathtt{accept}$.

4.3 Linearly Homomorphic MAC: Construction

The construction of the MAC scheme $\mathsf{S1} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Combine}, \mathsf{Verify})$ is as below.

4.3.1 Construction of $\mathsf{S1}$

$\underline{\mathsf{S1.KeyGen}(1^\lambda) \to (spk, ssk)}$

Choose at random a λ bits long prime number p. Choose at random a group element τ from $\mathbb{Z}_p^*$: $\tau \xleftarrow{\$} \mathbb{Z}_p^*$. Choose at random a seed, denoted as seed, from the key space of the pseudorandom function family $\{\mathsf{PRF}_{\mathsf{seed}} : \{0,1\}^{2\lambda} \to \mathbb{Z}_p\}$. The public key is $spk := (p)$ and the private key is $ssk := (\mathsf{seed}, \tau, p)$. Output (spk, ssk).

$\underline{\mathsf{S1.Sign}(ssk, i, \mathtt{M}) \to \sigma_i}$

The MAC value σ_i for a given message $\mathtt{M} \in \mathbb{Z}_p$ w.r.t. an index $i \in \{0,1\}^{2\lambda}$ is computed as below

$$\sigma_i := \mathsf{PRF}_{\mathsf{seed}}(i) \ + \ \tau\mathtt{M} \quad \bmod p. \tag{4.5}$$

$\underline{\mathsf{S1.Combine}(spk, \{(i, \mathtt{M}_i, \sigma_i, \nu_i) : i \in C\}) \to (\mathtt{M}, \sigma)}$

The MAC value for the linear combination $\mathtt{M} := \sum_{i \in C} \nu_i \mathtt{M}_i \bmod p$ is computed as below

$$\sigma := \sum_{i \in C} \nu_i \sigma_i \quad \bmod p. \tag{4.6}$$

$\underline{\mathsf{S1.Verify}\,(ssk,\ \mathtt{M},\ \sigma,\ \{(i, \nu_i) : i \in C\}) \to \mathtt{accept\ or\ reject}}$

Check the following equality with the private key $ssk = (\mathsf{seed}, \tau, p)$. Output `accept` if the equality holds; otherwise output `reject`.

$$\sum_{i \in C} \nu_i \ \mathsf{PRF}_{\mathsf{seed}}(i) \ + \ \tau \cdot \mathtt{M} \ \overset{?}{=} \ \sigma \quad \bmod p \tag{4.7}$$

4.3.2 Correctness

The following lemma is straightforward:

Lemma 4.1 (S1 is a linearly homomorphic MAC) *The proposed MAC scheme* S1 *is a linearly homomorphic MAC scheme under Definition 7.*

4.3.3 S1 is Symmetric Key Signcryption

We realize that the proposed MAC scheme S1 actually functions as (private key) signature scheme and encryption scheme simultaneously, i.e. S1 is a symmetric key signcryption scheme [Zhe97]. The algorithm S1.Sign also takes the role of encryption. The corresponding decryption algorithm is as below:

$\underline{\mathsf{S1.Decrypt}(ssk, \sigma, i) \to \mathtt{M}}$

The plaintext $\mathtt{M}$ is computed as below.

$$\mathtt{M} := (\sigma - \mathsf{PRF}_{\mathsf{seed}}(i))\, \tau^{-1} \mod p. \tag{4.8}$$

Consequently, $\mathsf{S1}$ may have applications in constructing authenticated error correcting code [BJO09b, LTT10].

4.4 POS1: A Proofs of Retrievability scheme constructed from Homomorphic MAC S1

4.4.1 Construction of POS1

Let $\mathsf{S1} = (\mathsf{S1.KeyGen}, \mathsf{S1.Sign}, \mathsf{S1.Combine}, \mathsf{S1.Verify})$ be a linearly homomorphic MAC scheme. We construct a $\mathcal{POR}$ scheme named $\mathsf{POS1}$ based on $\mathsf{S1}$ as below.

$\underline{\mathsf{POS1.KeyGen}(1^\lambda) \to (pk, sk)}$

Invoke the key generating algorithm $\mathsf{S1.KeyGen}$ of the MAC scheme $\mathsf{S1}$ to generate the public-private key pair $(pk, sk) := \mathsf{S1.KeyGen}(1^\lambda)$.

$\underline{\mathsf{POS1.DEncode}(sk, \mathtt{F}) \to (\mathsf{id}, \hat{\mathtt{F}}, n)}$

Let $\rho \in (0, 1)$ be a system parameter. Apply rate-ρ error erasure code on data file $\mathtt{F}$ to generate blocks $(\mathtt{F}_0, \ldots, \mathtt{F}_{n-1})$, such that each block $\mathtt{F}_i \in \mathbb{Z}_p$ and any ρn number of blocks $\mathtt{F}_i$'s can recover the original file $\mathtt{F}$. Choose a unique identifier id for file $\mathtt{F}$ from the space $\{0, 1\}^\lambda$. For each $i \in [0, n-1]$, generate a tag σ_i for block $\mathtt{F}_i$ as below:

$$\sigma_i := \mathsf{S1.Sign}(sk,\ \mathsf{id}\|i,\ \mathtt{F}_i), \tag{4.9}$$

where $\|$ denotes a string concatenation that allows unique decomposition. The encoded file is $\hat{\mathtt{F}} := \{(i, \mathtt{F}_i, \sigma_i) : i \in [0, n-1]\}$. Send $(\mathsf{id}, \hat{\mathtt{F}})$ to the cloud storage server and keep only (id, n) in local storage.

$\mathsf{POS1.Prove}(pk, \mathsf{id}, \hat{\mathtt{F}}, \{(i, \nu_i) : i \in C\}) \to (\mathtt{M}, \sigma)$

Receive $(\mathsf{id}, \{(i, \nu_i) : i \in C\})$ from the verifier as the challenge where C is a subset of $[0, n-1]$. Find the encoded file $\hat{\mathtt{F}} = \{(i, \mathtt{F}_i, \sigma_i) : i \in [0, n-1]\}$ associated with identifier id. Compute a message $\mathtt{M}$ and a MAC value σ using the homomorphic property of the MAC scheme $\mathsf{S1}$ as below

$$(\mathtt{M}, \sigma) \quad := \quad \mathsf{S1.Combine}(pk, \{(\mathsf{id} \| i, \mathtt{F}_i, \sigma_i, \nu_i) : i \in C\}). \tag{4.10}$$

The prover sends $(\mathtt{M}, \sigma)$ to the verifier as response.

$\mathsf{POS1.Verify}(sk, \mathsf{id}, \mathtt{M}, \sigma, \{(i, \nu_i) : i \in C\}) \to \mathtt{accept}$ **or** $\mathtt{reject}$

Invoke the algorithm $\mathsf{S1.Verify}$ to obtain $b \in \{\mathtt{accept}, \mathtt{reject}\}$ as below:

$$b := \mathsf{S1.Verify}(sk, \mathtt{M}, \sigma,\ \{(\mathsf{id} \| i,\ \nu_i) : i \in C\}). \tag{4.11}$$

Output b.

Remark 2 *To simplify the security proof, we assume that ν_i's, $i \in C$, forms a simple geometric sequence. More precisely, denote the elements of set C as $\{i_j \in [0, n-1] : j \in [0, \ell-1]\}$, where ℓ is the size of set C and $0 \le i_0 < i_1 < i_2 \ldots < i_{\ell-1} < n$. There exists some element $\nu \in \mathbb{Z}_p^*$, for each $j \in [0, \ell-1]$, $\nu_{i_j} = \nu^j \mod p$. Thus, ℓ distinct vectors of form $(\nu^0, \nu^1, \ldots, \nu^{\ell-1}) \in \left(\mathbb{Z}_p^*\right)^\ell$ can constitute a vandermonde matrix [MS58].*

4.4.2 Completeness

Lemma 4.2 (POS1 is Complete) *The above construction* POS1 *is complete under Definition 5 (on page 5).*

The completeness of POS1 is implied by the correctness of the underlying linearly homomorphic MAC scheme S1 (Lemma 4.1). Recall that, a $\mathcal{POR}$ scheme is complete, if any genuine proof generated by following the $\mathcal{POR}$ scheme honestly upon an intact copy of the file uploaded by Alice, will be accepted by the verifier Alice.

4.5 Performance Analysis

The scheme $\mathsf{POS1}$ is very efficient in communication and computation. The proof size is 2λ bits. In the setup, only one group multiplication, one group addition and one pseudorandom function evaluation are required to compute a tag per each data block. Let ℓ be the size of set C in a verification. In each verification, the prover takes 2ℓ group multiplications/additions to generate a proof; the verifier takes ℓ group multiplications/additions and ℓ number of pseudorandom function evaluations to verify the proof.

The drawback is that an authentication tag is as large as a data block, and the storage overhead due to the authentication tags is equal to the file size (after error erasure encoding). In applications where large data redundancy is required, this drawback is mitigated due to the fact that the underlying MAC scheme $\mathsf{S1}$ also functions as a symmetric key encryption key (i.e. a symmetric key signcryption [Zhe97]): All authentication tags together is another copy of the file in the encrypted domain.

4.6 Security Analysis of MAC scheme $\mathsf{S1}$

4.6.1 Security Model for Linearly Homomorphic MAC

Let $\mathsf{S} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Combine}, \mathsf{Verify})$ be a linearly homomorphic MAC scheme as described in Section 4.3. We define the security game $\mathsf{Game}^{\mathsf{CMA}}_{\mathsf{S},\mathcal{A}}$ between a challenger $\mathcal{C}$ and an *existential forgery* adversary $\mathcal{A}$ w.r.t. a linearly homomorphic MAC scheme S, under *adaptive chosen message attack* as below.

Setup.

The challenger $\mathcal{C}$ generates a pair of public-private key (spk, ssk) by running the key generating algorithm $\mathsf{KeyGen}(1^\lambda)$ with security parameter λ, and gives the public key spk to the adversary $\mathcal{A}$ and keeps the private key ssk securely. The challenger $\mathcal{C}$ maintains a state variable state, which is used to assign a unique index to each message to be signed.

Learning.

The adversary $\mathcal{A}$ can adaptively make queries, where each query is in one of the following forms:

- **SignQuery**($\mathtt{M}$): Given a message $\mathtt{M}$ chosen by the adversary $\mathcal{A}$, the challenger $\mathcal{C}$ chooses a unique index i based on the current value of state and updates state. The challenger $\mathcal{C}$ denotes the message as $\mathtt{M}_i$ and responses the query with a signature $\sigma_i := \mathsf{Sign}(ssk, i, \mathtt{M}_i)$.
- **VerifyQuery**($\mathtt{M}, \sigma, \{(i, \nu_i) : i \in C\}$): Let $\mathbf{I}$ denote the set of all indices i's chosen by the challenger in answering all **SignQuery**. For each tuple $(\mathtt{M}, \sigma, \{(i, \nu_i) : i \in C\})$ chosen by the adversary $\mathcal{A}$, if $C \subset \mathbf{I}$, then the challenger $\mathcal{C}$ responses with $b := \mathsf{Verify}(ssk, \mathtt{M}, \sigma, \{(i, \nu_i) : i \in C\}) \in \{\mathtt{accept}, \mathtt{reject}\}$. If $C \not\subset \mathbf{I}$, the challenger does nothing.

Forge.

The adversary $\mathcal{A}$ outputs $(\mathtt{M}', \sigma', \{(i, \nu_i) : i \in C\})$ with $C \subset \mathbf{I}$. Let $(\mathtt{M}, \sigma) := \mathsf{S1.Combine}(spk, \{(i, \mathtt{M}_i, \sigma_i, \nu_i) : i \in C\})$ be the corresponding genuine output. The adversary $\mathcal{A}$ wins the game if and only if

$$\mathsf{Verify}(ssk, \mathtt{M}', \sigma', \{(i, \nu_i) : i \in C\}) \ = \ \mathtt{accept} \text{ and } (\mathtt{M}', \sigma') \neq (\mathtt{M}, \sigma). \tag{4.12}$$

4.6.2 The Linearly Homomorphic MAC Scheme S1 is Secure

Theorem 4.3 (S1 is secure) *Suppose* $\{\mathsf{PRF}_{\mathsf{seed}} : \{0,1\}^{2\lambda} \to \mathbb{Z}_p\}$ *is a secure pseudorandom function family. Then the MAC scheme* $\mathsf{S1}$ *is existentially unforgeable under adaptive chosen message attack. For any PPT adversary* $\mathcal{A}$*, the advantage of*

$\mathcal{A}$ *against* $\mathsf{S1}$ *defined as below is negligible in the security parameter.*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{S1}} \overset{\text{def}}{=} \mathsf{Pr}[\mathcal{A} \textit{ wins } \mathsf{Game}_{\mathsf{S1},\mathcal{A}}^{\mathsf{CMA}}]. \tag{4.13}$$

We start by proving the unforgeability in a simplified setting, where all acceptance or rejection decisions are kept secret from the adversary. That is, the challenger does not answer any $\mathsf{VerifyQuery}$ in the security game. We refer to this simplified setting as *no-feedback* setting, and refer to the original setting as *feedback* setting. Next, we prove the security in feedback setting, based on the security in no-feedback setting.

It is worthy to point out that, we achieve security in feedback setting by "lifting" the security in no-feedback setting, in contrast with verifiable cloud computing [GGP10, CKV10] that achieve security only in no-feedback setting (for a much larger class of delegated functions). We emphasize that this is possible because our scheme $\mathsf{S1}$ has an essential difference with [GGP10, CKV10]: Informally, in $\mathsf{S1}$, we prove that if a candidate aggregated message-MAC pair $(\mathtt{M}, \sigma)$ w.r.t. $\{(i, \nu_i) : i \in C\}$ is accepted by the verifier, then both of $\mathtt{M}$ and σ should be genuine (i.e. equal to the corresponding value computed by an honest user); while in the case of [GGP10,CKV10], their security proof only ensures that the computation result (counterpart of $\mathtt{M}$) is genuine and cannot ensure whether the proof (counterpart of σ) is exactly the one generated by an honest user. In [GGP10,CKV10], indeed anyone with the public key of the fully homomorphic encryption scheme can output many different proofs (for the correct computation result) such that the verifier will accept all of them.

4.6.2.1 $\mathsf{S1}$ is secure in no-feedback setting

Lemma 4.4 *If there exists a PPT adversary* $\mathcal{A}$ *which wins* $\mathsf{Game}_{\mathsf{S1},\mathcal{A}}^{\mathsf{CMA}}$ *in the no-feedback setting with non-negligible probability, then there exists a PPT adversary* $\mathcal{B}$ *which breaks the security of the pseudorandom function* PRF*. Precisely,*

$$\mathsf{Pr}[\mathcal{A} \textit{ wins } \mathsf{Game}_{\mathsf{S1},\mathcal{A}}^{\mathsf{CMA}} \textit{ in no-feedback setting }] \leq \frac{1}{p-1} + N_{\mathsf{PRF}} \cdot \mathsf{Adv}_{\mathcal{B}}^{\mathsf{PRF}},$$

where N_{PRF} is the number of distinct evaluations of pseudorandom function PRF *required to answer all* **SignQuery***s made by $\mathcal{A}$ (one* PRF *evaluation for one* **SignQuery***), and $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{PRF}}$ denotes the probability that $\mathcal{B}$ can distinguish the output of* PRF *from true randomness.*

Proof of Lemma 4.4:

Game 1. The first game is the same as $\mathsf{Game}_{\mathsf{S1},\mathcal{A}}^{\mathsf{CMA}}$, except that the challenger will not answer **VerifyQuery** made by the adversary $\mathcal{A}$. Therefore, $\mathsf{Pr}[\mathcal{A}$ wins **Game 1** $] =$ $\mathsf{Pr}[\mathcal{A}$ wins $\mathsf{Game}_{\mathsf{S1},\mathcal{A}}^{\mathsf{CMA}}$ in no-feedback setting $]$.

Game 2. The second game is the same as **Game 1**, except that in the scheme $\mathsf{S1}$, the pseudorandom function $\mathsf{PRF}_{\mathsf{seed}}(\cdot)$ is replaced by a simulator $\mathsf{PRF}^{\mathsf{Sim}}$ which outputs true randomness over the range of $\mathsf{PRF}_{\mathsf{seed}}$. Precisely, the function $\mathsf{PRF}^{\mathsf{Sim}}$ is evaluated in the following way:

- The challenger keeps a table, which is empty at the very beginning, to store all previous encountered input-output pairs $(v, \mathsf{PRF}^{\mathsf{Sim}}(v))$.
- Given an input v, the challenger lookups the table for v, if there exists an entry (v, u), then return u as output. Otherwise, choose u at random from the range of $\mathsf{PRF}_{\mathsf{seed}}$, insert $(v, \mathsf{PRF}^{\mathsf{Sim}}(v) := u)$ into the table and return u as output.

Claim 4.6.1 *If there is a non-negligible difference in an PPT adversary $\mathcal{A}$'s success probability between* **Game 1** *and* **Game 2***, then there exists another PPT adversary $\mathcal{B}$ which can break the security of the pseudorandom function* PRF*. More precisely,*

$$\left|\mathsf{Pr}[\mathcal{A} \textit{ wins } \textbf{Game 1}] - \mathsf{Pr}[\mathcal{A} \textit{ wins } \textbf{Game 2}]\right| \leq N_{\mathsf{PRF}} \cdot \mathsf{Adv}_{\mathcal{B}}^{\mathsf{PRF}},$$

where N_{PRF} is the number of distinct evaluations of pseudorandom function PRF *required to answer all* **SignQuery***s made by $\mathcal{A}$ (one* PRF *evaluation for one* **SignQuery***), and $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{PRF}}$ denotes the probability that $\mathcal{B}$ can distinguish the output of* PRF *from true randomness.*

The above Claim 4.6.1 can be proved using a standard hybrid argument [Gol06]. Here we save the details.

Claim 4.6.2 *For any computationally unbounded adversary $\mathcal{A}$, after interacting in* **Game 2***, the probability that $\mathcal{A}$ finds the value of τ is* $1/(p-1)$.

Proof of Claim 4.6.2: The secret value τ is only involved in the MAC values σ_i's. Since in **Game 2**, the pseudorandom function PRF is replaced by a simulator $\mathsf{PRF}^{\mathsf{Sim}}$ which outputs *truly* uniform randomness over $\mathbb{Z}_p$, the MAC values σ_i's reveal absolutely *no* information to adversary $\mathcal{A}$ about the secret τ at all, although $\mathcal{A}$ is computationally unbounded. That is, the entropy of τ to the adversary $\mathcal{A}$ is unchanged before and after $\mathcal{A}$'s interaction with the challenger in the **Game 2**, and the probability that $\mathcal{A}$ can find τ is exactly $1/(p-1)$. Recall that τ is chosen uniformly randomly from $\mathbb{Z}_p^*$. $\square$

Claim 4.6.3

$$\mathsf{Pr}[\mathcal{A} \textit{ wins } \textbf{Game 2}] \leq \frac{1}{p-1}. \tag{4.14}$$

Proof of Claim 4.6.3: Recall that $(\mathtt{M}', \sigma', \{(i, \nu_i) : i \in C\})$ denotes the forgery output by the adversary $\mathcal{A}$ in the **Forge** phase of **Game 2**, and $(\mathtt{M}, \sigma, \{(i, \nu_i) : i \in C\})$ denotes the corresponding genuine output[1] which shares the same value $\{(i, \nu_i) : i \in C\}$ with the forgery output. We assume the forgery is valid, that is, the forgery output $(\mathtt{M}', \sigma', \{(i, \nu_i) : i \in C\})$ is accepted by the verifier algorithm $\mathsf{S1.Verify}$, and $(\mathtt{M}', \sigma') \neq (\mathtt{M}, \sigma)$.

Since both the forgery output $(\mathtt{M}', \sigma', \{(i, \nu_i) : i \in C\})$ and the genuine output $(\mathtt{M}, \sigma, \{(i, \nu_i) : i \in C\})$ are accepted by $\mathsf{S1.Verify}$, we substitute them into the Equation (4.7) (on page 34) separately and obtain the below equations:

$$\sum_{i \in C} \nu_i \mathsf{PRF}_{\mathsf{seed}}(i) + \tau \mathtt{M}' = \sigma' \mod p; \tag{4.15}$$

$$\sum_{i \in C} \nu_i \mathsf{PRF}_{\mathsf{seed}}(i) + \tau \mathtt{M} = \sigma \mod p. \tag{4.16}$$

[1]The adversary can compute the genuine output by keeping an intact copy of users' file from the very beginning.

Subtract Equation (4.16) from Equation (4.15), we have

$$\tau(\mathtt{M}' - \mathtt{M}) = \sigma' - \sigma \mod p. \tag{4.17}$$

If $\mathtt{M}' = \mathtt{M}$, then $\sigma' - \sigma = \tau(\mathtt{M}' - \mathtt{M}) = 0 \mod p$, which is a contradiction with our hypothesis that $(\mathtt{M}', \sigma') \neq (\mathtt{M}, \sigma)$. Thus $\mathtt{M}' \neq \mathtt{M}$ and the secret value τ can be found

$$\tau = \frac{\sigma' - \sigma}{\mathtt{M}' - \mathtt{M}} \mod p. \tag{4.18}$$

By the Claim 4.6.2, we conclude

$$\mathsf{Pr}[\mathcal{A} \text{ wins } \textbf{Game 2}] \leq \mathsf{Pr}[\mathcal{A} \text{ finds } \tau \text{ in } \textbf{Game 2}] \leq \frac{1}{p-1}. \tag{4.19}$$

Thus, Claim 4.6.3 is proved. □

Therefore, Lemma 4.4 is inferred by combining Claim 4.6.1 and Claim 4.6.3. □

4.6.2.2 S1 is secure in feedback setting

Lemma 4.5 S1 *is unforgeable in feedback setting.*

Proof of Lemma 4.5:

Game 1. The first game is just $\mathsf{Game}^{\mathsf{CMA}}_{\mathsf{S1},\mathcal{A}}$ in the no-feedback setting, where all acceptance or rejection decisions are kept secret from the adversary. It is identical to the **Game 1** in the proof of Lemma 4.4 (on page 39).

For each integer $k \geq 0$, we define the following game:

Game 2.k. This game is the same as **Game 1**, except that, $\mathcal{A}$ adaptively makes k verification queries in the **Learning** phase and all acceptance or rejection decisions are provided to $\mathcal{A}$ at the end of each query.

We describe two different verification strategies as below, where the first one is adopted by the challenger of the security game **Game 2.k** and the second one serves as the reference:

- **SimulatedVerifier**: The challenger keeps a local copy of messages and MACs, where messages are chosen by the adversary in a **SignQuery** and MAC values are generated by the challenger in response to that **SignQuery**. The challenger also plays the role of an honest user of the MAC scheme $\mathsf{S1}$. For each tuple $(\mathtt{M}', \sigma', \{(i, \nu_i) : i \in C\})$ received from the adversary $\mathcal{A}$ in a **VerifyQuery**, the challenger computes the corresponding genuine tuple $(\mathtt{M}, \sigma, \{(i, \nu_i) : i \in C\})$ from the challenger's local copy of messages and MACs. If adversary's output are the same as the genuine output, i.e. $(\mathtt{M}', \sigma') = (\mathtt{M}, \sigma)$, then outputs `accept`; otherwise outputs `reject`.

- **ImaginaryVerifier**: An imaginary verification oracle $\mathcal{O}^{\mathsf{S1.Verify}(ssk;\cdot)}$ which somehow has access to the private key ssk.

Note that (1) the simulated verifier accepts only genuine output while the imaginary verifier oracle accepts all valid outputs which include genuine outputs; (2) the simulated verifier provides absolutely *no* new information to the adversary $\mathcal{A}$, since $\mathcal{A}$ itself can simulate such verifier by keeping another intact copy of the messages and MAC values from the very beginning.

Let us code `accept` with the bit '1' and code `reject` with the bit '0', and denote with $a_i \in \{0, 1\}$ be the decision bit output by the imaginary verification oracle $\mathcal{O}^{\mathsf{S1.Verify}(ssk;\cdot)}$ for the i-th **VerifyQuery** made by the adversary $\mathcal{A}$ in **Game 2.k**; $b_i \in \{0, 1\}$ be the corresponding decision bit output by the simulated verifier. Furthermore, let $A_k := a_1 a_2 \ldots a_k \in \{0, 1\}^k$ and $B_k := b_1 b_2 \ldots b_k \in \{0, 1\}^k$. We notice that

- $a_{k+1} \neq b_{k+1}$ indicates the event that the adversary wins **Game 2.k**.
  - $(a_{k+1} = 1, b_{k+1} = 0)$ indicates the event that the adversary wins **Game 2.k**, since the adversary's output is valid (accepted by **ImaginaryVerifier**), but different from the genuine output (rejected by **SimulatedVerifier**).
  - $(a_{k+1} = 0, b_{k+1} = 1)$ is impossible, since the MAC scheme $\mathsf{S1}$ is correct such that all genuine MAC values are valid.
- $a_{k+1} = b_{k+1}$ indicates the event that the adversary loses **Game 2.k**.

Claim 4.6.4 *Let ξ be a negligible function implied in Lemma 4.4, such that for any PPT adversary $\mathcal{A}$, $\mathsf{Pr}[\mathcal{A}$ wins **Game 1**$] \le \xi$. Then $\mathsf{Pr}[\mathcal{A}$ wins **Game 2.0**$] \le \xi$.*

Claim 4.6.5 *If $\mathsf{Pr}[A_k = B_k] \ge X$, then $\mathsf{Pr}[A_{k+1} = B_{k+1}] \ge X(1-\xi)$.*

Proof of Claim 4.6.5:

$$\begin{aligned}
\mathsf{Pr}[A_{k+1} = B_{k+1}] &= \mathsf{Pr}[A_k = B_k \wedge a_{k+1} = b_{k+1}] && (4.20)\\
&= \mathsf{Pr}[A_k = B_k] \times \mathsf{Pr}[a_{k+1} = b_{k+1} \mid A_k = B_k] && (4.21)\\
&\ge \mathsf{Pr}[A_k = B_k] \times \mathsf{Pr}[\mathcal{A} \text{ loses } \textbf{Game 1}] && (4.22)\\
&\ge X(1-\xi). && (4.23)
\end{aligned}$$

□

Claim 4.6.6 $\mathsf{Pr}[A_k = B_k] \ge (1-\xi)^k$.

Proof of Claim 4.6.6: We prove the above claim using mathematical induction.
Base Case: $k = 1$. $\mathsf{Pr}[A_1 = B_1] = \mathsf{Pr}[a_1 = b_1] = \mathsf{Pr}[\mathcal{A}$ loses **Game 2.0**$] \ge (1-\xi)$.
Induction Step: from k to $k+1$. This is just Claim 4.6.5. □

Claim 4.6.7 $\mathsf{Pr}[\mathcal{A}$ *wins* **Game 2.k**$] \le \xi + \big(1-(1-\xi)^k\big) = (k+1)\xi + o(\xi)$.

Notice that here $o(\cdot)$ denote the little-O notation.

Proof of Claim 4.6.7:

$$\begin{aligned}
&\mathsf{Pr}[\mathcal{A} \text{ wins } \textbf{Game 2.k}]\\
=\ &\mathsf{Pr}[\mathcal{A} \text{ wins } \textbf{Game 2.k} \wedge A_k = B_k] \ + \ \mathsf{Pr}[\mathcal{A} \text{ wins } \textbf{Game 2.k} \wedge A_k \ne B_k]\\
\le\ &\mathsf{Pr}[\mathcal{A} \text{ wins } \textbf{Game 2.k} \mid A_k = B_k] \times \mathsf{Pr}[A_k = B_k] \ + \ \mathsf{Pr}[A_k \ne B_k]\\
\le\ &\mathsf{Pr}[\mathcal{A} \text{ wins } \textbf{Game 2.k} \mid A_k = B_k] \ + \ \mathsf{Pr}[A_k \ne B_k]\\
\le\ &\mathsf{Pr}[\mathcal{A} \text{ wins } \textbf{Game 1}] \ + \ \mathsf{Pr}[A_k \ne B_k]\\
\le\ &\xi + \big(1-(1-\xi)^k\big) = (k+1)\xi + o(\xi).
\end{aligned}$$

Notice that $\mathsf{Pr}[\mathcal{A}$ wins **Game 2.k** $\mid A_k = B_k] \leq \mathsf{Pr}[\mathcal{A}$ wins **Game 1**], since in **Game 2.k**, $A_k = B_k$ indicates that the adversary gains absolutely no information from the k **VerifyQuery** in the **Learning** phase. □

Therefore, Lemma 4.5 is concluded from Claim 4.6.7. □

Now it is time to prove the Theorem 4.3.

Proof of Theorem 4.3: By Lemma 4.4 and Lemma 4.5, for any PPT adversary $\mathcal{A}$ which makes N_{PRF} number of **SignQuery** and $N_{\mathtt{verify}}$ number of **VerifyQuery** in the security game $\mathsf{Game}^{\mathsf{CMA}}_{\mathsf{S1},\mathcal{A}}$, the probability that $\mathcal{A}$ wins the security game is

$$\mathsf{Pr}[\mathcal{A} \text{ wins } \mathsf{Game}^{\mathsf{CMA}}_{\mathsf{S1},\mathcal{A}}] \leq (N_{\mathtt{verify}} + 1)\left(\frac{1}{p-1} + N_{\mathsf{PRF}} \cdot \mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{B}}\right),$$

which is negligible in the security parameter $\lambda = \log p$. □

4.7 Security Analysis of $\mathcal{POR}$ scheme POS1

Theorem 4.6 *If $\{\mathsf{PRF}_{\mathsf{seed}} : \{0,1\}^{2\lambda} \to \mathbb{Z}_p\}$ is a secure pseudorandom function family, then the proposed $\mathcal{POR}$ scheme* POS1 *is sound under Definition 6 (on page 26).*

We first prove two lemmas about random sampling and then apply these two lemmas, together with the security property of S1, to prove the above theorem.

4.7.1 Two Lemmas on Random Sampling

Lemma 4.7 *Let $\mathbb{X}$ and $\mathbb{Y}$ be two finite sets. Let $U_{\mathbb{X}}$ denote a uniform random variable over the domain $\mathbb{X}$ and $U_{\mathbb{Y}}$ denote a (independent) uniform random variable over the domain $\mathbb{Y}$. Consider any function $f : \mathbb{X} \times \mathbb{Y} \to \{0,1\}$. Let $\epsilon = \mathsf{Pr}[f(U_{\mathbb{X}}, U_{\mathbb{Y}}) = 1]$. For any constant $a \in (0, \frac{1}{2})$, define a set $\mathbf{S}_a = \{x \in \mathbb{X} : \mathsf{Pr}[f(x, U_{\mathbb{Y}}) = 1] \geq a\epsilon\}$. We have*

$$\mathsf{Pr}[U_{\mathbb{X}} \in \mathbf{S}_a] = \frac{|\mathbf{S}_a|}{|\mathbb{X}|} \geq \left(\frac{1-a}{\frac{1}{\epsilon} - a}\right) = \mathcal{O}(\epsilon).$$

Proof of Lemma 4.7: Let $N = |\mathbf{S}_a|$ be the size of set $\mathbf{S}_a$. We have

$$\begin{aligned}
\epsilon &= \mathsf{Pr}[f(U_{\mathbb{X}}, U_{\mathbb{Y}}) = 1] && (4.24)\\
&= \frac{1}{|\mathbb{X}|}\left(\sum_{x\in\mathbf{S}_a}\mathsf{Pr}[f(x,U_{\mathbb{Y}})=1] \;+\; \sum_{x\in\mathbb{X}\setminus\mathbf{S}_a}\mathsf{Pr}[f(x,U_{\mathbb{Y}})=1]\right) && (4.25)\\
&\le \frac{1}{|\mathbb{X}|}\left(N + a\epsilon(|\mathbb{X}|-N)\right) && (4.26)\\
\Rightarrow \quad & N \ge |\mathbb{X}|\left(\frac{1-a}{\frac{1}{\epsilon}-a}\right) && (4.27)\\
\Rightarrow \quad & \frac{N}{|\mathbb{X}|} \ge \left(\frac{1-a}{\frac{1}{\epsilon}-a}\right) \in (\frac{1}{2}\epsilon,\ \epsilon] = \mathcal{O}(\epsilon). && (4.28)
\end{aligned}$$

□

Lemma 4.8 *Let κ be an integer. Let $\delta, \epsilon \in (0,1]$ be two real values and $\delta \ge \epsilon$. Let $t = \lceil \frac{\kappa}{\epsilon} \rceil$. Independently sample t number of values $r_1, \ldots, r_t$ from $\{0,1\}$ under the Bernoulli distribution with probability δ. Let d be a positive integer and $d \le \kappa^c$ for some real valued constant $c \in (0,1)$. Then with overwhelming high probability (w.r.t. κ), there exists d distinct indices $i_1, i_2, \ldots, i_d \in [1,t]$ such that $\forall j \in [1,d], r_{i_j} = 1$.*

We remark that the original form of Chernoff bound [Che52, Gol06] does not serve our purpose.

Proof of Lemma 4.8: Let us consider the set $\mathbf{S} = \{i \in [1,t] : r_i = 1\}$. We will show that the size of set $\mathbf{S}$ is at least d with overwhelming high probability.

For each index $i \in [1, t]$, we have $\mathsf{Pr}[r_i = 1] = \delta \geq \epsilon$.

$$\begin{aligned}
\mathsf{Pr}[|\mathbf{S}| < d] &= \sum_{j=0}^{d-1} \mathsf{Pr}[|\mathbf{S}| = j] = \sum_{j=0}^{d-1} \binom{t}{j} \delta^j (1-\delta)^{t-j} && (4.29)\\
&\leq \sum_{j=0}^{d-1} \binom{t}{j} \epsilon^j (1-\epsilon)^{t-j} && (4.30)\\
&\leq \sum_{j=0}^{d-1} \binom{t}{d-1} \epsilon^{d-1} (1-\epsilon)^{t-d+1} && (4.31)\\
&\leq d(\kappa+1)^{d-1} e^{-\kappa+\mathcal{O}(\kappa^c)} && (4.32)\\
&\leq \frac{\kappa^c}{e^{\mathcal{O}(\kappa)}} && (4.33)
\end{aligned}$$

where e is the base of natural logarithm.

Now we explain the above inference. Equation (4.30) can be derived from Equation (4.29), since for each $j \in \{0, 1, 2, \ldots, d-1\}$, function $f(\delta) = \delta^j (1-\delta)^{t-j}, \delta \in (0, 1)$, has a negative first order derivative since $\kappa > \kappa^c \geq d$, and is thus a monotone decreasing function of $\delta \in (0, 1)$.

Equation (4.31) can be derived from Equation (4.30), since the probability mass function of a binomial distribution is a bell shape (like normal distribution) and reach its maximum at $j = t \cdot \epsilon \geq \kappa > d$. That is, for $0 \leq j < d < t \cdot \epsilon$, the function $g(j) = \binom{t}{j} (\epsilon)^j (1-\epsilon)^{t-j}$ is a monotone increasing function of j.

Equation (4.32) can be derived from Equation (4.31), due to the following two Equations (4.34) and (4.35).

$$\begin{aligned}
(1-\epsilon)^{t-d+1} &= (1-\epsilon)^{\frac{1}{\epsilon} \times (\epsilon t + \epsilon(-d+1))} \leq (1-\epsilon)^{\frac{1}{\epsilon} \times (\kappa - \epsilon \kappa^c)}\\
&< \left(\frac{1}{e}\right)^{\kappa - \mathcal{O}(\kappa^c)} = e^{-\kappa + \mathcal{O}(\kappa^c)}. && (4.34)
\end{aligned}$$

$$\binom{t}{d-1} \epsilon^{d-1} \leq t^{d-1} \epsilon^{d-1} = (t\epsilon)^{d-1} \leq (\kappa+1)^{d-1} \qquad (4.35)$$

Equation (4.33) can be derived from Equation (4.32), since

$$\frac{(\kappa+1)^{d-1}}{e^{\kappa-\mathcal{O}(\kappa^c)}} \leq \frac{(\kappa+1)^{(\kappa^c-1)}}{e^{\kappa-\mathcal{O}(\kappa^c)}} = e^{(\kappa^c-1)\ln(\kappa+1)-\kappa+\mathcal{O}(\kappa^c)} = e^{-\mathcal{O}(\kappa)}. \tag{4.36}$$

So $de^{-\mathcal{O}(\kappa)} \leq \kappa^c \cdot e^{-\mathcal{O}(\kappa)}$ is negligible in κ. Recall that $d < \kappa^c$ for constant $c \in (0,1)$.

Therefore, $\mathsf{Pr}[|\mathbf{S}| < d]$ is negligible in κ and the set $\mathbf{S}$ has size at least d with overwhelming high probability $(1 - \mathsf{Pr}[|\mathbf{S}| < d])$ in κ. □

4.7.2 Scheme POS1 is Sound

Proof of Theorem 4.6: Suppose in the **Retrieve** phase of the $\mathcal{POR}$ security game between an adversary $\mathcal{A}$ and a challenger, the challenger initiates $\zeta = \mathcal{O}(t^2)$ verifications, and the adversary $\mathcal{A}$ correctly answers ϵ fraction of verification queries, where $t = \lceil \frac{n\lambda}{\epsilon} \rceil$. Here n is the number of blocks in the error erasure encoded file.

Recall that a challenge query $(C, \vec{\boldsymbol{\nu}})$ consists of two parts: a subset $C \subset [1,n]$ with size ℓ, and a weight vector $\vec{\boldsymbol{\nu}} = (\nu, \nu^2, \ldots, \nu^{d+1}) \in (\mathbb{Z}_p)^\ell$. Let us denote the domain of C as $\mathbb{C}$ and the domain of weight as $\mathbb{W}$.

We remark that (1) a negligible function w.r.t. $n\lambda$ is also a negligible function w.r.t. λ; (2) both sizes of domain $\mathbb{C}$ and $\mathbb{W}$ are exponentially large in λ, since $\ell = \mathcal{O}(\lambda)$.

Extractor Strategy The challenger chooses challenge queries in this way: For each $i \in [1,t]$, independently choose a random subset $C_i \in \mathbb{C}$, and independently choose t number of random weights $\vec{\boldsymbol{\nu}}_{i,j} \in \mathbb{W}, j \in [1,t]$. For each $(i,j) \in [1,t] \times [1,t]$, send $(C_i, \vec{\boldsymbol{\nu}}_{i,j})$ to the adversary $\mathcal{A}$, receive response $(\mathtt{M}_{i,j}, \sigma_{i,j})$ from the adversary. Define a function $f : \mathbb{C} \times \mathbb{W} \to \{0,1\}$, such that if $(\mathtt{M}_{i,j}, \sigma_{i,j})$ is accepted, then $f(C_i, \vec{\boldsymbol{\nu}}_{i,j}) = 1$, otherwise $f(C_i, \vec{\boldsymbol{\nu}}_{i,j}) = 0$.

The challenger finds all indices $i \in [1,t]$, such that for at least ℓ distinct indices $j_\iota, \iota \in [1,\ell]$, the adversary $\mathcal{A}$'s responses $(\mathtt{M}_{i,j_\iota}, \sigma_{i,j_\iota})$ w.r.t. query $(C_i, \vec{\boldsymbol{\nu}}_{i,j_\iota})$ are accepted. From such responses $\mathtt{M}_{i,j_\iota}$, the challenger can solve the following linear equation system to obtain the data blocks $\mathtt{F}_u$'s with index u in the set C_i. Let

$\vec{\mathtt{M}} = (\mathtt{M}_{i,j_1}, \ldots, \mathtt{M}_{i,j_\ell})^\top$ be a column vector and $\boldsymbol{\mu}$ be a $\ell \times \ell$ matrix, such that ι-th row of $\boldsymbol{\mu}$ is just the row vector $\vec{\boldsymbol{\nu}}_{i,j_\iota}$, $\iota \in [1, \ell]$. The linear equation system with unknown $\vec{\boldsymbol{x}}$ is as below

$$\boldsymbol{\mu}\vec{\boldsymbol{x}} = \vec{\mathtt{M}} \tag{4.37}$$

Note that $\boldsymbol{\mu}$ is a *vandermonde matrix* [MS58] and has rank ℓ as long as all $\vec{\boldsymbol{\nu}}_{i,j_\iota}$'s, $\iota \in [1, \ell]$, are distinct.

From ρn number of blocks $\mathtt{F}_u$'s, the original file can be recovered using the error erasure decoding algorithm. Recall that ρ is the rate of the error erasure code.

Analysis of the Extractor Strategy Let $a = \frac{1}{3}$. For a subset $C \in \mathbb{C}$, we say C is *good*, if the $\mathsf{Pr}[f(C, \vec{\boldsymbol{\nu}}) = 1] \geq a\epsilon$, where the probability is over uniformly randomly chosen weight vector $\vec{\boldsymbol{\nu}} \in \mathbb{W}$. Let $\mathbf{S}_a$ be the set of all *good* subset C. By applying Lemma 4.7, a uniformly random chosen subset C is *good* with probability at least $\mathcal{O}(\epsilon)$: $\mathsf{Pr}[C \in \mathbf{S}_a] \geq \mathcal{O}(\epsilon)$. By Lemma 4.8 with $\kappa = n\lambda$ and constant $d = \rho n < \kappa$, among these t number of independent samples C_i's, $i \in [1, t]$, with overwhelming high probability, there exists at least $d = \rho n$ number of *good* C_i's. For each *good* C_i, applying the Lemma 4.8 on the random variable $\vec{\boldsymbol{\nu}}$ with $\kappa = n\lambda$ and $d = \ell < \kappa$, with overwhelming high probability, there exists at least d samples $\vec{\boldsymbol{\nu}}_{i,j_\iota}, \iota \in [1, d], j_\iota \in [1, t]$, such that $f(C_i, \vec{\boldsymbol{\nu}}_{i,j_\iota}) = 1$ (We call such $\vec{\boldsymbol{\nu}}_{i,j_\iota}$ as *good* weight). Therefore, with overwhelming high probability, the challenger can find sufficient number of good subsets and weight vectors to form ρn number of linear equation systems as in Equation (4.37).

Since the underlying MAC scheme $\mathsf{S1}$ is unforgeable (Theorem 4.3), a valid proof $(\mathtt{M}, \sigma)$ is genuine with overwhelming high probability. Additionally, since the domain of subset C_i (respectively, weights $\vec{\boldsymbol{\nu}}_{i,j_\iota}$) is exponentially large w.r.t. λ, all good C_i's (respectively, good weights $\vec{\boldsymbol{\nu}}_{i,j_\iota}$) are distinct with overwhelming high probability. Thus the challenger's above extractor algorithm succeeds with overwhelming high probability. This completes the proof of Theorem 4.6. □

We remark that the above proof assumes that the adversary's response w.r.t. a given particular query is deterministic for simplicity. In the more generic case where the adversary's response to a given query is probabilistic, we can write out the adversary's random coin explicitly and apply the Lemma 4.8 on this random coin for one more time. After that the above argument still holds. The cost is that in this more generic case, the extractor runs $\mathcal{O}(t^3)$ number of verification interactions.

4.8 Summary

This chapter presented a linearly homomorphic MAC scheme $\mathsf{S1}$, which allows any third party to compute a MAC value for a linear combination of messages $\mathtt{M}_i$'s, from the MAC values of these messages $\mathtt{M}_i$'s and a public key. $\mathsf{S1}$ is also a symmetric key signcryption scheme, that is, it functions as a symmetric key signature (i.e. MAC) scheme and a symmetric encryption scheme simultaneously. Based on $\mathsf{S1}$, a Proofs of Retrievability scheme $\mathsf{POS1}$ is constructed. The $\mathcal{POR}$ scheme $\mathsf{POS1}$ is very efficient in communication and computation: (1) only linear (w.r.t. the security parameter) communication bits per verification are required; and (2) only group multiplication/addition and pseudorandom function evaluations are required in data preprocess and proof generation. The drawback is that the authentication tags are as large as the data file size. Full security proof of $\mathsf{S1}$ and $\mathsf{POS1}$ under assumption of existence of secure pseudorandom function are provided. The security property of $\mathsf{S1}$ is proved in two steps: (1) in the first step, security of $\mathsf{S1}$ is proved in a simplified setting, where all acceptance/rejection decisions of each verification are kept secret from the adversary; (2) in the second step, security of $\mathsf{S1}$ is proved in the original setting, using the result in the first step.

Chapter 5

Proofs of Retrievability from Linearly Predicate-Homomorphic Message Authentication Code

A *linearly predicate-homomorphic* MAC scheme satisfies two homomorphic properties:

- It is a linearly homomorphic MAC scheme as described in previous Chapter 4. That is, any third party can compute a MAC value for a linear combination of messages $\mathtt{M}_i$'s from the MAC values of these messages $\mathtt{M}_i$'s, using the public key.
- It is a predicate-homomorphic MAC scheme, such that given a MAC value of a message $\mathtt{M}$, any third party can compute a MAC value for another message $\mathtt{M}'$ with the public key, as long as the two messages $(\mathtt{M}, \mathtt{M}')$ satisfies the designated predicate.

This chapter will propose a linearly predicate-homomorphic MAC scheme, which we refer to as $\mathsf{S2}$, and apply it to construct a proofs of storage scheme, which we refer to as $\mathsf{POS2}$ in this dissertation. The proposed scheme $\mathsf{POS2}$ is very efficient in communication and storage, and is practical in computation. The result in this chapter is published in Xu and Chang [XC12].

5.1 Overview

5.1.1 A Brief Description of proofs of storage scheme $\mathsf{POS2}$

The $\mathcal{POR}$ scheme $\mathsf{POS1}$ constructed in the previous Chapter 4 is very efficient in communication—only a single message-MAC pair is returned as the response in a verification. However, in $\mathsf{POS1}$, the size of authentication tags are equally large as the size of data file (after erasure encoding). Can we reduce the size of authentication data?

Asymmetric Linear Homomorphic MAC—A MAC value is shorter than a message Suppose a data block is large and consists of m sectors. We wish to build an authentication tag with size equal to one sector. This can be done in this way: Choose a secret random vector $\vec{\boldsymbol{s}} \in (\mathbb{Z}_p)^m$. Treat a data block $\vec{\mathsf{F}}_i \in (\mathbb{Z}_p)^m$ as a vector. Compute the inner product $\left\langle \vec{\mathsf{F}}_i,\ \vec{\boldsymbol{s}} \right\rangle$ of the two vectors $\vec{\mathsf{F}}_i$ and $\vec{\boldsymbol{s}}$, and then apply the MAC scheme $\mathsf{S1}$ on the inner product to generate a MAC value σ_i as below,

$$\sigma_i := \mathsf{PRF}_{\mathsf{seed}}(i) \ + \ \tau \left\langle \vec{\mathsf{F}}_i,\ \vec{\boldsymbol{s}} \right\rangle \mod p \tag{5.1}$$

where (seed, τ) is the private key of the MAC scheme $\mathsf{S1}$. The generated MAC values σ_i's are also linearly homomorphic w.r.t. the data blocks $\vec{\mathsf{F}}_i$'s, in the sense that, for any weights $\nu_i \in \mathbb{Z}_p$

$$\sum_i \nu_i \sigma_i := \sum_i \nu_i \mathsf{PRF}_{\mathsf{seed}}(i) \ + \ \tau \left\langle \sum_i \nu_i \vec{\mathsf{F}}_i,\ \vec{\boldsymbol{s}} \right\rangle \mod p \tag{5.2}$$

The linear combination $\sum_i \nu_i \vec{\mathsf{F}}_i$ and its MAC value $\sum_i \nu_i \sigma_i$ have to be returned to the verifier for validity check. However, the size of block $\sum_i \nu_i \vec{\mathsf{F}}_i \pmod p$ is $m\lambda$ bits and the proof size increases from 2λ bits to $(m+1)\lambda$ bits. In summary, the size of all authentication tags is reduced by a multiplicative factor m, at the cost of increase in proof size by a multiplicative factor $\frac{m+1}{2}$. We remark that the above is the essential idea of Shacham and Waters [SW08a].

Predicate Homomorphism Can we *recursively* apply some Proofs of Retrievability scheme on the data $\sum_i \nu_i \vec{\mathtt{F}}_i$ with authentication tag $\sum_i \nu_i \sigma_i$, although both the data and the authentication tag are generated dynamically?

A wishful thought is that: Suppose somehow our MAC scheme supports an additional homomorphism, such that given a message-MAC pair $(\mathtt{M}, \sigma)$, anyone can produce a valid MAC value σ' for another *shorter* message $\mathtt{M}'$, as long as $(\mathtt{M}', \mathtt{M})$ satisfy some designated predicate. Thus the shorter message-MAC pair $(\mathtt{M}', \sigma')$ can be sent back as response for verification, instead of the original $(\mathtt{M}, \sigma)$. Furthermore, there are many different shorter messages $\mathtt{M}'_i$s such that $(\mathtt{M}'_i, \mathtt{M})$ satisfy the designated predicate, and from these shorter messages $\mathtt{M}'_i$s, the original long message $\mathtt{M}$ can be recovered (or retrieved) efficiently. In other words, $\mathtt{M}'_i$s are codewords of $\mathtt{M}$.

Illustration Picture Figure 5.1 illustrates the above idea. In Figure 5.1, each rectangle consists of three squares. Each rectangle represents a data block and each square within a rectangle represents a sector in that data block. Thus in the case of Figure 5.1 the block size is $m = 3$. Those shaded rectangles represent data blocks that are generated due to the error erasure code. A rectangle represents a data block and the circle that lies just below represents the corresponding authentication tag generated by some underlying MAC scheme. An authentication tag is about the same size of one data sector, and thus is one third of a data block. During a verification, a subset of three data blocks and their authentication tags are selected, from which a single pair of data block and an authentication tag is generated by applying the linear homomorphism of the underlying MAC scheme. Instead of sending the large data block back directly, some *predicate homomorphism* is applied to generate a shorter message-MAC pair, which is sent back to the verifier as response.

5.1.2 Organization

The rest of this chapter is organized as below: The next Section 5.2 presents the definition of a linearly predicate-homomorphic MAC scheme, followed by a construction (named $\mathsf{S2}$) of such a homomorphic MAC scheme in Section 5.3. In Section 5.4, a proofs of storage scheme (named $\mathsf{POS2}$) is constructed using the newly constructed

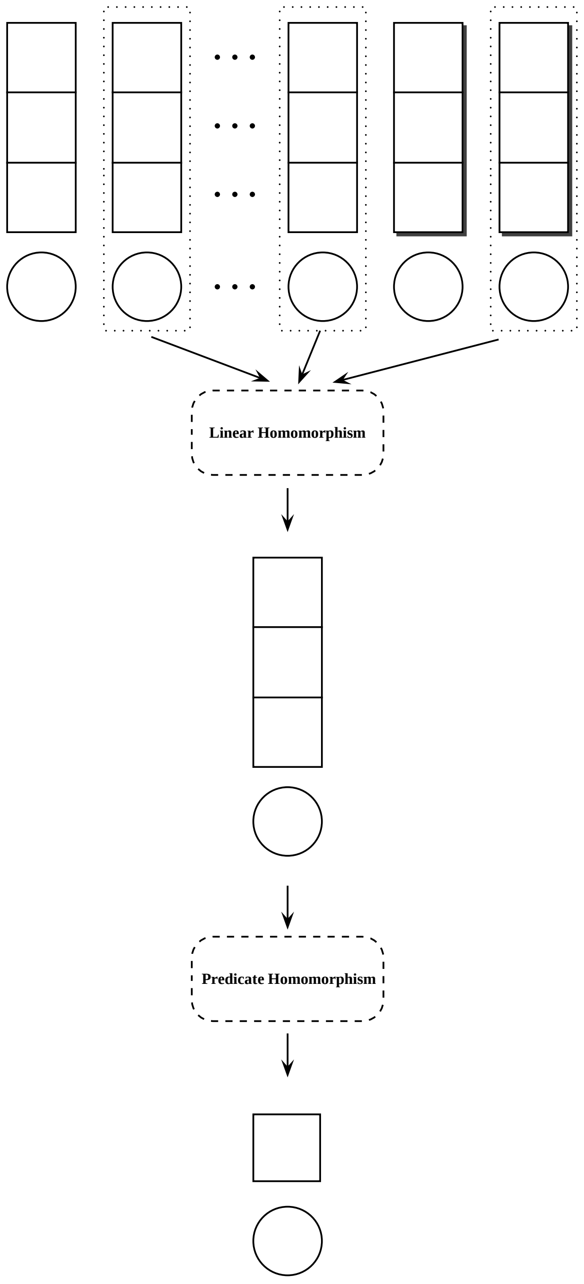


Figure 5.1: $\mathcal{POR}$ scheme $\mathsf{POS2}$ constructed from linearly predicate-homomorphic MAC scheme $\mathsf{S2}$. Detailed explanation is in the paragraph with title “Illustration Picture”.

homomorphic MAC scheme. The performance of $\mathsf{POS2}$ is analyzed in Section 5.5. Next, we present the security formulation of linearly predicate-homomorphic MAC and provide the full security proof for our construction S2 in Section 5.6. The security proof for scheme POS2 under $\mathcal{POR}$ security formulation is in Section 5.7. At the last, Section 5.8 summarizes this chapter.

5.2 Linearly Predicate-Homomorphic MAC: Definition

In this section, we describe the definition of a linearly predicate-homomorphic Message Authentication Code scheme. We defer the security formulation later to Section 5.6, where we will analyze the security property of the proposed construction of such homomorphic MAC scheme.

Ahn *et al.* [ABC$^+$11a] proposed the new concept of "Predicate-Signature", which can be viewed as a counterpart of predicate encryption [BW07, KSW08, SW08b]. In this dissertation, we are interested in a small class of predicates—function.

Let $F : \mathbb{M} \times \mathbb{R} \to \mathbb{M}'$ be a two inputs function. We are interesting in the following predicate w.r.t. function F:

$$P_{F,r}(\mathtt{M}', \mathtt{M}) = \begin{cases} 1 & (F(\mathtt{M}, r) = \mathtt{M}') \\ 0 & (\text{o.w.}) \end{cases} \tag{5.3}$$

A linearly predicate-homomorphic MAC scheme w.r.t. a predicate family $\{P_{F,r} : r \in \mathbb{R}\}$ consists of six algorithms ($\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Combine}, \mathsf{P\text{-}Sign}, \mathsf{Verify}, \mathsf{P\text{-}Verify}$), where each algorithm is described as below.

- $\mathsf{KeyGen}(1^\lambda) \to (spk, ssk)$: The probabilistic key generating algorithm takes the security parameter λ as input, and outputs a pair of public and private keys (spk, ssk).
- $\mathsf{Sign}(ssk, i, \mathtt{M}) \to \sigma$: The signing algorithm takes the private key ssk, an index i, and a message $\mathtt{M} \in \mathbb{M}$ as input, and outputs a signature σ.
- $\mathsf{Combine}(spk, \{(i, \mathtt{M}_i, \sigma_i, \nu_i) : i \in C\}) \to (\mathtt{M}, \sigma)$: The algorithm $\mathsf{Combine}$ implements the linearly homomorphic property. Taking as input the public key spk,

a sequence of tuples $(i, \mathtt{M}_i, \sigma_i, \nu_i)$, $i \in C$, where σ_i is the MAC value of a message $\mathtt{M}_i$ w.r.t. an index i, and ν_i is a weight of $\mathtt{M}_i$ in a linear combination, the algorithm $\mathsf{Combine}$ outputs a message-MAC pair $(\mathtt{M} := \sum_{i \in C} \nu_i \mathtt{M}_i,\ \sigma)$.

- $\mathsf{P\text{-}Sign}(spk, \mathtt{M}, \sigma, r) \rightarrow (F(\mathtt{M}, r), \hat{\sigma})$: This algorithm implements the predicate-homomorphic property. Algorithm $\mathsf{P\text{-}Sign}$ takes the public key spk, a message-MAC pair $(\mathtt{M}, \sigma)$ and a value r as input, and outputs a message-MAC pair $(F(\mathtt{M}, r), \hat{\sigma})$.
  Note: In order to construct an efficient Proofs of Retrievability scheme using such predicate homomorphism, we wish that (1) $\hat{\sigma}$ is a valid MAC value for $F(\mathtt{M}, r)$; (2) the bit-length of $(F(\mathtt{M}, r), \hat{\sigma})$ is sublinear in the bit-length of $(\mathtt{M}, \sigma)$; (3) $F(\mathtt{M}, r)$ is a codeword of $\mathtt{M}$ for any value r from a proper domain $\mathbb{R}$, such that the message $\mathtt{M}$ can be decoded (or retrieved) efficiently from a subset of $\{F(\mathtt{M}, r) : r \in \mathbb{R}\}$.

- $\mathsf{Verify}(ssk, \mathtt{M}, \sigma, \{(i, \nu_i) : i \in C\}) \rightarrow$ `accept` or `reject`: The deterministic verification algorithm Verify takes the private key ssk, a message-MAC pair $(\mathtt{M}, \sigma)$ and a sequence of pairs of index i and weight ν_i, $i \in C$ as input, and outputs either `accept` or `reject`.

- $\mathsf{P\text{-}Verify}(ssk, \mathtt{M}, \sigma, \{(i, \nu_i) : i \in C\}, r) \rightarrow$ `accept` or `reject`: The deterministic predicate-verification algorithm $\mathsf{P\text{-}Verify}$ takes the private key ssk, a message-MAC pair $(\mathtt{M}, \sigma)$, a sequence of pairs of index i and weight ν_i, $i \in C$, and a value r as input, and outputs either `accept` or `reject`.

Intuitively, a MAC scheme is linearly homomorphic, if the MAC value of a linear combination of messages $\mathtt{M}_i$ can be generated without the private key from the MAC values of these messages $\mathtt{M}_i$'s; a MAC scheme is predicate-homomorphic, if the MAC value of a message $\mathtt{M}'$ can be generated without the private key from the MAC value of another message $\mathtt{M}$ as long as the two messages $(\mathtt{M}', \mathtt{M})$ satisfy the designated predicate.

Definition 8 (Linearly Predicate-Homomorphic Message Authentication Code) *Let* $\mathsf{S} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{P\text{-}Sign}, \mathsf{Verify}, \mathsf{P\text{-}Verify})$ *be a MAC scheme. We say* S *is a* linearly predicate-homomorphic MAC *scheme, if the following conditions hold:*

- Linear Homomorphism*: If for any public-private key pair* $(spk, ssk) := \mathsf{KeyGen}(1^\lambda)$ *generated by the key generating algorithm* KeyGen *and for any sequence of message-MAC pairs* $\{(i, \mathtt{M}_i, \sigma_i), i \in C\}$ *generated by the signing algorithm* Sign *under the private key ssk, the output* $(\sum_{i\in C} \nu_i \mathtt{M}_i,\ \sigma) := \mathsf{Combine}(spk,\ \{(i, \mathtt{M}_i, \sigma_i, \nu_i) : i \in C\})$ *is accepted as a valid message-MAC pair by the verification algorithm* Verify *under the private key ssk w.r.t.* $\{(i, \nu_i) : i \in C\}$*.*

- Predicate Homomorphism*: If for any public-private key pair* $(spk, ssk) := \mathsf{KeyGen}(1^\lambda)$ *generated by the key generating algorithm* KeyGen*, and for any message-MAC pair* $(\mathtt{M}, \sigma) := \mathsf{Combine}(spk,\ \{(i, \mathtt{M}_i, \mathsf{Sign}(ssk, i, \mathtt{M}_i), \nu_i) : i \in C\})$ *generated by the algorithm* $\mathsf{Combine}$ *under the public key spk w.r.t.* $\{(i, \nu_i) : i \in C\}$*, and for any value* $r \in \mathbb{R}$*, the output* $(F(\mathtt{M}, r), \hat{\sigma}) := \mathsf{P\text{-}Sign}(spk, \mathtt{M}, \sigma, r)$ *is accepted as a valid message-MAC pair by the verification algorithm* $\mathsf{P\text{-}Verify}$ *under the private key ssk w.r.t.* $\{(i, \nu_i) : i \in C\}$*.*

5.3 Linearly Predicate-Homomorphic MAC: Construction

We first briefly review the polynomial commitment scheme proposed by Kate and Zaverucha and Goldberg [KZG10]. Next, we present our construction of linearly predicate-homomorphic MAC, which integrates the linearly homomorphic MAC scheme $\mathsf{S1}$ presented in Chapter 4 and the idea of polynomial commitment scheme [KZG10].

5.3.1 Background on Short Polynomial Commitment Scheme

Kate and Zaverucha and Goldberg [KZG10] proposed a constant size commitment scheme [Blu81, BCC88] for polynomial functions, which has a special feature in supporting efficient verification of polynomial evaluation. Their scheme exploits a simple and elegant algebraic property of polynomials: *For any polynomial* $f(x) \in \mathbb{Z}_p[x]$ *in variable* x *with coefficients in group* $\mathbb{Z}_p$*, and for any scalar input* $r \in \mathbb{Z}_p$*, the polynomial* $x - r$ *divides the polynomial* $f(x) - f(r)$*.*

Let us denote with $f_{\vec{\boldsymbol{u}}}(x) \in \mathbb{Z}_p[x]$ the polynomial with coefficient vector $\vec{\boldsymbol{u}} = (u_0, \ldots, u_{m-1}) \in (\mathbb{Z}_p)^m$, that is, $f_{\vec{\boldsymbol{u}}}(x) \equiv \sum_{j=0}^{m-1} u_j x^j$. Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative group of prime order p and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map.

We brief how the commitment scheme in [KZG10] supports verification of polynomial evaluation as below. In the setup, a trust party chooses a public key $pk := (g, g^{\alpha}, \ldots, g^{\alpha^{m-1}}) \in \mathbb{G}^m$, where g is a generator of group $\mathbb{G}$ and $\alpha \in \mathbb{Z}_p$ is chosen at random and kept secret. In order to commit a polynomial $f_{\vec{\boldsymbol{u}}}(x) := \sum_{j=0}^{m-1} u_j x^j$ with coefficient vector $\vec{\boldsymbol{u}} = (u_0, \ldots, u_{m-1}) \in (\mathbb{Z}_p)^m$, a committer can compute a commitment $\mathcal{C}$ using the public key pk, that is, $\mathcal{C} := \prod_{j=0}^{m-1} \left(g^{\alpha^j}\right)^{u_j} = g^{f_{\vec{\boldsymbol{u}}}(\alpha)} \in \mathbb{G}$, and then publish $\mathcal{C}$. Later, for any scalar $r \in \mathbb{Z}_p$, the committer can compute $y := f_{\vec{\boldsymbol{u}}}(r) \in \mathbb{Z}_p$ and generate a *short* proof ψ ([KZG10] calls it witness) to convince a verifier that y is indeed the correct evaluation of $f_{\vec{\boldsymbol{u}}}(r)$, without revealing the polynomial $f_{\vec{\boldsymbol{u}}}(x)$. The proof (or witness) ψ is generated as below:

- Divide the polynomial $f_{\vec{\boldsymbol{u}}}(x) - f_{\vec{\boldsymbol{u}}}(r)$ with $(x - r)$ using *polynomial long division*, and denote the coefficient vector of the resulting quotient polynomial as $\vec{\boldsymbol{w}} = (w_0, w_1, \ldots, w_{m-2})$, that is, $f_{\vec{\boldsymbol{w}}}(x) \equiv \frac{f_{\vec{\boldsymbol{u}}}(x) - f_{\vec{\boldsymbol{u}}}(r)}{x - r}$.
- Then compute $\psi := g^{f_{\vec{\boldsymbol{w}}}(\alpha)}$ using the public key pk in the same way as computing $g^{f_{\vec{\boldsymbol{u}}}(\alpha)}$, i.e. $\psi := \prod_{j=0}^{m-2} \left(g^{\alpha^j}\right)^{w_j} = g^{f_{\vec{\boldsymbol{w}}}(\alpha)} \in \mathbb{G}$.

After receiving (r, y, ψ) from the committer, a verifier can verify whether $y \overset{?}{=} f_{\vec{\boldsymbol{u}}}(r)$ with the proof ψ and public key $pk = (g, g^{\alpha}, \ldots, g^{\alpha^{m-1}})$ and the public commitment $\mathcal{C}$ of the unknown polynomial $f_{\vec{\boldsymbol{u}}}(x)$, using a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$:

$$e(g, \mathcal{C})/e(g, g)^y \overset{?}{=} e(\psi, g^{\alpha}/g^r). \tag{5.4}$$

Note that the left hand side of above Equation (5.4) is $e(g, \mathcal{C})/e(g, g)^y = e(g, g)^{f_{\vec{\boldsymbol{u}}}(\alpha) - y}$, and the right hand side is $e(\psi, g^{\alpha}/g^r) = e(g^{f_{\vec{\boldsymbol{w}}}(\alpha)}, g^{\alpha - r}) = e(g, g)^{(\alpha - r) f_{\vec{\boldsymbol{w}}}(\alpha)}$.

In summary, the commitment scheme proposed by Kate and Zaverucha and Goldberg [KZG10] allows the owner of a polynomial $f(x)$ to generate a constant size proof for the correctness of the polynomial evaluation $f(r)$ at *any* particular point $x = r$, without revealing the polynomial $f(x)$.

5.3.2 Construction of $\mathsf{S2}$: Integrating KZG Polynomial Commitment and the Linearly Homomorphic MAC $\mathsf{S1}$

In this subsection, we construct a linearly predicate-homomorphic MAC scheme named $\mathsf{S2}$. The new MAC scheme $\mathsf{S2}$ integrates the main idea of polynomial commitment scheme [KZG10] and the linearly homomorphic MAC scheme $\mathsf{S1}$ we proposed in previous Chapter 4. Recall that:

- The notation $f_{\vec{\boldsymbol{u}}}(x)$ denotes the polynomial with coefficient vector $\vec{\boldsymbol{u}} = (u_0, \dots, u_{m-1})$, that is, $f_{\vec{\boldsymbol{u}}}(x) \equiv \sum_{j=0}^{m-1} u_j x^j$;
- The construction described below exploits an algebraic property of polynomials: for any polynomial $f(x)$ in variable x and for any scalar input r, the polynomial $x - r$ divides the polynomial $f(x) - f(r)$.

The construction of $\mathsf{S2} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Combine}, \mathsf{P\text{-}Sign}, \mathsf{Verify}, \mathsf{P\text{-}Verify})$ as below. We emphasize that the construction of $\mathsf{S2}$ can be instantiated over elliptic curve group [1], although in the following description, $\mathsf{S2}$ is constructed over a modulo group.

$\underline{\mathsf{S2.KeyGen}(1^\lambda) \to (spk, ssk)}$

> Choose at random a λ bits safe prime q such that $p := (q-1)/2$ is also prime. Let $\mathcal{QR}_q$ be the subgroup of quadratic residues modulo q in $\mathbb{Z}_q^*$. Choose at random a generator g of group $\mathcal{QR}_q$. Choose at random two elements τ, α from $\mathbb{Z}_p^*$: $\tau, \alpha \xleftarrow{\$} \mathbb{Z}_p^*$. Choose at random a PRF key, denoted as seed, from the key space of a pseudorandom function family $\{\mathsf{PRF}_{\mathsf{seed}} : \{0,1\}^{2\lambda} \to \mathbb{Z}_p\}$. The public key is $spk := (p, q, \{g^{\alpha^j} \mod q\}_{j=0}^{m-1})$ and the private key is $ssk := (p, q, \mathsf{seed}, \alpha, \tau)$. *Note: (1) Both the size of group $\mathcal{QR}_q$ and the multiplicative order of g modulo q are equal to p; (2) Zero is not in $\mathbb{Z}_q^*$ or $\mathcal{QR}_q$.*

[1] Bilinear map is not required, since $\mathsf{S2}$ does not support public verification.

$\underline{\mathsf{S2.Sign}(ssk, i, \vec{\mathtt{M}}) \to \sigma}$

The input is the private key ssk, an index $i \in \{0,1\}^{2\lambda}$, and a vector $\vec{\mathtt{M}} = (\mathtt{M}_0, \ldots, \mathtt{M}_{m-1}) \in (\mathbb{Z}_p)^m$. Compute the MAC value σ for $\vec{\mathtt{M}}$ as below:

$$\sigma_i := \mathsf{PRF}_{\mathsf{seed}}(i) \ + \ \tau \cdot f_{\vec{\mathtt{M}}}(\alpha) = \mathsf{PRF}_{\mathsf{seed}}(i) \ + \ \tau \sum_{j=0}^{m-1} \mathtt{M}_j \alpha^j \mod p. \tag{5.5}$$

$\underline{\mathsf{S2.Combine}(spk, \{(i, \vec{\mathtt{M}}_i, \sigma_i, \nu_i) : i \in C\}) \to (\vec{\boldsymbol{\mu}}, \sigma)}$

The input consists of a public key spk, a sequence of tuples $(i, \vec{\mathtt{M}}_i, \sigma_i, \nu_i)$, $i \in C$, where i is an index, $\vec{\mathtt{M}}_i \in (\mathbb{Z}_p)^m$ is a message in vector form, σ_i is the MAC value of $\vec{\mathtt{M}}_i$, and $\nu_i \in \mathbb{Z}_p^*$ is a weight for linear combination. Compute a message-MAC pair $(\vec{\boldsymbol{\mu}}, \sigma)$ as below

$$\vec{\boldsymbol{\mu}} := \sum_{i \in C} \nu_i \vec{\mathtt{M}}_i \mod p \ \in (\mathbb{Z}_p)^m; \qquad \sigma := \sum_{i \in C} \nu_i \sigma_i \mod p \in \mathbb{Z}_p. \tag{5.6}$$

Output $(\vec{\boldsymbol{\mu}}, \sigma)$.

$\underline{\mathsf{S2.P\text{-}Sign}(spk, \vec{\boldsymbol{\mu}}, \sigma, r) \to (f_{\vec{\boldsymbol{\mu}}}(r), \hat{\sigma})}$

The input consists of a public key spk, a message $\vec{\boldsymbol{\mu}} = (\mu_0, \ldots, \mu_{m-1}) \in (\mathbb{Z}_p)^m$ in vector form, a MAC value σ for $\vec{\boldsymbol{\mu}}$, and a value $r \in \mathbb{Z}_p$. Evaluate polynomial $f_{\vec{\boldsymbol{\mu}}}(x)$ at point $x = r$ to obtain $y := f_{\vec{\boldsymbol{\mu}}}(r) \mod p$. Divide the polynomial $f_{\vec{\boldsymbol{\mu}}}(x) - f_{\vec{\boldsymbol{\mu}}}(r)$ in variable x with $(x - r)$ using polynomial long division, and denote the coefficient vector of the resulting quotient polynomial as $\vec{\boldsymbol{w}} = (w_0, \ldots, w_{m-2}) \in (\mathbb{Z}_p)^{m-1}$, that is,

$$f_{\vec{\boldsymbol{w}}}(x) \equiv \frac{f_{\vec{\boldsymbol{\mu}}}(x) - f_{\vec{\boldsymbol{\mu}}}(r)}{x - r}.$$

Compute ψ with the public key $spk = (p, q, \{g^{\alpha^j} \mod q\}_{j=0}^{m-1})$ as below

$$\psi := \prod_{j=0}^{m-2} \left(g^{\alpha^j}\right)^{w_j} = g^{f_{\vec{w}}(\alpha)} \mod q. \tag{5.7}$$

The MAC value for y is $\hat{\sigma} := (\psi, \sigma)$. Output $(y, \hat{\sigma})$.

S2.Verify$(ssk, \vec{\boldsymbol{\mu}}, \sigma, \{(i, \nu_i) : i \in C\}) \rightarrow$ `accept` or `reject`

The input consists of a private key ssk, a message $\vec{\boldsymbol{\mu}} \in (\mathbb{Z}_p)^m$ in vector form, and a sequence of tuples (i, ν_i), $i \in C$, where i is an index and ν_i is a weight for linear combination. Check the following equality with the private key $ssk = (p, q, \mathsf{seed}, \alpha, \tau)$. If Equality (5.8) holds, then output `accept`; otherwise output `reject`.

$$\left(\sum_{i \in C} \nu_i \mathsf{PRF}_{\mathsf{seed}}(i)\right) + \tau \cdot f_{\vec{\boldsymbol{\mu}}}(\alpha) \overset{?}{=} \sigma \mod p. \tag{5.8}$$

S2.P-Verify$(ssk, y, (\psi, \sigma), \{(i, \nu_i) : i \in C\}, r) \rightarrow$ `accept` or `reject`

The input consists of a private key ssk, a message $y \in \mathbb{Z}_p$, a MAC value (ψ, σ) for y, a sequence of tuples (i, ν_i), $i \in C$, where i is an index and ν_i is a weight for linear combination, and value $r \in \mathbb{Z}_p$. Check the following equality with the private key $ssk = (p, q, \mathsf{seed}, \alpha, \tau)$. If Equality (5.9) holds and $\psi \in \mathcal{QR}_q$ is a quadratic residue, then output `accept`; otherwise output `reject`.

$$\psi^{\alpha - r} \overset{?}{=} g^{\tau^{-1}\left(\sigma - \sum_{i \in C} \nu_i \mathsf{PRF}_{\mathsf{seed}}(i)\right) - y} \mod q \tag{5.9}$$

Remark 3

- *In case that $\alpha - r$ is even, if (y, ψ, σ) satisfies Equation (5.9), so does $(y, -\psi, \sigma)$.*
- *We check whether ψ is a quadratic residue modulo q in algorithm* S2.P-Verify, *in order to avoid accepting $(y, -\psi, \sigma)$. The reasons are as below:*

– *From motivation point of view, we hope to authenticate both the message y and the MAC value (ψ, σ), so that we can achieve stronger security.*

– *The correct value ψ generated by the Equation (5.7) is*

$$\psi = g^{f_{\vec{w}}(r)} \mod q \quad \textit{is within group } \mathcal{QR}_q,$$

since $g \in \mathcal{QR}_q$.

– *Recall that $q = 2p+1$ is a safe prime with p being prime, thus $q = 3 \mod 4$. For such kind of prime q, the negation of a quadratic residue modulo q is not a quadratic residue [Gol06].*

- $\mathsf{S2}$ *can be alternatively instantiated over an elliptic curve group, and bilinear map operation is not required. This is different from [KZG10], since* $\mathsf{S2}$ *only supports private key verification.*

5.3.3 Correctness

Recall that $f_{\vec{u}}(x)$ denotes the polynomial in variable x with vector $\vec{\boldsymbol{u}} = (u_0, \ldots, u_{m-1})$ as coefficients: $f_{\vec{u}}(x) = \sum_{i=1}^{m-1} u_i x^i$. We define a predicate family $\{P_r : \mathbb{Z}_p \times (\mathbb{Z}_p)^m \to \{0,1\}\}$ w.r.t. the polynomial function[2] $f_{\vec{u}}$ as below

$$P_r(y, \vec{\boldsymbol{u}}) = \begin{cases} 1 & (\text{ if } f_{\vec{u}}(r) = y \mod p) \\ 0 & (\text{o.w.}) \end{cases} \tag{5.10}$$

Lemma 5.1 ($\mathsf{S2}$ is a Linearly Predicate-Homomorphic MAC) *The proposed MAC scheme* $\mathsf{S2}$ *is both linearly homomorphic and predicate-homomorphic w.r.t. predicate family $\{P_r : r \in \mathbb{Z}_p\}$ as defined in Equation (5.10), under Definition 8 (on page 56).*

[2]Here the corresponding two input function $F(\cdot,\cdot)$ is $F(\vec{\boldsymbol{u}}, r) = f_{\vec{u}}(r)$.

Proof of Lemma 5.1: One can verify that the following Equality (5.11) holds through some straightforward algebra manipulation over polynomials.

$$\sum_{i\in C} \nu_i f_{\vec{\mathtt{M}}_i}(\alpha) = f_{\vec{\boldsymbol{u}}}(\alpha), \quad \text{where } \vec{\boldsymbol{u}} = \sum_{i\in C} \nu_i \vec{\mathtt{M}}_i. \tag{5.11}$$

The following proof will utilize the above equality.

Linear-Homomorphism Let $\vec{\boldsymbol{\mu}} = \sum_{i\in C} \nu_i \vec{\mathtt{M}}_i$ and $\sigma = \sum_{i\in C} \nu_i \sigma_i$ be as computed by the algorithm $\mathsf{S2.Combine}$. The RHS (right hand side) of Equation (5.8) is

$$\begin{aligned}
RHS &= \sum_{i\in C} \nu_i \sigma_i = \sum_{i\in C} \nu_i \big(\mathsf{PRF}_{\mathsf{seed}}(i) \ + \ \tau \cdot f_{\vec{\mathtt{M}}_i}(\alpha)\big) && (5.12)\\
&= \sum_{i\in C} \nu_i \mathsf{PRF}_{\mathsf{seed}}(i) \ + \ \tau \sum_{i\in C} \nu_i f_{\vec{\mathtt{M}}_i}(\alpha) && (5.13)\\
&= \sum_{i\in C} \nu_i \mathsf{PRF}_{\mathsf{seed}}(i) \ + \ \tau f_{\sum_{i\in C} \nu_i \vec{\mathtt{M}}_i}(\alpha) \quad (\text{ Since Equality (5.11) }) && (5.14)\\
&= \sum_{i\in C} \nu_i \mathsf{PRF}_{\mathsf{seed}}(i) \ + \ \tau f_{\vec{\boldsymbol{\mu}}}(\alpha) && (5.15)\\
&= LHS. && (5.16)
\end{aligned}$$

Predicate-Homomorphism Suppose $\psi = g^{f_{\vec{\boldsymbol{w}}}(\alpha)} \mod p$, $\sigma = \sum_{i\in C} \nu_i \sigma_i \mod p$, $y = f_{\vec{\boldsymbol{\mu}}}(r) \mod p$. Then $\psi \in \mathcal{QR}_q$ is a quadratic residue, and the LHS (left hand side) of Equation (5.9) is

$$LHS = \left(g^{f_{\vec{\boldsymbol{w}}}(\alpha)}\right)^{\alpha - r} = g^{f_{\vec{\boldsymbol{w}}}(\alpha)\times(\alpha-r)} = g^{\frac{f_{\vec{\boldsymbol{\mu}}}(\alpha) - f_{\vec{\boldsymbol{\mu}}}(r)}{\alpha - r}\times(\alpha - r)} = g^{f_{\vec{\boldsymbol{\mu}}}(\alpha) - f_{\vec{\boldsymbol{\mu}}}(r)} \mod q. \tag{5.17}$$

The RHS (right hand side) of Equation (5.9) is

$$\begin{aligned} RHS &= g^{\tau^{-1}\left(\sum_{i\in C}\nu_i\sigma_i \;-\; \sum_{i\in C}\nu_i \mathsf{PRF}_{\mathsf{seed}}(i)\right) \;-\; y} && (5.18)\\ &= g^{\tau^{-1}\left(\sum_{i\in C}\nu_i(\sigma_i - \mathsf{PRF}_{\mathsf{seed}}(i))\right) \;-\; y} && (5.19)\\ &= g^{\tau^{-1}\left(\sum_{i\in C}\;\nu_i\;\cdot\;\tau f_{\vec{\mathtt{M}}_i}(\alpha)\right) \;-\; y} && (5.20)\\ &= g^{\left(f_{\sum_{i\in C}\nu_i\vec{\mathtt{M}}_i}\;(\alpha)\right)\;-\;y} && (5.21)\\ &= g^{f_{\vec{\boldsymbol{\mu}}}(\alpha)\;-\;y} \qquad (\text{ Since } \vec{\boldsymbol{\mu}} = \sum_{i\in C}\nu_i\vec{\mathtt{M}}_i) && (5.22)\\ &= g^{f_{\vec{\mu}}(\alpha)\;-\;f_{\vec{\mu}}(r)} = LSH \mod q. && (5.23) \end{aligned}$$

Note that we obtain Equation (5.21) from Equation (5.20), i.e. the equality between $\sum_{i\in C}\nu_i f_{\vec{\mathtt{M}}_i}(\alpha)$ and $f_{\vec{\boldsymbol{u}}}(\alpha)$ with vector $\vec{\boldsymbol{u}} = \sum_{i\in C}\nu_i\vec{\mathtt{M}}_i$, due to Equality (5.11). $\square$

5.4 POS2: A Proofs of Retrievability scheme constructed from Homomorphic MAC S2

In this section, we construct an efficient $\mathcal{POR}$ scheme with private verifiability, by applying the newly constructed linearly predicate-homomorphic MAC scheme S2. We refer to this scheme as POS2 in this dissertation.

5.4.1 Construction of POS2

$\underline{\mathsf{POS2.KeyGen}(1^\lambda) \to (pk, sk)}$

Invoke the key generating algorithm S2.KeyGen to generate the public-private key pair: $(pk, sk) := \mathsf{S2.KeyGen}(1^\lambda)$. Output (pk, sk).

$\underline{\mathsf{POS2.DEncode}(sk, \mathtt{F}) \to (\mathsf{id}, \hat{\mathtt{F}}, n)}$

The input consists of a private key sk and a file $\mathtt{F}$. Let $\rho \in (0, 1)$ be a system parameter. Apply a rate-ρ error erasure code on the file $\mathtt{F}$ to generate file blocks $(\vec{\mathtt{F}}_0, \ldots, \vec{\mathtt{F}}_{n-1})$, such that each block $\vec{\mathtt{F}}_i = (\mathtt{F}_{i,0}, \ldots, \mathtt{F}_{i,m-1}) \in (\mathbb{Z}_p)^m$ is a vector of sectors $\mathtt{F}_{i,j} \in \mathbb{Z}_p$, and any ρn number of blocks $\vec{\mathtt{F}}_i$'s can recover the original file $\mathtt{F}$. Choose a unique identifier id from domain $\{0,1\}^\lambda$. For each block $\vec{\mathtt{F}}_i$, $i \in [0, n-1]$, generate a MAC value σ_i using the MAC scheme $\mathsf{S2}$ under private key sk as below:

$$\sigma_i := \mathsf{S2.Sign}(sk,\ \mathsf{id}\|i,\ \vec{\mathtt{F}}_i). \tag{5.24}$$

The encoded file $\hat{\mathtt{F}}$ is $\{(i, \vec{\mathtt{F}}_i, \sigma_i) : i \in [0, n-1]\}$. Send $(\mathsf{id}, \hat{\mathtt{F}})$ to the cloud storage server and keep (id, n) in local storage. Output $(\mathsf{id}, \hat{\mathtt{F}}, n)$.

Note: Figure 5.2 illustrates the data organization. In this figure, a rectangle represents a data block $\vec{\mathtt{F}}_i$ and a square within a rectangle represents a data sector $\mathtt{F}_{i,j}$ in the data block $\vec{\mathtt{F}}_i$. A circle below represents the corresponding authentication tag σ_i. Those shaded rectangles represent data blocks that are generated by the error erasure code.

$\underline{\mathsf{POS2.Prove}(pk, \mathsf{id}, \hat{\mathtt{F}}, \{(i, \nu_i) : i \in C\}, r) \to (y, \psi, \sigma)}$

The input consists of a public key pk, a file identifier id, an encoded file $\hat{\mathtt{F}} = \{(i, \vec{\mathtt{F}}_i, \sigma_i) : i \in [0, n-1]\}$ which is associated with the identifier id, and a challenge $(\{(i, \nu_i) : i \in C\}, r)$, where $i \in C \subseteq [0, n-1]$ is index, $\nu_i \in \mathbb{Z}_p^*$ is a weight for linear combination, and $r \in \mathbb{Z}_p$. Apply the linear homomorphism of $\mathsf{S2}$ to compute a message-MAC pair $(\vec{\boldsymbol{\mu}}, \sigma_{\vec{\boldsymbol{\mu}}})$ as below

$$(\vec{\boldsymbol{\mu}}, \sigma_{\vec{\boldsymbol{\mu}}}) := \mathsf{S2.Combine}(pk, \{(\mathsf{id}\|i, \vec{\mathtt{F}}_i, \sigma_i, \nu_i) : i \in C\}). \tag{5.25}$$

Next, apply the predicate-homomorphism of $\mathsf{S2}$ to generate a message-MAC pair $(y, \hat{\sigma})$ as below

$$(y, \hat{\sigma}) := \mathsf{S2.P\text{-}Sign}(pk, \vec{\boldsymbol{\mu}}, \sigma_{\vec{\boldsymbol{\mu}}}, r). \tag{5.26}$$

Parse $\hat{\sigma}$ as (ψ, σ) and send (y, ψ, σ) back to verifier (i.e. data owner). Output (y, ψ, σ).

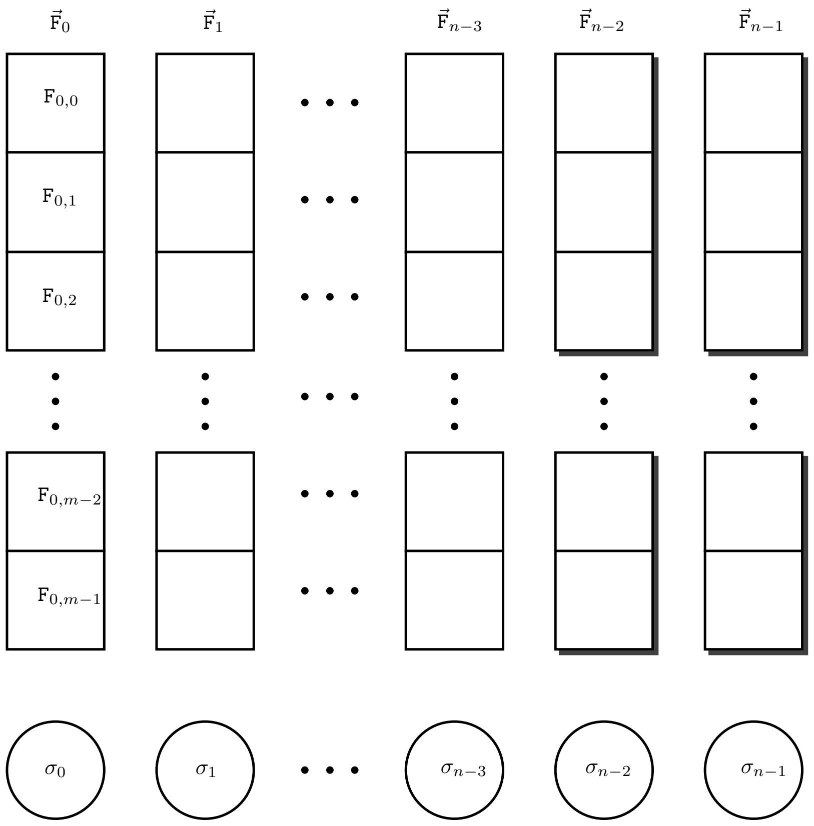


Figure 5.2: Data Organization of POS2

$\mathsf{POS2.Verify}(sk, \mathsf{id}, (y, \psi, \sigma), \{(i, \nu_i) : i \in C\}, r) \rightarrow \mathsf{accept}$ or reject

The input consists of a private key sk, a file identifier id, the response (y, ψ, σ) from the prover (i.e. the cloud storage server), and the challenge $(\{(i, \nu_i) : i \in C\}, r)$, where $i \in C \subseteq [0, n-1]$ is an index, $\nu_i \in \mathbb{Z}_p^*$ is a weight for linear combination, and $r \in \mathbb{Z}_p$. Invoke the verification algorithm $\mathsf{S2.P\text{-}Verify}$ of MAC scheme $\mathsf{S2}$ to obtain $b \in \{\mathtt{accept}, \mathtt{reject}\}$ with private key sk as below, and output b.

$$b := \mathsf{S2.P\text{-}Verify}(sk, y, (\psi, \sigma), \{(\mathsf{id}\|i, \nu_i) : i \in C\}, r) \ \in \{\mathtt{accept}, \mathtt{reject}\}. \quad (5.27)$$

Remark 4

- *The challenge* $\{(\mathsf{id}\|i, \nu_i) : i \in C, \nu_i \in \mathbb{Z}_p^*\}$ *can be represented compactly with two short random seeds of a pseudorandom function under random oracle model [SW08a, DVW09]. Alternatively, Dodis, Vadhan and Wichs [DVW09]'s PoR Code can be applied.*

- *To simplify the proof, we assume that* ν_i*'s,* $i \in C$*, forms a simple geometric sequence. More precisely, denote the elements of set* C *as* $\{i_j \in [0, n-1] : j \in [0, \ell - 1]\}$*, where* ℓ *is the size of set* C *and* $0 \leq i_0 < i_1 < i_2 \ldots < i_{\ell-1} < n$*. There exists some element* $\nu \in \mathbb{Z}_p^*$*, for each* $j \in [0, \ell-1]$*,* $\nu_{i_j} = \nu^j \mod p$*. In other words, the sequence* $(\ldots, \nu_i, \ldots)_{i \in C}$*, ordered by increasing* i*, forms a simple geometric sequence* $(\nu^0, \nu^1, \ldots, \nu^{\ell-1})$*. Thus,* ℓ *distinct vectors of form* $(\nu^0, \nu^1, \ldots, \nu^{\ell-1}) \in \left(\mathbb{Z}_p^*\right)^\ell$ *can constitute a vandermonde matrix [MS58].*

- *Compared to Shacham and Waters [SW08a] scheme, the algorithm* Prove *in* POS2 *is able to aggregate* m *number of weighted sums* $\mu_0, \mu_1, \ldots, \mu_{m-1}$ *into two short numbers* y *and* ψ *using the idea in the polynomial commitment scheme [KZG10], where* $y = f_{\vec{\mu}}(r) = \sum_{j=0}^{m-1} \mu_j r^j \in \mathbb{Z}_p$ *(*r *is a random nonce chosen by the data owner) and* $\psi = g^{\frac{f_{\vec{\mu}}(\alpha) - f_{\vec{\mu}}(r)}{\alpha - r}} \in \mathbb{Z}_q^*$*. In this way,* POS2 *requires only* $\mathcal{O}(\lambda)$ *communication bits per verification. In comparison, the Shacham and Waters [SW08a] scheme requires* $\mathcal{O}(m\lambda)$ *communication bits per verification, since* $(\mu_0, \mu_1, \ldots, \mu_{m-1})$ *are sent back directly as the response.*

5.4.2 Completeness

Lemma 5.2 (POS2 is Complete) *The above construction* POS2 *is complete under Definition 5 (on page 24).*

The completeness of POS2 is implied by the correctness of the underlying homomorphic MAC scheme S2 (Lemma 5.1). Recall that, a $\mathcal{POR}$ scheme is complete, if any genuine proof generated by following the $\mathcal{POR}$ scheme honestly will be accepted by the verifier.

5.5 Performance Analysis

In this section, we analyze the performance of our proposed scheme POS2 in communication, storage, and computation. We also compare POS2 with existing works by Shacham and Waters [SW08a] and Ateniese *et al.* [ABC+07, ABC+11b]. We remark that, the scheme POS2 can be instantiated over a modulo group of size 2^{1024}, or over an elliptic curve with much smaller group size 2^{160} (Bilinear map is not required).

5.5.1 Communication

During a verification, the communication cost is the size of a challenge plus the size of its corresponding response (or proof). In our scheme POS2, the challenge consists of a set $\{(i, \nu_i) : i \in C\}$ and a group element r. As mentioned previously, the set $\{(i, \nu_i) : i \in C\}$ can be represented compactly with two 80 bits PRF seeds. The group element r is used to retrieve a polynomial function value $f(r)$ for some polynomial $f(x)$ determined by a linear combination of the data blocks specified in the set C. In the security analysis, the goal of r is to retrieve multiple function values $f(r_i)$'s for different inputs r_i, and then recover the polynomial $f(x)$ by solving a linear equation system. For this reason, we can simply choose r from a smaller range $[1, 2^{80}]$ without any sacrificing in the security. As a result, the challenge size is $80 \times 3 = 240$ bits.

In our scheme POS2, a response, i.e. the proof, consists of three group elements y, σ, ψ, which are derived from the challenge, the data blocks and authentication tags. So the size of a response is 3λ bits. Therefore, the communication cost per verification is $3\lambda + 240$ bits.

5.5.2 Storage

During verification, the data owner only keeps the private key, and identifier and size (in term of number of blocks) of each file in her local storage. Similar to Shacham and Waters [SW08a], in the scheme POS2, the data owner can sign the meta-data of each file and store the meta-data together with data owner's digital signature in the cloud storage. Before conducting a verification, the data owner retrieves the meta-data for the file of interest from the cloud storage, and check its digital signature. Thus, the storage cost on data owner side is just the size of private key, which is $3\lambda + 80$ bits.

The storage overhead (due to authentication tags) on the cloud storage server side is $1/m$ of the data file size, where the system parameter m is the block size and is equal to the ratio of the size of a data block to the size of an authentication tag. The public key is also kept in the cloud storage and its size is $(m+1)\lambda$ bits. Note that in our scheme, there is only one public key per user, without regarding to the number of files the user stores in the cloud storage server.

5.5.3 Computation

Our scheme is very efficient in setup. Key generation requires m number of group exponentiations. Suppose a $nm\lambda$ bits data file consists of n data blocks, each block has m group elements and each group element has λ bits. The data preprocess (i.e. the DEncode algorithm) requires only nm number of group multiplications and additions, together with n PRF evaluations. Note that the PRF can be simulated [BH05] with an AES cipher [DR02] in counter mode.

During a verification, the computation complexity on the cloud storage server side is dominated by the computation of ψ in Equation (5.7) in the algorithm P-Sign (on page 61), which is invoked by algorithm Prove. This dominant step takes $(m-1)$ number of group exponentiations, and is the bottleneck of efficiency of our scheme when the block size m becomes large. We will measure the actual running time in Section 5.5.6.

5.5.4 Recommended System Parameters

We recommend the following system parameter for our proposed scheme POS2: The error erasure rate is 0.98, block size m is around 160, the challenge size is around 500. In this setting, the false accept rate is 4.1024×10^{-5} (False accept rate is analyzed in Section 2.3.2 and example values are listed in Table 2.1 on page 14), the number of communication bits required in a verification is 720 for elliptic curve group or 3312 for modulo group, the storage overhead is about 2% due to erasure encoding and

Table 5.1: Comparison with an example among the $\mathcal{PDP}$ scheme by Ateniese *et al.* [ABC+07], the $\mathcal{POR}$ scheme by Shacham and Waters [SW08a], and the $\mathcal{POR}$ scheme POS2 proposed in this chapter. After erasure encoding, the file size is 1GB, block size is $m = 100$, and storage overhead due to authentication tags is about 10MB for all schemes. For all schemes listed below, we assume that, during a verification, the challenge $\{(i, \nu_i) : i \in C\}$ are represented by two 80 bits PRF seeds. System parameter ℓ represents the size of set C. All computation times are represented by the corresponding dominant factor. `exp` and `mul` denote the group exponentiation and group multiplication respectively in the corresponding group. Note that one 1024 bits modular exponentiation or one 160 bits elliptic curve exponentiation takes roughly 5 millisecond in a standard PC.

Scheme	**Group Element Size (bits)**	**Communication bits**	**Computation (Data Preprocess)**	**Computation (Prove)**
Ateniese [ABC+07]	$\lambda = 1024$	$2\lambda + 160 + 256 = 2464$	2^{23} `exp.` over $\mathbb{Z}_N^*$	$(100 + \ell)$ `exp.` over $\mathbb{Z}_N^*$
SW [SW08a]	$\lambda = 80$	$(m+1)\lambda + 160 = 8240$	2^{27} `mul.` over $\mathbb{Z}_p$	100ℓ `mul.` over $\mathbb{Z}_p$
POS2 (E.C.)	$\lambda = 160$	$3\lambda + 240 = 720$	2^{26} `mul.` over $\mathbb{Z}_p$	100 `exp.` over Elliptic Curve
POS2 ($\mathbb{Z}_q^*$)	$\lambda = 1024$	$3\lambda + 240 = 3312$	2^{23} `mul.` over $\mathbb{Z}_p$	100 `exp.` over $\mathbb{Z}_q^*$

$1/160 = 0.625\%$ due to authentication tags. Our experiment will confirm that the query latency is within 1 second.

5.5.5 Comparison

We give a comparison on the performances of our scheme with Shacham and Waters [SW08a] and Ateniese *et al.* [ABC+07] in Table 5.1 with an example. A more detailed and generic comparison is given in Table 5.2 (on page 71). Note that in our proposed scheme POS2, the practical choice of value m is bounded by the computation on the server side, which is similar to the case of Ateniese [ABC+07]. In contrast, in SW [SW08a], the largest practical value of m (SW [SW08a] uses "s" to denote this value) is limited by the communication requirement.

Table 5.2: Performance Comparison. All schemes support private verification only. In each scheme, a challenge set $\{(i, \nu_i) : i \in C\}$ contains ℓ index-coefficient pairs and is represented compactly by two 80 bits PRF seeds. In the table, `exp`, `mul` and `add` represent exponentiation, multiplication and addition in the corresponding groups/fields; notation $|\mathtt{F}|$ denotes the file size in bits. *Note: In Ateniese's $\mathcal{PDP}$ scheme, exponentiation with a large integer of $m\lambda$ bits is required. We represent such exponentiation as a number of m normal group exponentiation* `exp`*, where the exponent is λ bits long. Similar for the RSA based scheme.*

Scheme	Finite Field Element Size (bits)	Communication (bits)	Storage Overhead	Computation (Prover)	Computation (Verifier)	Computation (Data Preprocess)
RSA-based scheme [DQS03]	$\lambda = 1024$	2λ	Zero	$\lvert\mathtt{F}\rvert/\lambda$ `exp.`	1 `exp.`	1 `exp.`
PolyCommit [KZG10]	$\lambda = 160$	3λ	Zero	$\lvert\mathtt{F}\rvert/\lambda$ `exp.` + $2\lvert\mathtt{F}\rvert/\lambda$ (`mul.` + `add.`)	2 `pairing`	$\lvert\mathtt{F}\rvert/\lambda$ (`mul.` + `add.`) + 1 `exp.`
PolyDelegation [BGV11]	$\lambda = 160$	$2\lambda + 240$	$\lvert\mathtt{F}\rvert$	$\lvert\mathtt{F}\rvert/\lambda$ (`exp.` + `mul.` + `add.`)	2 `exp.`	$\lvert\mathtt{F}\rvert/\lambda$ (`exp.` + `mul.` + `add.`)
Ateniese [ABC$^+$11b, ABC$^+$07]	$\lambda = 1024$	$2\lambda + 320$	$\lvert\mathtt{F}\rvert/m$	$(\ell+m)$ `exp.` + 2ℓ `mult.` + ℓ `add` + 1 `hash` + 2ℓ `PRF`	ℓ (`exp.` + `mult.`) + 1 `hash`	$\lvert\mathtt{F}\rvert/\lambda$ `exp.` + n `hash`
S.W. [SW08a](Private Verification)	$\lambda = 80$	$(m+1)\lambda +$ 160	$\lvert\mathtt{F}\rvert/m$	$m\ell$ (`add` + `mult`) + 2ℓ `PRF`	$(\ell + m)$ (`add` + `mult`) + 3ℓ `PRF`	$\lvert\mathtt{F}\rvert/\lambda$ (`mul.` + `add.`) + $\lvert\mathtt{F}\rvert/(\lambda m)$ `PRF`
POS2 (Elliptic Curve)	$\lambda = 160$	$3\lambda + 240$	$\lvert\mathtt{F}\rvert/m$	$(m-1)$ `exp.` + $(m\ell+m+\ell)$ (`add` + `mul`) + 2ℓ `PRF`	2 `exp.` + ℓ (`add` + `mult`) + 3ℓ `PRF`	$\lvert\mathtt{F}\rvert/\lambda$ (`mul.` + `add.`) + $\lvert\mathtt{F}\rvert/(\lambda m)$ `PRF`
POS2 ($\mathbb{Z}_q^*$)	$\lambda = 1024$	$3\lambda + 240$	$\lvert\mathtt{F}\rvert/m$	$(m-1)$ `exp.` + $(m\ell+m+\ell)$ (`add` + `mul`) + 2ℓ `PRF`	2 `exp.` + ℓ (`add` + `mult`) + 3ℓ `PRF`	$\lvert\mathtt{F}\rvert/\lambda$ (`mul.` + `add.`) + $\lvert\mathtt{F}\rvert/(\lambda m)$ `PRF`

`Notations:` λ `is the size of group/field and also functions as the security parameter,` n `is the number of blocks in a file,` m `is the number of group elements in a block, and` ℓ `is the number of blocks accessed during a verification.`

We also compare the proposed scheme POS2 with SW scheme [SW08a] in communication and storage overhead. For a 1GB data file, we plot the number of communication bits (i.e. the size of a challenge and a proof) against the storage overhead for both schemes in Figure 5.3.

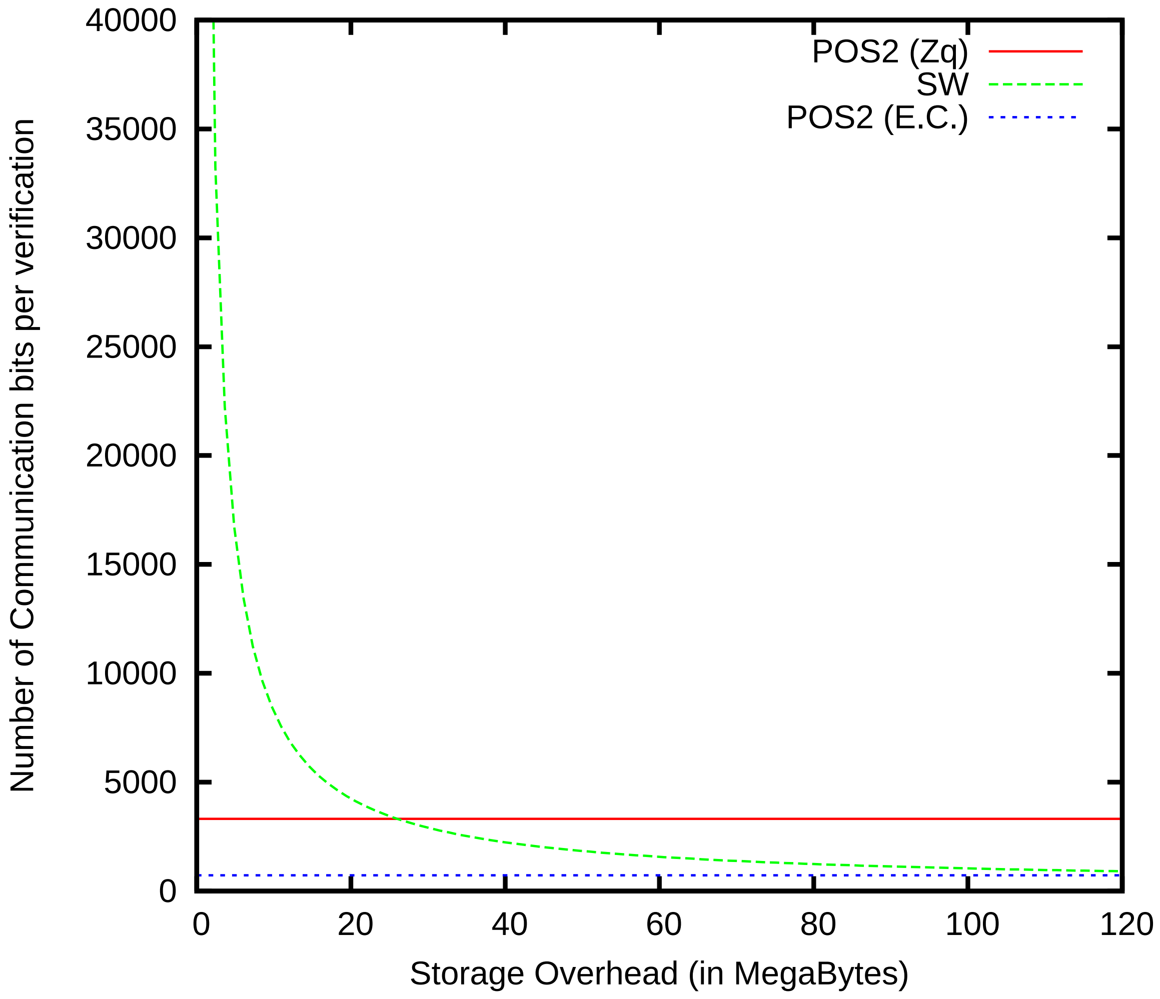


Figure 5.3: Comparison on communication (in bits) and storage overhead (in megabytes) w.r.t. a 1GB data file. **SW** denotes the $\mathcal{POR}$ scheme with private verification by Shacham and Waters [SW08a]; POS2 (E.C.) denotes our proposed scheme instantiated over elliptic curve group; POS2 ($\mathbb{Z}_q^*$) denotes our proposed scheme instantiated over group $\mathbb{Z}_q^*$.

Table 5.3: The choices of values of various system parameters in our experiment

System Parameter	**Semantics**	**Choices of Values**
λ	Group element size in bits	1024
m	The number of group elements in a data block	40, 80, 160, 320, 640, or 960
ℓ	The number of data blocks accessed in a verification	100, 300, 500 or 700.

5.5.6 Experiment: Measuring the computation time

The goal of this experiment is to measure the actual running time of the four algorithms $\mathsf{KeyGen}, \mathsf{DEncode}, \mathsf{Prove}, \mathsf{Verify}$ in the proposed scheme $\mathsf{POS2}(\mathbb{Z}_q^*)$, with disk IO time included and networking communication time excluded. Note that the reported query latency includes of running time of Prove and Verify and disk IO time.

5.5.6.1 Experiment Environment and Setting

We have implemented a prototype of our $\mathsf{POS2}$ scheme in C programming language. The large integer arithmetic is computed using GNU MP [GMP] library with version 5.0.1. The pseudorandom function PRF are simulated with AES symmetric cipher provided in OpenSSL [Ope] library with version 1.0.0d. The disk IO is handled by C library function `mmap`. We observe a low memory consumption for all experiments conducted. Our implementation is not optimized and further performance improvements of our scheme can be expected.

Our test machine is a laptop computer, which is equipped with a 2.5GHz Intel Core 2 Duo CPU (model T9300), a 3GB PC2700-800MHZ RAM and a 7200RPM hard disk. The test machine runs 32 bits version of Gentoo Linux OS with kernel 2.6.36. The file system is EXT4 with 4KB page size.

Our test data files are of size 16MB, 32MB, 64MB, 128MB, 256MB and 512MB, respectively (We assume these are the file sizes after error erasure encoding). The choices of values of various system parameters, i.e. group element size λ, block size m and challenge size ℓ, are listed in Table 5.3.

Our experiments are conducted in this way:

- Key generation: For each choice of block size m, we generate a key pair with size m using the key generating program KeyGen. The generated public key consists of m group elements.

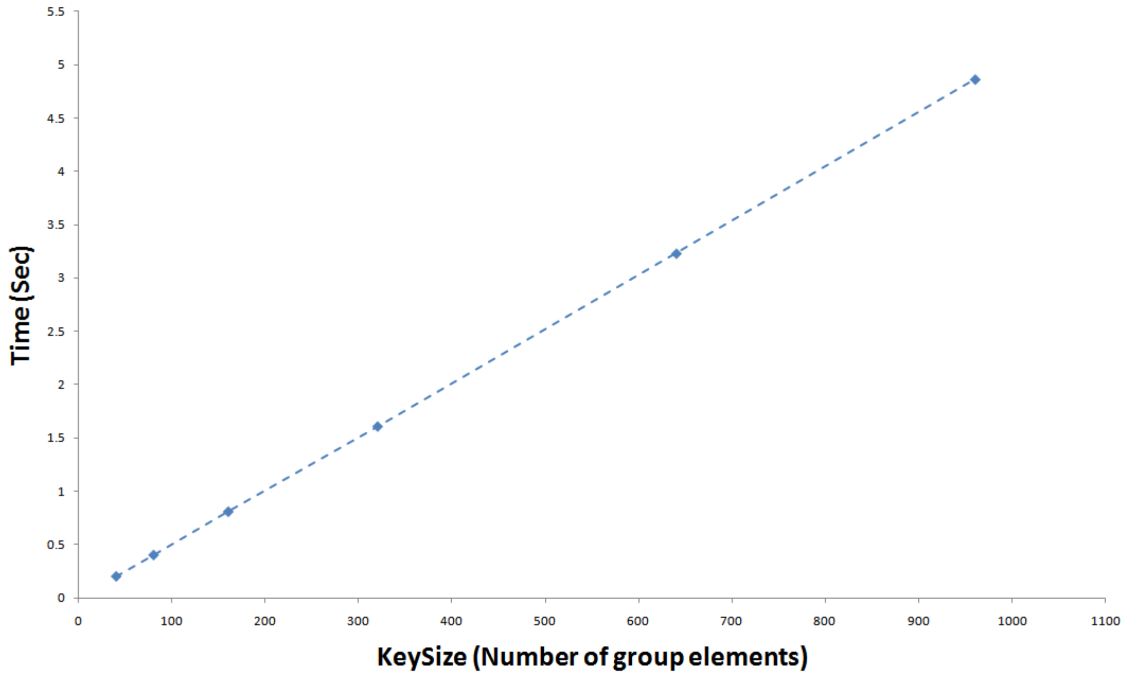
- Data preprocess: For each test file, for each choice of value of block size m, we run the data encoding program DEncode to generate a set of authentication tags.
- Verification: For each test file, for each choice of value of block size m, for each choice of value of challenge size ℓ, we run the Prove and Verify programs to simulate the interaction between the data owner and cloud storage server.

Every single experiment case is repeated for 10 times and the reported timing data are the averages. We remark that experiment trials are run in sequence without parallelism.

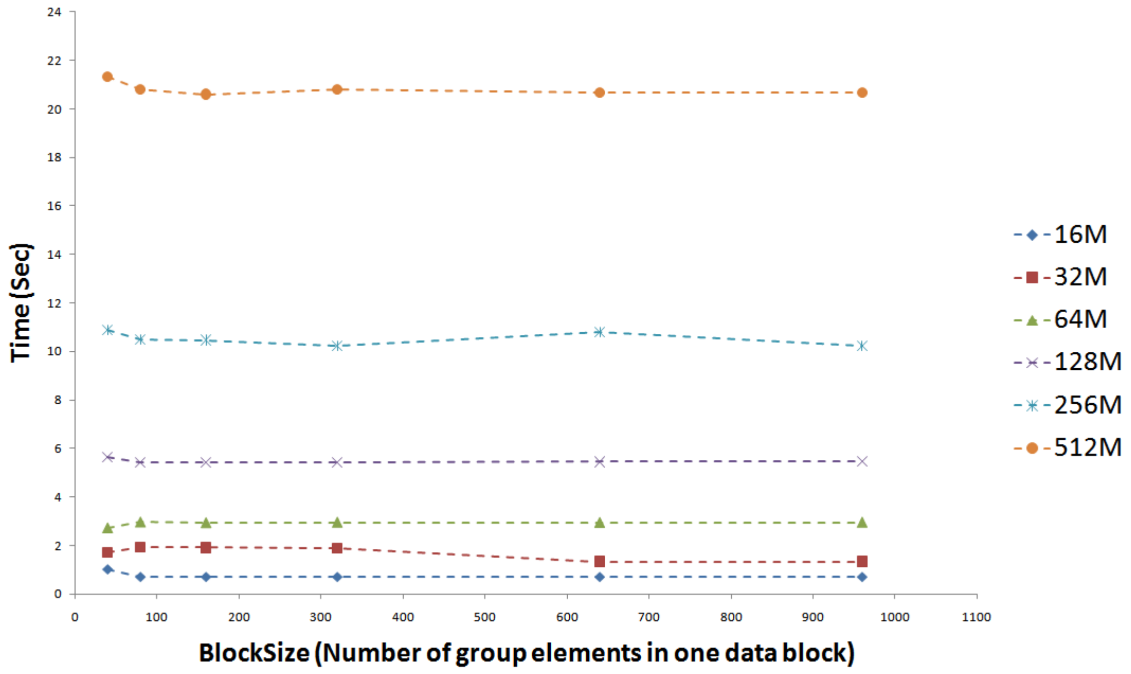
5.5.6.2 Experiment Results

The experiment results are showed in Figure 5.4 and Figure 5.5. All experiment results are averaged over 10 trials. Since all experiment results vary little across different trials, we do not report the variances or confidence intervals.

Our experiment result in Figure 5.4(a) indicates that the key generating time is proportional to the key size, i.e. the number of group elements in a key. The experiment result in Figure 5.4(b) indicates that the data preprocess time (particularly, DEncode) is proportional to the data file size and almost independent on the block size s. The experiment (Figure 5.5(a) and Figure 5.5(b)) also shows that the query latency is proportional to the block size m, almost independent on the file size, and grows very slowly with the challenge size ℓ, suggesting that the computation of exponentiations becomes the bottleneck when m is so large. All of these results agree with our analysis.

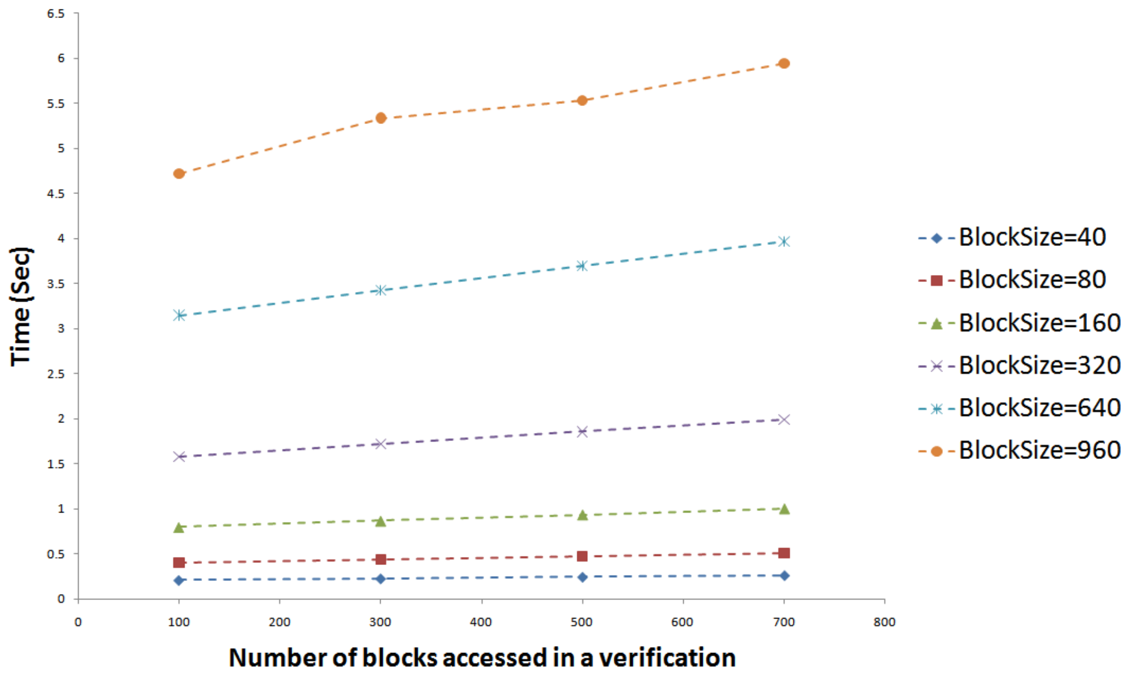


(a) Time to generate a key VS the key size

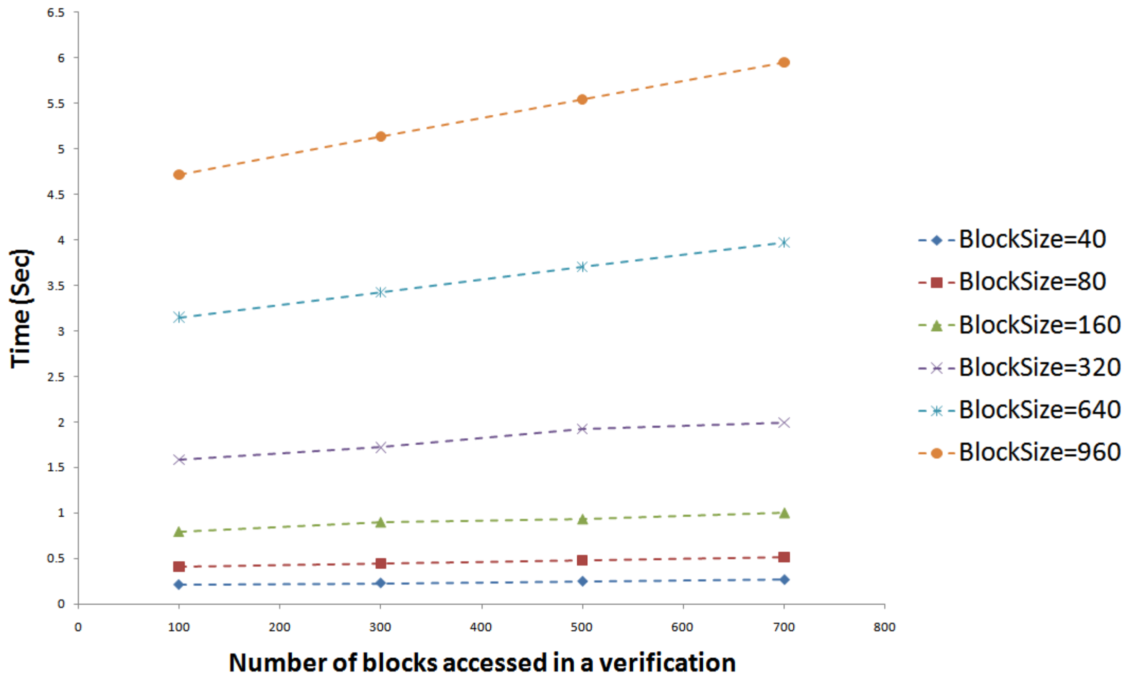


(b) Data preprocess time VS the block size. Each line is labeled with the size (in megabytes) of corresponding data file.

Figure 5.4: Computation Time of algorithms KeyGen and DEncode.



(a) Query latency VS the challenge size for a 128MB data file. Each line is labeled with the corresponding block size.



(b) Query latency VS the challenge size for a 512MB data file. Each line is labeled with the corresponding block size.

Figure 5.5: Computation Time of algorithm Prove and Verify.

5.6 Security Analysis of MAC scheme $\mathsf{S2}$

In this section, we give the security model for linearly predicate-homomorphic MAC scheme, introduce the Strong Diffie-Hellman Assumption, and then prove the security of our construction $\mathsf{S2}$ under this assumption.

5.6.1 Security Model for Linearly Predicate-Homomorphic MAC

Let $\mathsf{S} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Combine}, \mathsf{P\text{-}Sign}, \mathsf{Verify}, \mathsf{P\text{-}Verify})$ be a linearly predicate homomorphic MAC scheme as described in Section 5.2 (on page 55). We define the *existential forgery* security game $\mathsf{Game}^{\mathsf{CMA}}_{\mathsf{S},\mathcal{A}}(1^\lambda)$ between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ w.r.t. the MAC scheme S under *adaptive chosen message attack* model as below.

Setup.

The challenger $\mathcal{C}$ generates a pair of public-private key (spk, ssk) by running the key generating algorithm $\mathsf{KeyGen}(1^\lambda)$ with security parameter λ, and gives the public key spk to the adversary $\mathcal{A}$ and keeps the private key ssk securely. The challenger $\mathcal{C}$ maintains a state variable state, which will be used to assign unique index to each message to be signed.

Learning.

The adversary $\mathcal{A}$ can adaptively make queries, where each query is in one of the following forms:

- **SignQuery**($\mathtt{M}$): Given a message $\mathtt{M}$ chosen by the adversary $\mathcal{A}$, the challenger $\mathcal{C}$ chooses a unique index i based on the current value of state and updates state. The challenger $\mathcal{C}$ denotes the message $\mathtt{M}$ as $\mathtt{M}_i$ and responses the query with a signature $\sigma_i := \mathsf{Sign}(ssk, i, \mathtt{M}_i)$.

- **P-VerifyQuery**$(\mathtt{M}, \sigma, \{(i, \nu_i) : i \in C\}, r)$: Let the state variable $\mathbf{I}$ denote the set of all indices i's chosen by the challenger in answering all previous **SignQuery**. For each tuple $(\mathtt{M}, \sigma, \{(i, \nu_i) : i \in C\}, r)$ chosen by the adversary $\mathcal{A}$, if $C \subset \mathbf{I}$, then the challenger $\mathcal{C}$ responses with $b := \mathsf{P\text{-}Verify}(ssk, \mathtt{M}, \sigma, \{(i, \nu_i) : i \in C\}, r) \in \{\mathtt{accept}, \mathtt{reject}\}$. If $C \not\subset \mathbf{I}$, then the challenger does nothing.

Forge.

The adversary $\mathcal{A}$ outputs $(\mathtt{M}', \sigma', \{(i, \nu_i) : i \in C\}, r)$ with $C \subset \mathbf{I}$. Let the corresponding genuine output be

$$(\mathtt{M}, \sigma) := \mathsf{P\text{-}Sign}(spk,\ \mathsf{Combine}(spk, \{(i, \mathtt{M}_i, \sigma_i, \nu_i) : i \in C\}),\ r)$$

The adversary $\mathcal{A}$ wins the game if and only if

$$\mathsf{P\text{-}Verify}(ssk, \mathtt{M}', \sigma', \{(i, \nu_i) : i \in C\}, r) \ = \ \mathtt{accept} \text{ and } (\mathtt{M}', \sigma') \neq (\mathtt{M}, \sigma).$$

5.6.2 Assumption

Let p and $q = 2p + 1$ be prime. Denote the subgroup of quadratic residues in $\mathbb{Z}_q^*$ as $\mathcal{QR}_q$. The order of group $\mathcal{QR}_q$ is p. The Strong Diffie-Hellman Assumption [BB04, BB08] over the group $\mathcal{QR}_q$ is described as below.

Assumption 1 (m-SDH Assumption [BB04, BB08]) *Let p and $q = 2p + 1$ be prime, and $\mathcal{QR}_q$ be the subgroup of quadratic residues in $\mathbb{Z}_q^*$. Let g be a random generator of $\mathcal{QR}_q$. Let $\alpha \xleftarrow{\$} \mathbb{Z}_p$ be a random element from $\mathbb{Z}_p$. Let $T = (g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^{m-1}}) \in \left(\mathbb{Z}_q^*\right)^m$. Given as input a tuple (p, q, T), for any PPT adversary $\mathcal{A}$, the probability*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{SDH}} \stackrel{\text{def}}{=} \mathsf{Pr}\left[w = g^{1/(\alpha+c)} \text{ where } (c, w) := \mathcal{A}(p, q, T)\right]$$

is negligible in $\lambda = \log p$.

We remark that when the MAC scheme $\mathsf{S2}$ is alternatively instantiated using elliptic curve, the SDH Assumption over elliptic curve group is required.

5.6.3 The Linearly Predicate-Homomorphic MAC scheme $\mathsf{S2}$ is Secure

Theorem 5.3 **($\mathsf{S2}$ is Secure)** *Suppose the SDH Assumption 1 holds and* $\{\mathsf{PRF}\}$ *is a secure pseudorandom function family. Then the proposed MAC scheme* $\mathsf{S2}$ *is existentially unforgeable under adaptive chosen message attack. More precisely, for any PPT adversary* $\mathcal{A}$*, the advantage* $\mathsf{Adv}^{\mathsf{CMA}}_{\mathsf{S2},\mathcal{A}}$ *of* $\mathcal{A}$ *against* $\mathsf{S2}$ *is negligible in the security parameter, where* $\mathsf{Adv}^{\mathsf{CMA}}_{\mathsf{S2},\mathcal{A}}$ *is defined as below*

$$\mathsf{Adv}^{\mathsf{CMA}}_{\mathsf{S2},\mathcal{A}}(1^\lambda) \overset{\text{def}}{=} \mathsf{Pr}\left[\mathcal{A} \textit{ wins } \mathsf{Game}^{\mathsf{CMA}}_{\mathsf{S2},\mathcal{A}}(1^\lambda)\right]. \tag{5.28}$$

5.6.3.1 Outline of Proof of Theorem 5.3

We show that the MAC scheme $\mathsf{S2}$ is secure in two steps:

- Security in no-feedback setting: We prove the unforgeability in a simplified setting, where all decision bits (acceptance or rejection) are kept secret from the adversary.
- Security in feedback setting: We prove the security in the feedback setting, which is as stated in Theorem 5.3, based on the security in the simplified no-feedback setting.

5.6.3.2 $\mathsf{S2}$ is unforgeable in no-feedback setting.

Lemma 5.4 (Unforgeability in no-feedback setting) *In the no-feedback setting, where all decision bits (acceptance or rejection) are kept secret from the adversary in* $\mathsf{Game}^{\mathsf{CMA}}_{\mathsf{S2},\mathcal{A}}$ *w.r.t. scheme* $\mathsf{S2}$*, for any PPT adversary* $\mathcal{A}$*, there exist PPT adversaries* $\mathcal{B}_1$ *and* $\mathcal{B}_2$*, such that*

$$\begin{aligned} & \mathsf{Pr}\left[\mathcal{A} \textit{ wins } \mathsf{Game}^{\mathsf{CMA}}_{\mathsf{S2},\mathcal{A}} \textit{ in no-feedback setting }\right] \\ \leq\ & \mathsf{Adv}^{\mathsf{SDH}}_{\mathcal{B}_1} + \frac{1}{p-1} + N_{\mathsf{PRF}} \cdot \mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{B}_2}, \end{aligned} \tag{5.29}$$

where N_{PRF} is the number of distinct evaluations of the pseudorandom function PRF in $\mathsf{Game}^{\mathsf{CMA}}_{\mathsf{S2},\mathcal{A}}$ (N_{PRF} is equal to the number of **SignQuery***s made by $\mathcal{A}$ in $\mathsf{Game}^{\mathsf{CMA}}_{\mathsf{S2},\mathcal{A}}$), $\mathsf{Adv}^{\mathsf{SDH}}_{\mathcal{B}_1}$ is the probability that $\mathcal{B}_1$ breaks the SDH Assumption 1 and $\mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{B}_2}$ is the probability that $\mathcal{B}_2$ can distinguish the output of pseudorandom function PRF from true randomness.*

Proof of Lemma 5.4:

Game 1. The first game is the same as $\mathsf{Game}^{\mathsf{CMA}}_{\mathsf{S2},\mathcal{A}}$, except that all decision bits (acceptance or rejection) are kept secret from the adversary $\mathcal{A}$, i.e. the challenger in **Game 1** does not answer any **P-VerifyQuery** made by the adversary $\mathcal{A}$.

Let $(y', \phi', \sigma', \{(i, \nu_i) : i \in C\}, r)$ be the output of $\mathcal{A}$ in the **Forge** phase of the **Game 1**, and $(y, \phi, \sigma, \{(i, \nu_i) : i \in C\}, r)$ be the corresponding genuine output that shares the same values $\{(i, \nu_i) : i \in C\}$ and r. Let (spk, ssk) be the public-private key pair chosen by the challenger. By the definition of security game $\mathsf{Game}^{\mathsf{CMA}}_{\mathsf{S2},\mathcal{A}}$ and **Game 1**, the following claim can be derived straightforwardly.

Claim 5.6.1

$$\begin{aligned}
&\Pr\left[\mathcal{A} \textit{ wins } \mathsf{Game}^{\mathsf{CMA}}_{\mathsf{S2},\mathcal{A}} \textit{ in no-feedback setting}\right] \\
&= \Pr\left[\mathcal{A} \textit{ wins } \textbf{Game 1}\ \right] \\
&= \Pr\left[\mathsf{S2.P\text{-}Verify}(ssk, y', \phi', \sigma', \{(i, \nu_i) : i \in C\}, r) = \mathtt{accept} \wedge (y', \phi', \sigma') \neq (y, \phi, \sigma)\right]
\end{aligned}$$

Game 2. The second game is the same as **Game 1**, except that the pseudorandom function $\mathsf{PRF}_{\mathsf{seed}}(\cdot)$ in the scheme S2 is replaced by a simulator $\mathsf{PRF}^{\mathsf{Sim}}$, which outputs true randomness over the range of $\mathsf{PRF}_{\mathsf{seed}}$. Precisely, the function $\mathsf{PRF}^{\mathsf{Sim}}$ is evaluated in the following way:

- The challenger keeps a table, which is empty at the very beginning, to store all previous encountered input-output pairs $(v, \mathsf{PRF}^{\mathsf{Sim}}(v))$.

- Given an input v, the challenger lookups the table for v, if there exists an entry (v, u), then return u as output. Otherwise, choose u uniformly randomly from the range of $\mathsf{PRF}_{\mathsf{seed}}$, insert $(v, \mathsf{PRF}^{\mathsf{Sim}}(v) := u)$ into the table and return u as output.

Game 3. The third game is the same as **Game 2**, except that adversary $\mathcal{A}$ wins if and only if $\mathsf{S2.P\text{-}Verify}(ssk, y', \phi', \sigma', \{(i, \nu_i) : i \in C\}, r) = \mathtt{accept}$ and $(y', \psi', \sigma') \neq (y, \psi, \sigma)$ and $\sigma' = \sigma$.

Game 4. The fourth game is the same as **Game 2**, except that adversary $\mathcal{A}$ wins if and only if $\mathsf{S2.P\text{-}Verify}(ssk, y', \phi', \sigma', \{(i, \nu_i) : i \in C\}, r) = \mathtt{accept}$ and $(y', \psi', \sigma') \neq (y, \psi, \sigma)$ and $\sigma' \neq \sigma$.

From the definitions of **Game 2**, **Game 3** and **Game 4**, the following claim is straightforward.

Claim 5.6.2

$$\mathsf{Pr}[\mathcal{A} \textit{ wins } \textbf{Game 2}] = \mathsf{Pr}[\mathcal{A} \textit{ wins } \textbf{Game 3}] + \mathsf{Pr}[\mathcal{A} \textit{ wins } \textbf{Game 4}].$$

Claim 5.6.3 *If there is a non-negligible difference in a PPT adversary $\mathcal{A}$'s success probability between* **Game 1** *and* **Game 2***, then there exists another PPT adversary $\mathcal{B}$, which can break the security of the pseudorandom function* PRF. *More precisely,*

$$\Big|\mathsf{Pr}[\mathcal{A} \textit{ wins } \textbf{Game 1}] - \mathsf{Pr}[\mathcal{A} \textit{ wins } \textbf{Game 2}]\Big| \leq N_{\mathsf{PRF}} \cdot \mathsf{Adv}_{\mathcal{B}}^{\mathsf{PRF}},$$

where N_{PRF} is the number of distinct evaluations of pseudorandom function PRF *required to answer all* **SignQuery***s made by $\mathcal{A}$ (one* PRF *evaluation for one* **SignQuery***), and $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{PRF}}$ denotes the probability that $\mathcal{B}$ can distinguish the output of* PRF *from true randomness.*

The above Claim 5.6.3 can be proved using a standard hybrid argument [Gol06]. Here we save the details.

Claim 5.6.4 *If a PPT adversary $\mathcal{A}$ wins* **Game 3** *with non-negligible probability, then these exists a PPT algorithm $\mathcal{B}$ that can break the SDH Assumption 1. More precisely,*

$$\mathsf{Adv}_{\mathcal{B}}^{\mathsf{SDH}} \overset{\text{def}}{=} \mathsf{Pr}\left[\mathcal{B} \textit{ solves SDH problem}\right] \geq \mathsf{Pr}\left[\mathcal{A} \textit{ wins } \textbf{Game 3}\ \right]. \tag{5.30}$$

Proof of Claim 5.6.4: We construct a PPT algorithm $\mathcal{B}$, based on the adversary $\mathcal{A}$, to solve the SDH problem.

Given an input $(p, q, \{g^{\alpha^j}\}_{j=0}^{m-1})$ of the m-SDH problem, the algorithm $\mathcal{B}$ can simulate the **Game 3**, where $\mathcal{B}$ plays the role of challenger and $\mathcal{A}$ plays the role of adversary. The simulated game **Sim** is as below:

Sim.Setup The challenger $\mathcal{B}$ simulates the algorithm $\mathsf{S2.KeyGen}$:

- Choose τ, seed in the same way as in the algorithm $\mathsf{S2.KeyGen}$.
- Let public key $spk := (p, q, \{g^{\alpha^j}\}_{j=0}^{m-1})$ and conceptually set private key $ssk := (p, q, \mathsf{seed}, \alpha, \tau)$, where α is unknown to the challenger $\mathcal{B}$.

The challenger $\mathcal{B}$ gives the public key spk to the adversary $\mathcal{A}$.

Sim.Learning

Sim.SignQuery($\vec{\mathtt{M}}$): Given a message $\vec{\mathtt{M}} \in (\mathbb{Z}_p)^m$ chosen by the adversary $\mathcal{A}$, the challenger $\mathcal{B}$ chooses a unique index i for message $\vec{\mathtt{M}}$ based on a state variable and updates the state variable. The challenger $\mathcal{B}$ denotes the message $\vec{\mathtt{M}}$ as $\vec{\mathtt{M}}_i$ and generates the authentication MAC σ_i by randomly choosing an element from group $\mathbb{Z}_p$: $\sigma_i \xleftarrow{\$} \mathbb{Z}_p$. $\mathcal{B}$ provides σ_i to $\mathcal{A}$ as response.
Note: (1) There exists some unknown $s_i \in \mathbb{Z}_p$, such that $\sigma_i = s_i + \tau f_{\vec{\mathtt{M}}_i}(\alpha) \bmod p$. (2) s_i is uniformly randomly distributed over $\mathbb{Z}_p$ since σ_i is uniformly randomly distributed over $\mathbb{Z}_p$. (3) (Conceptually) Set the value of $\mathsf{PRF}^{\mathsf{Sim}}(i)$ as the unknown value s_i.

Sim.P-VerifyQuery: The challenger $\mathcal{B}$ does nothing. In the no-feedback setting, the challenger will not answer this query, and will keep all verification results (i.e. acceptance or rejection) secret from the adversary $\mathcal{A}$.

Sim.Forge The conditions that the adversary $\mathcal{A}$ wins the simulated **Game Sim** are the same as the conditions that $\mathcal{A}$ wins **Game 3**. That is, the adversary $\mathcal{A}$ wins if and only if

- $\mathcal{A}$ outputs a valid forgery $(y', \psi', \sigma', \{(i, \nu_i) : i \in C\}, r)$ as in the **Forge** phase of the game $\mathsf{Game}^{\mathsf{CMA}}_{\mathsf{S},\mathcal{A}}$.
- $\sigma' = \sigma$, where σ is from the corresponding genuine output $(y, \psi, \sigma, \{(i, \nu_i) : i \in C\}, r)$ which shares the same values $\{(i, \nu_i) : i \in C\}$ and r with the forgery output.

Note that $\mathcal{B}$ can compute the genuine output $(y, \psi, \sigma, \{(i, \nu_i) : i \in C\}, r)$ by itself. After interacting with the adversary $\mathcal{A}$ in **Game Sim**, the adversary $\mathcal{B}$ computes (c, w) as a solution to the SDH problem as below:

- If $y = y'$: find any $c \neq -r \in \mathbb{Z}_p$ and set $w := g^{1/(r+c)} \mod q$;
- If $y \neq y'$: set $c = -r$ and $w := \left(\frac{\psi}{\psi'}\right)^{1/(y'-y)} \mod q$.

Until now, the construction of the adversary $\mathcal{B}$ is complete.

Next, we want to show the following Claim 5.6.5 and Claim 5.6.6, which together imply the Claim 5.6.4:

Claim 5.6.5 $\mathsf{Pr}[\mathcal{A}$ *wins* **Game 3**$] = \mathsf{Pr}[\mathcal{A}$ *wins* **Game Sim**$]$.

Proof (Sketch proof of Claim 5.6.5): The above security game **Game Sim** between the challenger $\mathcal{B}$ and the adversary $\mathcal{A}$ is identical to the **Game 3** to the view of the adversary $\mathcal{A}$ (even if $\mathcal{A}$ is computationally unbounded), since all information that $\mathcal{A}$ obtains from the challenger in game **Sim** is identically distributed as information that $\mathcal{A}$ can obtain from the challenger in **Game 3**: All MAC values σ_i's received as the response of **SignQuery** are uniformly randomly distributed over $\mathbb{Z}_p$. Furthermore, the conditions that $\mathcal{A}$ wins in **Game Sim** and in **Game 3** are identical. Thus,

$$\mathsf{Pr}[\mathcal{A} \text{ wins } \textbf{Game 3}] = \mathsf{Pr}[\mathcal{A} \text{ wins } \textbf{Game Sim}].$$

□

Claim 5.6.6 *If $\mathcal{A}$ wins the* **Game Sim**, *i.e.* $\mathsf{S2.P\text{-}Verify}(ssk, y', \psi', \sigma', \{(i,\nu_i) : i \in C\}, r) = \mathtt{accept}$ *and* $(y', \psi', \sigma') \neq (y, \psi, \sigma)$ *and* $\sigma' = \sigma$*, then the output* (c, w) *of the adversary* $\mathcal{B}$ *is a correct solution to the SDH problem, i.e.* $g^{\frac{1}{\alpha+c}} = w$*. That is,*

$$\mathsf{Pr}[\mathcal{A} \textit{ wins } \textbf{Game Sim}] \leq \mathsf{Pr}[\mathcal{B} \textit{ solves } \text{SDH } \textit{problem}].$$

Proof of Claim 5.6.6: Since both the forgery output $(y', \psi', \sigma', \{(i, \nu_i) : i \in C\}, r)$ and the corresponding genuine output $(y, \psi, \sigma, \{(i, \nu_i) : i \in C\}, r)$ are accepted by the verifier algorithm $\mathsf{S2.P\text{-}Verify}$, the two tuples satisfy the Equation (5.9) (on page 61):

$$\psi^{\alpha - r} = g^{\tau^{-1}\left(\sigma - \sum_{i\in C} \nu_i \mathsf{PRF}_{\mathsf{seed}}(i)\right) \ - \ y} \mod q \tag{5.31}$$

$$\psi'^{\alpha - r} = g^{\tau^{-1}\left(\sigma' - \sum_{i\in C} \nu_i \mathsf{PRF}_{\mathsf{seed}}(i)\right) \ - \ y'} \mod q \tag{5.32}$$

Dividing Equation (5.31) with Equation (5.32), we obtain

$$\left(\frac{\psi}{\psi'}\right)^{\alpha - r} = g^{\tau^{-1}(\sigma - \sigma') \ + \ y' - y} = g^{\ y' - y} \mod q \qquad (\text{ Since } \sigma' = \sigma) \tag{5.33}$$

Recall that if $\mathcal{A}$ wins **Game Sim** (or **Game 3**), then $\sigma' = \sigma$.

Now we do a case analysis on whether y' is equal to y.

Case 1: $y' = y \mod p$. The equality that $y' = y$, implies $\psi' \neq \psi$, since $(y', \psi', \sigma') \neq (y, \psi, \sigma)$ and $\sigma' = \sigma$. Note that the verifier algorithm $\mathsf{S2.P\text{-}Verify}$ accepts the forgery output (genuine output respectively) only if ψ' (ψ respectively) is a quadratic residue modulo q. In the subgroup $\mathcal{QR}_q$ of quadratic residue modulo q, all elements, except unity element 1, have multiplicative order p modulo q. We know that $\psi'/\psi \neq 1$, so the element $\psi'/\psi \in \mathcal{QR}_q$ has multiplicative order p. Thus, Equation (5.33) and $y' = y \mod p$ together imply $\alpha = r \mod p$. Thus adversary $\mathcal{B}$'s output $(c, w = g^{1/(r+c)} = g^{1/(\alpha+c)})$ is a valid solution to the m-SDH problem.

Case 2: $y' \neq y \mod p$. Equation (5.33) and $y' \neq y \mod p$ together imply that $\alpha \neq r \mod p$. Recall that in case $y' \neq y$, the output of $\mathcal{B}$ to the SDH problem is

$(c = -r, w = \left(\frac{\psi}{\psi'}\right)^{1/(y'-y)} \mod q)$. Substituting $\frac{\psi}{\psi'}$ with $\left(w^{y'-y} \mod q\right)$ into the Equation (5.33), we have

$$w^{(y'-y)(\alpha-r)} = g^{y'-y} \mod q \tag{5.34}$$

Since $y' - y \neq 0 \mod p$ and $\alpha - r \neq 0 \mod p$, their inverses $1/(y'-y) \mod p$ and $1/(\alpha - r) \mod p$ exist. Therefore, the following equality can be derived from Equation (5.34):

$$w = \left(g^{y'-y}\right)^{\frac{1}{(y'-y)(\alpha-r)}} = g^{\frac{1}{\alpha-r}} \mod q \tag{5.35}$$

The above Equation (5.35) shows that the adversary $\mathcal{B}$'s output $(c = -r, w)$ is a valid solution to the SDH problem.

We remark that Case 2 in the above proof for Claim 5.6.6 borrows ideas from the proof in Kate, Zaverucha and Goldberg [KZG10]. □

By combining the results in Claim 5.6.5 and Claim 5.6.6, Claim 5.6.4 is proved:

$$\mathsf{Pr}[\mathcal{A} \text{ wins } \textbf{Game 3}] = \mathsf{Pr}[\mathcal{A} \text{ wins } \textbf{Game Sim}] \leq \mathsf{Pr}[\mathcal{B} \text{ solves SDH problem}].$$

□

Claim 5.6.7 *For any computationally unbounded adversary $\mathcal{A}$, after interacting in* **Game 4***, the probability that $\mathcal{A}$ finds the value of τ is $1/(p-1)$.*

Proof of Claim 5.6.7: In **Game 4**, the adversary $\mathcal{A}$ is given the public key $spk = (p, q, \{g^{\alpha^j} \mod q\}_{j=0}^{m-1})$ and authentication MAC values σ_i's for any message $\vec{\mathtt{M}}_i$ (vectors) chosen by the adversary $\mathcal{A}$:

$$\sigma_i = \mathsf{PRF}^{\mathsf{Sim}}(i) + \tau f_{\vec{\mathtt{M}}_i}(\alpha) \mod p.$$

The secret value τ is only involved in the MAC values σ_i's. Since in **Game 4**, the pseudorandom function PRF is replaced by a simulator $\mathsf{PRF}^{\mathsf{Sim}}$, which outputs uniform randomness over $\mathbb{Z}_p$, the MAC values σ_i's reveal absolutely *no* information to adversary $\mathcal{A}$ about the secret τ at all, although $\mathcal{A}$ is computationally unbounded. That is, the entropy of τ to the adversary $\mathcal{A}$ is unchanged before and after $\mathcal{A}$'s interaction with the challenger in the **Game 4**, and the probability that $\mathcal{A}$ can find τ is exactly $1/(p-1)$. Recall that τ is chosen at random from $\mathbb{Z}_p^*$.

$\square$

Claim 5.6.8 *For any computationally unbounded adversary $\mathcal{A}$,*

$$\mathsf{Pr}\left[\mathcal{A} \textit{ wins } \textbf{Game 4}\right] \le \frac{1}{p-1}.$$

Proof of Claim 5.6.8: Let $(y, \psi, \sigma, \{(i, \nu_i) : i \in C\}, r)$ and $(y', \psi', \sigma', \{(i, \nu_i) : i \in C\}, r)$ be as specified in **Game 1**. Suppose a computationally unbounded adversary $\mathcal{A}$ wins **Game 4**, i.e. $\mathsf{S2.P\text{-}Verify}(ssk, y', \phi', \sigma', \{(i, \nu_i) : i \in C\}, r) = \texttt{accept}$ and $(y', \psi', \sigma') \neq (y, \psi, \sigma)$ and $\sigma' \neq \sigma$.

Similar as the proof of Claim 5.6.6 (precisely Equation (5.33)), we have

$$\left(\frac{\psi}{\psi'}\right)^{\alpha - r} = g^{\tau^{-1}(\sigma - \sigma') \ + \ y' - y} \mod q \tag{5.36}$$

The computationally unbounded adversary $\mathcal{A}$ can find α from the public key *spk* by solving discrete log problem, and eventually find τ from the above Equation (5.36). However, by Claim 5.6.7, the probability that $\mathcal{A}$ finds τ has to be $1/p$. Consequently, the adversary $\mathcal{A}$ wins **Game 4** with probability

$$\mathsf{Pr}\left[\mathcal{A} \text{ wins } \textbf{Game 4}\right] \le \mathsf{Pr}\left[\mathcal{A} \text{ finds } \tau\right] = \frac{1}{p-1}.$$

Thus, the Claim 5.6.8 is proved. $\square$

In summary, we have showed that

$$\begin{aligned}
& \mathsf{Pr}\left[\mathcal{A} \text{ wins } \mathsf{Game}^{\mathsf{CMA}}_{\mathsf{S2},\mathcal{A}} \text{ in no-feedback setting}\right] \\
= \; & \mathsf{Pr}[\mathcal{A} \text{ wins } \textbf{Game 1}] \leq \mathsf{Pr}[\mathcal{A} \text{ wins } \textbf{Game 2}] + N_{\mathsf{PRF}} \cdot \mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{B}_2} \\
= \; & \Big(\mathsf{Pr}[\mathcal{A} \text{ wins } \textbf{Game 3}] + \mathsf{Pr}[\mathcal{A} \text{ wins } \textbf{Game 4}]\Big) + N_{\mathsf{PRF}} \cdot \mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{B}_2} \\
\leq \; & \mathsf{Adv}^{\mathsf{SDH}}_{\mathcal{B}_1} + \frac{1}{p-1} + N_{\mathsf{PRF}} \cdot \mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{B}_2}. \qquad (5.37)
\end{aligned}$$

Therefore, Lemma 5.4 is proved. □

5.6.3.3 S2 is unforgeable.

Lemma 5.5 (Unforgeable in feedback setting) *Let ξ be the probability that a PPT adversary $\mathcal{A}$ can win the security game $\mathsf{Game}^{\mathsf{CMA}}_{\mathsf{S2},\mathcal{A}}$ in the no-feedback setting. Then the probability $\mathsf{Adv}^{\mathsf{CMA}}_{\mathsf{S2},\mathcal{A}}$ that the adversary $\mathcal{A}$ can win the security game $\mathsf{Game}^{\mathsf{CMA}}_{\mathsf{S2},\mathcal{A}}$ in the feedback setting is as below*

$$\mathsf{Adv}^{\mathsf{CMA}}_{\mathsf{S2},\mathcal{A}} \leq \xi \; + \; \left(1 - (1-\xi)^{N_{ver}}\right) \leq \xi + N_{ver} \cdot \xi + o(\xi), \qquad (5.38)$$

where N_{ver} is the number of verification query **P-VerifyQuery***s made by the adversary $\mathcal{A}$ during the security game, and $o(\cdot)$ is the little-O notation.*

The proof of Lemma 5.5 is essentially identical to the proof of Lemma 4.5 (on page 42). We save the details.

Proof of Theorem 5.3: By Lemma 5.4 and Lemma 5.5, for any PPT adversary $\mathcal{A}$ which makes N_{PRF} number of **SignQuery** and N_{ver} number of **P-VerifyQuery** in the security game $\mathsf{Game}^{\mathsf{CMA}}_{\mathsf{S2},\mathcal{A}}$, the advantage of $\mathcal{A}$ against the MAC scheme S2 is

$$\mathsf{Adv}^{\mathsf{CMA}}_{\mathsf{S2},\mathcal{A}} = \mathsf{Pr}\left[\mathcal{A} \text{ wins } \mathsf{Game}^{\mathsf{CMA}}_{\mathsf{S2},\mathcal{A}}\right] \leq (N_{ver}+1)\left(\mathsf{Adv}^{\mathsf{SDH}}_{\mathcal{B}_1} + \frac{1}{p-1} + N_{\mathsf{PRF}} \cdot \mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{B}_2}\right),$$

which is negligible in $\lambda \approx \log p$. □

5.7 Security Analysis of $\mathcal{POR}$ scheme POS2

Theorem 5.6 *Suppose the SDH Assumption 1 holds and* $\{\mathsf{PRF}\}$ *is a secure pseudorandom function family. Then the proposed scheme* POS2 *is a sound Proof of Retrievability under Definition 6 (on page 26).*

Proof of Theorem 5.6: At first, we review Lemma 4.7 and Lemma 4.8 from Section 4.7.1 (on page 45) in the previous Chapter 4.
Lemma 4.7 (on page 45) *Let $\mathbb{X}$ and $\mathbb{Y}$ be two finite sets. Let $U_{\mathbb{X}}$ denote the uniform random variable over the domain $\mathbb{X}$ and $U_{\mathbb{Y}}$ denote the (independent) uniform random variable over the domain $\mathbb{Y}$. Consider any function $f : \mathbb{X} \times \mathbb{Y} \to \{0,1\}$. Let $\epsilon = \mathsf{Pr}[f(U_{\mathbb{X}}, U_{\mathbb{Y}}) = 1]$. For any constant $a \in (0, \frac{1}{2})$, define a set $\mathbf{S}_a = \{x \in \mathbb{X} : \mathsf{Pr}[f(x, U_{\mathbb{Y}}) = 1] \geq a\epsilon\}$. We have*

$$\mathsf{Pr}[U_{\mathbb{X}} \in \mathbf{S}_a] = \frac{|\mathbf{S}_a|}{|\mathbb{X}|} \geq \left(\frac{1-a}{\frac{1}{\epsilon} - a}\right) = \mathcal{O}(\epsilon).$$

Lemma 4.8 (on page 46) *Let κ be an integer. Let $\delta, \epsilon \in (0,1]$ be two real values and $\delta \geq \epsilon$. Let $t = \lceil \frac{\kappa}{\epsilon} \rceil$. Independently sample t number of values $r_1, \ldots, r_t$ from $\{0,1\}$ under the Bernoulli distribution with probability δ. Let d be a positive integer and $d \leq \kappa^c$ for some real valued constant $c \in (0,1)$. Then with overwhelming high probability (w.r.t. κ), there exists d distinct indices $i_1, i_2, \ldots, i_d \in [1,t]$ such that $\forall j \in [1,d], r_{i_j} = 1$.*

Extractor Strategy In the **Retrieve** phase of the $\mathcal{POR}$ security game between an adversary $\mathcal{A}$ and a challenger, the challenger runs $\zeta = \mathcal{O}(t^3)$ number of verifiability interactions with the adversary $\mathcal{A}$, where $\mathcal{A}$ can answer each query correctly with probability ϵ and $t = \lceil \frac{n\lambda}{a^2\epsilon} \rceil$ for some real constant $a \in (0, \frac{1}{2})$ (say $a = \frac{1}{3}$). We emphasize that the encoded file in the security game consists of n file blocks $\vec{\mathsf{F}}_u \in (\mathbb{Z}_p)^m$, $u \in [0, n-1]$. Here, we assume $m \leq n$ (In the alternative case that $m > n$, we will set $t = \lceil \frac{m\lambda}{a^2\epsilon} \rceil$).

A challenge query consists of three parts: a subset $C \subset [1,n]$ of size ℓ, a weights vector $\vec{\boldsymbol{\nu}} \in (\mathbb{Z}_p)^\ell$, and a value $r \in \mathbb{Z}_p$. Let $\mathbb{C}$, $\mathbb{W}$ and $\mathbb{R}$ denote the domain of $C, \vec{\boldsymbol{\nu}}$ and $\vec{\boldsymbol{r}} = (r^0, r^1, \dots, r^{m-1}) \in \mathbb{Z}_p^m$, respectively. Recall that in a verification, given a challenge query $(C, \vec{\boldsymbol{\nu}}, \vec{\boldsymbol{r}}) \in \mathbb{C} \times \mathbb{W} \times \mathbb{R}$, where $C = \{i_1, i_2, \dots, i_\ell\} \subset [0, n-1]$, $0 \le i_1 < i_2 < i_3 \dots i_\ell < n$, $\vec{\boldsymbol{\nu}} = (\nu_{i_1}, \nu_{i_2}, \dots, \nu_{i_\ell}) = (\nu^0, \nu^1, \dots, \nu^{\ell-1}) \in (\mathbb{Z}_p^*)^m$, and $\vec{\boldsymbol{r}} = (r^0, r^1, \dots, r^{m-1}) \in (\mathbb{Z}_p^*)^m$, an honest prover will return a value y together with proof (ψ, σ), where y is computed as below

$$y = \left\langle \sum_{i \in C} \nu_i \vec{\mathsf{F}}_i,\ \vec{\boldsymbol{r}} \right\rangle \in \mathbb{Z}_p, \qquad \vec{\mathsf{F}}_i \in (\mathbb{Z}_p)^m \text{ is the } i\text{-th file block.}$$

The challenger issues queries as in Algorithm 1.

Algorithm 1 Collect responses from adversary through $\mathcal{O}(t^3)$ number of verification interactions.

```
1: Initiate a three-dimensional array $f : \mathbb{C} \times \mathbb{W} \times \mathbb{R} \to \{0,1\}$.
2: for each $i \in [1,t]$ do
3:    Choose a subset $C_i \in \mathbb{C}$ at random
4:    for each $j \in [1,t]$ do
5:       Choose a weight vector $\vec{\boldsymbol{\nu}}_{i,j} \in \mathbb{W}$ at random
6:       for each $k \in [1,t]$ do
7:          Choose $\vec{\boldsymbol{r}}_{i,j,k} \in \mathbb{R}$ at random
8:          Send $(C_i, \vec{\boldsymbol{\nu}}_{i,j}, \vec{\boldsymbol{r}}_{i,j,k})$ as challenge query to the adversary $\mathcal{A}$ and get response
            $(y_{i,j,k}, \psi_{i,j,k}, \sigma_{i,j,k})$
9:          if the response $(y_{i,j,k}, \psi_{i,j,k}, \sigma_{i,j,k})$ is accepted by the verifier then
10:            Set $f(C_i, \vec{\boldsymbol{\nu}}_{i,j}, \vec{\boldsymbol{r}}_{i,j,k}) = 1$
11:         else
12:            Set $f(C_i, \vec{\boldsymbol{\nu}}_{i,j}, \vec{\boldsymbol{r}}_{i,j,k}) = 0$
```

Next, the challenger finds the set $\mathbf{I}$ of all $i \in [1,t]$, which satisfy this property: there exits a set $\mathbf{J}_i \subset [1,t]$ and sets $\mathbf{K}_{i,j} \subset [1,t], j \in \mathbf{J}_i$, such that (1) $\forall j \in \mathbf{J}_i, \forall k \in \mathbf{K}_{i,j}$, $f(C_i, \vec{\boldsymbol{\nu}}_{i,j}, \vec{\boldsymbol{r}}_{i,j,k}) = 1$; (2) $|\mathbf{J}_i| \ge \ell$ and $\forall j \in \mathbf{J}_i, |\mathbf{K}_{i,j}| \ge m$.

For each $i \in \mathbf{I}$, for each $j \in \mathbf{J}_i$, for each $k \in \mathbf{K}_{i,j}$, adversary's response $(y_{i,j,k}, \psi_{i,j,k}, \sigma_{i,j,k})$ for query $(C_i, \vec{\boldsymbol{\nu}}_{i,j}, \vec{\boldsymbol{r}}_{i,j,k})$ is accepted. Since the underlying MAC scheme $\mathsf{S2}$ is unforgeable (Theorem 5.3), with overwhelming high probability, we have $\forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i, \forall k \in$

$\mathbf{K}_{i,j}$,

$$\left\langle \sum_{u \in C_i} \nu_{i,j,u} \vec{\mathtt{F}}_u,\ \vec{\boldsymbol{r}}_{i,j,k} \right\rangle = y_{i,j,k} \in \mathbb{Z}_p, \text{where vector } (\ldots \nu_{i,j,u} \ldots)_{u \in C_i} = \vec{\boldsymbol{\nu}}_{i,j}.$$

Thus the linear combination $\sum_{u \in C_i} \nu_{i,j,u} \vec{\mathtt{F}}_u$ is a solution of the below linear equation system in unknown $\vec{\boldsymbol{x}}$

$$\begin{pmatrix} \vdots \\ \vec{\boldsymbol{r}}_{i,j,k} \\ \vdots \end{pmatrix}_{\substack{k \in \mathbf{K}_{i,j} \\ \text{ordered by increasing } k}} \times \vec{\boldsymbol{x}}^{\top} = \begin{pmatrix} \vdots \\ y_{i,j,k} \\ \vdots \end{pmatrix}_{\substack{k \in \mathbf{K}_{i,j} \\ \text{ordered by increasing } k}} \tag{5.39}$$

In turn, the file blocks $\vec{\mathtt{F}}_u, u \in C_i$, can be recovered from (at least) ℓ number of linear combinations $\sum_{u \in C_i} \nu_{i,j,u} \vec{\mathtt{F}}_u$, $j \in \mathbf{J}_i$, by solving the below linear equation system

$$\begin{pmatrix} \vdots \\ \vec{\boldsymbol{\nu}}_{i,j} \\ \vdots \end{pmatrix}_{\substack{j \in \mathbf{J}_i \\ \text{ordered by increasing } j}} \times \begin{pmatrix} \vdots \\ \vec{\mathtt{F}}_u \\ \vdots \end{pmatrix}_{\substack{u \in C_i \\ \text{ordered by increasing } u}} = \begin{pmatrix} \vdots \\ \sum_{u \in C_i} \nu_{i,j,u} \vec{\mathtt{F}}_u \\ \vdots \end{pmatrix}_{\substack{j \in \mathbf{J}_i \\ \text{ordered by increasing } j}} \tag{5.40}$$

Therefore, for each $i \in \mathbf{I}$, the challenger can recover file blocks $\mathtt{F}_u$'s with $u \in C_i$. If the challenger recovers ρn number of file blocks, then he/she can recover the original file $\mathtt{F}$ using the error erasure decoding algorithm.

We emphasize that the coefficient matrix in each of above linear equation systems in Equation (5.39) and (5.40) is a *vandermonde matrix* [MS58], since in $\mathsf{POS2}$, verifiers choose vectors $\vec{\boldsymbol{r}}$ and $\vec{\boldsymbol{\nu}}$ in the form $(z^0, z^1, z^2, \ldots)$. Consequently, the solution to each of the above linear equation system is unique.

Analysis of Extractor Strategy Recall that the adversary can answer correctly a random chosen query from domain $\mathbb{C} \times \mathbb{W} \times \mathbb{R}$ with probability at least ϵ. That is $\mathsf{Pr}[f(\mathtt{C}, \mathtt{W}, \mathtt{R}) = 1] \geq \epsilon$, where $(\mathtt{C}, \mathtt{W}, \mathtt{R})$ is chosen at random from $\mathbb{C} \times \mathbb{W} \times \mathbb{R}$. Recall that constant $a = \frac{1}{3}$. We say a set $C \in \mathbb{C}$ is *good*, if $\mathsf{Pr}[f(C, \mathtt{W}, \mathtt{R}) = 1] \geq a\epsilon$, where $(\mathtt{W}, \mathtt{R})$ is chosen at random from $\mathbb{W} \times \mathbb{R}$. By the Lemma 4.7, a random chosen $C \in \mathbb{C}$

is good with probability at least $\mathcal{O}(\epsilon)$ (more precisely at least $\frac{1}{2}\epsilon$). For a good C, we say $W \in \mathbb{W}$ is *good* w.r.t. C, if $\mathsf{Pr}[f(C, W, \mathtt{R}) = 1] \geq a^2\epsilon$, where C and W are fixed and $\mathtt{R}$ is chosen at random from $\mathbb{R}$. By Lemma 4.7, a randomly chosen W is *good* w.r.t. a good C with probability at least $\mathcal{O}(a\epsilon)$ (more precisely at least $\frac{1}{6}\epsilon$).

By Lemma 4.8 with $\kappa = n\lambda$ and $d = \rho n \leq n < \kappa^c$ for some constant[3] $c \in (\frac{\ln n}{\ln(n\lambda)}, 1)$, with overwhelming high probability, there are at least $d = \rho n$ *good* C_i's among $\{C_i : i \in [1, t]\}$. For each good C_i, by Lemma 4.8 with $\kappa = n\lambda$ and $d = \ell \leq n < \kappa^c$, with overwhelming high probability, there are at least ℓ good W_j's w.r.t. C_i among $\{\vec{\boldsymbol{\nu}}_{i,j} : j \in [1, t]\}$. For each good $\vec{\boldsymbol{\nu}}_{i,j}$ w.r.t. C_i, by Lemma 4.8 with $\kappa = n\lambda$ and $d = m \leq n < \kappa^c$, with overwhelming high probability, there are at least m number of $\vec{\boldsymbol{r}}_{i,j,k}$ among $\{\vec{\boldsymbol{r}}_{i,j,k} : k \in [1, t]\}$, such that $f(C_i, \vec{\boldsymbol{\nu}}_{i,j}, \vec{\boldsymbol{r}}_{i,j,k}) = 1$. Thus in the above extractor algorithm, the challenger can find sufficient number of correct responses from the adversary to form linear equation systems, with overwhelming high probability. Since the domain sizes of $\mathbb{C}$, $\mathbb{W}$ and $\mathbb{R}$ are all exponentially large, with overwhelming high probability, all good C_i's (respectively, $\vec{\boldsymbol{\nu}}_{i,j}$ or $\vec{\boldsymbol{r}}_{i,j,k}$) are distinct. $\square$

5.8 Summary

This chapter presented a predicate-homomorphic MAC scheme $\mathsf{S2}$, which allows any third party to compute MAC values for many *short* messages $\mathtt{M}_i$'s from a long message $\mathtt{M}$ and its MAC value σ using the public key, as long as two messages $(\mathtt{M}_i, \mathtt{M})$ satisfy some pre-determined predicates. Furthermore, each short message $\mathtt{M}_i$ is a codeword of the long message $\mathtt{M}$, such that $\mathtt{M}$ can be recovered from a certain number of distinct messages $\mathtt{M}_i$'s. Based on homomorphic MAC scheme $\mathsf{S2}$, a Proofs of Retrievability scheme $\mathsf{POS2}$ is constructed, which is very efficient in communication and storage. Empirical study confirmed that $\mathsf{POS2}$ is practical in computation. $\mathsf{POS2}$ is also compared with the existing works in various aspects. Full security proof for $\mathsf{S2}$ and $\mathsf{POS2}$ are provided, under Strong Diffie-Hellman assumption and the assumption of

[3]Note that the file size n is upper bounded by a polynomial in λ. Thus there exists a positive integer z such that $n \leq \lambda^z$. We can set constant $c = \frac{z}{z+1} \geq \frac{\ln n}{\ln(n\lambda)}$.

existence of secure pseudorandom function. The security property of $\mathsf{S2}$ is proved in two steps: (1) in the first step, security of $\mathsf{S2}$ is proved in a simplified setting, where every acceptance/rejection decision of each verification is kept secret from the adversary; (2) in the second step, security of $\mathsf{S2}$ is proved in the original setting, using the result in the first step.

Chapter 6

Provable Data Possession

Provable Data Possession ($\mathcal{PDP}$) is an alternative formulation of proofs of storage proposed by Ateniese *et al.* [ABC$^+$07]. This chapter will propose a proofs of storage scheme, which we refer to as $\mathsf{POS3}$. The proposed scheme $\mathsf{POS3}$ improves Ateniese *et al.* [ABC$^+$07, ABC$^+$11b] in computation aspect, and will be proved under $\mathcal{PDP}$ formulation.

6.1 Overview

Ateniese *et al.* [ABC$^+$07] proposed the first Provable Data Possession ($\mathcal{PDP}$ for short) scheme. Their scheme is very efficient in communication and storage: the size of a proof is a small multiple of the security parameter, and the storage overhead due to authentication tags is a fraction[1] of the size of the original data. However, their scheme requires a large number of modular exponentiation in both setup phase and verification phase, and is thus relative expensive in computation.

In this chapter, we will propose a new $\mathcal{PDP}$ construction named $\mathsf{POS3}$, which requires no modular exponentiation in the setup phase and a smaller number of modular exponentiations in verification phase, without sacrificing in communication or storage aspects.

[1]This fraction is a configurable system parameter.

6.1.1 A Brief Description of proofs of storage scheme $\mathsf{POS3}$

Setup.

Suppose Alice wants to backup her file $\mathtt{F}$ into a cloud storage server provided by Bob. Alice encodes file $\mathtt{F}$ with some error erasure code to obtain data blocks $(\mathtt{F}_0, \ldots, \mathtt{F}_{n-1})$. Alice chooses a RSA modulus $N = pq$, a secret seed, denoted as seed, of a pseudo-random function PRF, and a secret number τ. Let $\phi(N) = (p-1)(q-1)$. With the private key $sk = (\phi(N), \mathsf{seed}, \tau)$, Alice produces an authentication tag σ_i for each block $\mathtt{F}_i$:

$$\sigma_i := \tau\mathtt{F}_i \; + \; \mathsf{PRF}_{\mathsf{seed}}(i) \mod \phi(N). \tag{6.1}$$

It is worthy to point out that the generated authentication tag σ_i is much shorter than a data block $\mathtt{F}_i$. At the end of setup, Alice sends data blocks and tags $\{(i, \mathtt{F}_i, \sigma_i) : i \in [0, n-1]\}$ together with a public key $pk = (N)$ to Bob.

Verification.

Later, Alice may remotely verify the integrity of her data file stored with Bob periodically. In each verification session, Alice randomly selects a subset $C \subset [0, n-1]$ of indices and selects a random weights ν_i for each $i \in C$. Alice sends $\{(i, \nu_i) : i \in C\}$ as challenge to Bob. Bob then finds all data blocks $\mathtt{F}_i$'s and authentication tags σ_i's with index $i \in C$, and applies the linear homomorphism to compute an aggregated message-tag pair $(\mathtt{M}, \sigma)$ as below:

$$\mathtt{M} := \sum_{i \in C} \nu_i \mathtt{F}_i; \tag{6.2}$$

$$\sigma := \sum_{i \in C} \nu_i \sigma_i. \tag{6.3}$$

We emphasize that the above two equations are computed over integer domain, and thus the bit-length of the linear combination $\mathtt{M}$ (the aggregated authentication tag σ, respectively) is slightly larger than a data block $\mathtt{F}_i$ (an authentication tag σ_i, respectively).

Instead of sending the large message block $\mathtt{M}$ together with authentication tag σ

directly to the verifier Alice, Bob is able to produce a *shorter* message-tag pair with the help of Alice. Alice generates a pair of fresh public token pt and secret token st per each verification, where the public token pt is sent to the prover Bob and the secret token st is kept private. With pt and $(\mathtt{M}, \sigma)$, Bob is able to generate a *shorter* message-tag pair, which can be verified by the verifier Alice with the private key and the secret token st.

Illustration Picture

Figure 6.1 illustrates the scheme $\mathsf{POS3}$ briefed above. In Figure 6.1, a rectangle represents a data block, and a circle represents a short authentication tag corresponding to the data block represented by the rectangle that lies above. Those shaded rectangles represent data blocks that are generated by the error erasure code. In our scheme $\mathsf{POS3}$, a data block is treated as a single large integer (much larger than the RSA modulus N). Figure 6.1 shows an example where a data block is about three times larger than a tag, by dividing each rectangle with dashed lines.

In a verification, a subset of three pairs of blocks and tags are selected, which are aggregated into a single pair of block and tag through linear homomorphism. Since the linear combination is computed over integer domain, the aggregated block (tag, respectively) is slightly larger than an original data block (tag, respectively). With the help of the public token pt provided by the verifier, a shorter message-tag pair can be generated from the long aggregated message-tag pair. The verifier can verify the short message-tag pair using a secret token st.

6.1.2 Organization

The rest of this chapter is organized as below. The next Section 6.2 describes the definition of $\mathcal{PDP}$. Section 6.3 presents the construction of our $\mathcal{PDP}$ scheme $\mathsf{POS3}$. Then we analyze the performance of proposed scheme in Section 6.4 and security in Section 6.5. At the end, Section 6.6 summarizes this chapter.

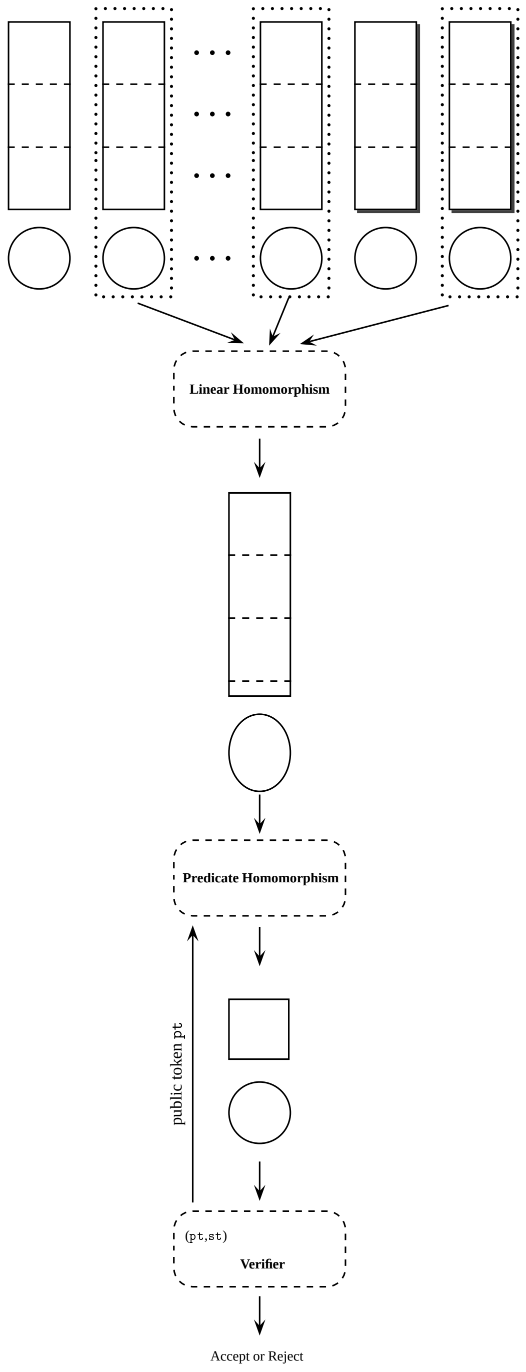


Figure 6.1: An Efficient $\mathcal{PDP}$ scheme $\mathsf{POS3}$. Detailed explanation is in the paragraph with title "Illustration Picture".

6.2 Provable Data Possession: Definition and Formulation

A $\mathcal{PDP}$ [ABC$^+$07] scheme consists of five polynomial algorithms ($\mathsf{KeyGen}, \mathsf{DEncode}$, $\mathsf{Challenge}$, $\mathsf{Prove}, \mathsf{Verify}$), which are described as below.

- $\mathsf{KeyGen}(1^\lambda) \to (pk, sk)$: Given security parameter λ, the probabilistic key generating algorithm outputs a public-private key pair (pk, sk).

- $\mathsf{DEncode}(sk, \mathtt{F}) \to (\mathsf{id}_\mathtt{F}, \hat{\mathtt{F}}, n)$: Given the private key sk and a data file $\mathtt{F}$ as input, the data encoding algorithm $\mathsf{DEncode}$ produces a unique identifier $\mathsf{id}_\mathtt{F}$, an encoded file $\hat{\mathtt{F}}$, and file size n (in term of number of blocks).

- $\mathsf{Challenge}(sk, \mathsf{id}, n) \to (\mathsf{pt}, \mathsf{st}, \mathsf{Chall})$: The probabilistic algorithm $\mathsf{Challenge}$ takes as input the private key sk, a file identifier id and the file size n (in term of number of blocks), and outputs a public token pt, a private token st and a query Chall.
  Note: (1) The public/secret token pair (pt*,* st*) is generated independently per each execution of algorithm* $\mathsf{Challenge}$*. (2) In both Ateniese* et al. *[ABC$^+$07] and the scheme* POS3 *that will be presented later in this chapter,* $\mathsf{Chall} = \{(i, \nu_i) : i \in C \subset [0, n-1]\}$ *is just a set of pairs of index i (in the range* $[0, n-1]$*) and weight ν_i (from some group) with $i \in C$, and has no secret information involved.*

- $\mathsf{Prove}(pk, \mathsf{id}_\mathtt{F}, \hat{\mathtt{F}}, \mathsf{pt}, \mathsf{Chall}) \to \psi$: Given as input the public key pk, an identifier $\mathsf{id}_\mathtt{F}$, an encoded file $\hat{\mathtt{F}}$, a public token pt and a challenge query Chall, the prover algorithm Prove produces a proof ψ.

- $\mathsf{Verify}(sk, \mathsf{id}_\mathtt{F}, \mathsf{st}, \mathsf{Chall}, \psi) \to$ `accept` or `reject`: Given the private key sk, an identifier $\mathsf{id}_\mathtt{F}$, a secret token st, a challenge query Chall, and a proof ψ as input, the deterministic verifier algorithm Verify will output either `accept` or `reject`.

Remark 5 *Compared to $\mathcal{POR}$, in the above description for $\mathcal{PDP}$, the prover algorithm* Prove *takes a public token* pt *as an additional input and the verifier algorithm*

Verify *takes a corresponding secret token* st *as an additional input, where the public/secret token pair* $(\mathsf{pt}, \mathsf{st})$ *is generated online by the verifier for each verification session.*

6.3 POS3: An Efficient $\mathcal{PDP}$ Scheme

In this section, we will construct the scheme $\mathsf{POS3}$ which is briefed previously in Section 6.1.

6.3.1 Construction of POS3

The description of scheme $\mathsf{POS3} = (\mathsf{KeyGen}, \mathsf{DEncode}, \mathsf{Challenge}, \mathsf{Prove}, \mathsf{Verify})$ is as below.

$\underline{\mathsf{POS3.KeyGen}(1^\lambda) \to (pk, sk)}$

Choose at random a λ bits RSA modulus $N = pq$, such that all of $p, q, p' = (p-1)/2$ and $q' = (q-1)/2$ are primes and the bit-lengths of p and q are the same. Let $\phi(N) = (p-1)(q-1) = 4p'q'$. Let $\mathcal{QR}_N$ denote the subgroup of quadratic residues modulo N. Choose at random a generator g of the subgroup $\mathcal{QR}_N$. Choose at random $\tau \overset{\$}{\leftarrow} \mathbb{Z}_{\phi(n)}$. Choose at random a seed, denoted as seed, from the key space of the pseudorandom function family $\{\mathsf{PRF}_{\mathsf{seed}} : \{0,1\}^{2\lambda} \to \mathbb{Z}_{\phi(N)}\}$. The public key is $pk = (N, g)$ and the private key is $sk := (g, p, q, \tau, \mathsf{seed})$.
Note: Size of subgroup $\mathcal{QR}_N$ [Gol06] is equal to $\frac{1}{4}\phi(N) = p'q'$.

$\underline{\mathsf{POS3.DEncode}(sk, \mathtt{F}) \to (\mathsf{id}, \hat{\mathtt{F}}, n)}$

Let $\rho \in (0,1)$ be a system parameter. Apply rate-ρ error erasure code on data file $\mathtt{F}$ to generate blocks $(\mathtt{F}_0, \ldots, \mathtt{F}_{n-1})$, such that each block $\mathtt{F}_i \in \{0,1\}^{m\lambda}$ and any ρn number of blocks $\mathtt{F}_i$'s can recover the original file $\mathtt{F}$. Choose a unique identifier $\mathsf{id} \in \{0,1\}^\lambda$ for the file $\mathtt{F}$. For each data block $\mathtt{F}_i, i \in [0, n-1]$, compute an authentication tag σ_i:

$$\sigma_i := \tau \mathtt{F}_i + \mathsf{PRF}_{\mathsf{seed}}(\mathsf{id} \| i) \mod \phi(N). \tag{6.4}$$

The encoded file is $\hat{\mathtt{F}} = \{(i, \mathtt{F}_i, \sigma_i) : i \in [0, n-1]\}$. Send $(\mathsf{id}, \hat{\mathtt{F}})$ to the cloud storage server, and keep (id, n) in the local storage. Output $(\mathsf{id}, \hat{\mathtt{F}}, n)$.

<u>$\mathsf{POS3.Challenge}(sk, \mathsf{id}, n) \rightarrow (\mathsf{pt}, \mathsf{st}, \{(i, \nu_i) : i \in C\})$</u>

Choose at random a secret value $d \xleftarrow{\$} \mathbb{Z}^*_{\phi(N)}$, and computes $g_d := g^d \mod N$. The public token is $\mathsf{pt} := g_d$ and the secret token is $\mathsf{st} := d$. Chooses a subset $C \subset [0, n-1]$ with size $|C| = \ell$ at random and choose weight $\nu_i \xleftarrow{\$} \mathbb{Z}^*_{\phi(N)}$ at random for each $i \in C$. Output $(\mathsf{pt}, \mathsf{st}, \{(i, \nu_i) : i \in C\})$.

<u>$\mathsf{POS3.Prove}(pk, \mathsf{id}, \hat{\mathtt{F}}, \mathsf{pt}, \{(i, \nu_i) : i \in C\}) \rightarrow (\psi_1, \psi_2)$</u>

Receive $(\mathsf{id}, \mathsf{pt}, \{(i, \nu_i) : i \in C\})$ from the verifier as the challenge. Find the encoded file $\hat{\mathtt{F}}$ corresponding to the identifier id. Find all selected blocks $\mathtt{F}_i$'s and tags σ_i's with index $i \in C$, and compute (π_1, π_2) as below over integer domain:

$$\pi_1 := \sum_{i \in C} \nu_i \mathtt{F}_i; \tag{6.5}$$

$$\pi_2 := \sum_{i \in C} \nu_i \sigma_i. \tag{6.6}$$

Compute (ψ_1, ψ_2) from (π_1, π_2) using the public key $pk = (N, g)$ and the public token $\mathsf{pt} = g_d$ as below

$$\psi_1 := g_d^{\pi_1} \mod N; \tag{6.7}$$

$$\psi_2 := g^{\pi_2} \mod N. \tag{6.8}$$

Send (ψ_1, ψ_2) to the verifier. Output (ψ_1, ψ_2).

$\underline{\mathsf{POS3.Verify}(sk, \mathsf{id}, \mathsf{st}, \{(i, \nu_i) : i \in C\}, \psi_1, \psi_2) \to \texttt{accept}\ \textbf{or}\ \texttt{reject}}$

With the private key $sk = (g, p, q, \tau, \mathsf{seed})$ and the secret token $\mathsf{st} = d$, check whether (1) $\psi_1 \in \mathcal{QR}_N$ is a quadratic residue modulo N and (2) the following equality holds.

$$(\psi_1)^\tau \overset{?}{=} \left(\frac{\psi_2}{g^{\sum_{i \in C} \nu_i \mathsf{PRF}_{\mathsf{seed}}(\mathsf{id} \| i)}} \right)^d \mod N \tag{6.9}$$

If both verifications succeed, then output `accept`; otherwise output `reject`.

Remark 6

- *We remark that in the above scheme, we can change the proof from (ψ_1, ψ_2) to $(\psi_1, \textbf{\textsf{SHA256}}(\psi_2))$ to reduce the size of response/proof from 2λ bits to $(\lambda + 256)$ bits using a secure hash function* **SHA256** *[NIS02], similar to Ateniese* et al. *[ABC+07].*

- *Like the first $\mathcal{PDP}$ scheme proposed by Ateniese [ABC+07] and different from previous* POS1 *and* POS2*, in our construction* POS3*, the prover requires a public token to generate a short proof and the verifier requires a corresponding secret token to verify the short proof, where this pair of public-secret tokens are generated by the verifier online per each verification. Therefore, the underlying homomorphic cryptography component of the* POS3 *is not a MAC scheme, and we do not separate it out as a standalone primitive.*

6.3.2 Completeness

In the above scheme POS3, if the response (ψ_1, ψ_2) is generated by an honest prover from intact data blocks and authentication tags, then the verifier always accepts (ψ_1, ψ_2), since

- $\psi_1 = g_d^{\pi_1} = g^{d\pi_1} \mod N \in \mathcal{QR}_N$ (recall that $g \in \mathcal{QR}_N$);

- The RHS (right hand side) of Equation (6.9) is

$$\begin{aligned}
RHS &= \left(\frac{g^{\pi_2}}{g^{\sum_{i\in C}\nu_i \mathsf{PRF}_{\mathsf{seed}}(\mathsf{id}\|i)}}\right)^d \\
&= \left(\frac{g^{\sum_{i\in C}\nu_i\left(\mathsf{PRF}_{\mathsf{seed}}(\mathsf{id}\|i)+\tau \mathtt{F}_i\right)}}{g^{\sum_{i\in C}\nu_i \mathsf{PRF}_{\mathsf{seed}}(\mathsf{id}\|i)}}\right)^d \\
&= \left(g^{\tau\sum_{i\in C}\nu_i \mathtt{F}_i}\right)^d = \left(g^{\tau\pi_1}\right)^d = \left(g^{d\pi_1}\right)^\tau \\
&= \left(g_d^{\pi_1}\right)^\tau = (\psi_1)^\tau = LHS \pmod{N}.
\end{aligned}$$

6.4 Performance Analysis

The proposed scheme POS3 is efficient in storage, communication and computation. The storage overhead due to authentication tags is $1/m$ of the file size (after error erasure encoding). The proof size in a verification is 2λ bits. In the setup, the data encoding algorithm DEncode requires n number of pseudorandom function evaluations, and modular additions/multiplications. In a verification session, the computation of the prover algorithm Prove is dominated by the exponentiation with large integer exponent in Equation (6.7), which is equivalent to $(m+2)$ number of group exponentiation in $\mathbb{Z}_N^*$ with exponent in $\mathbb{Z}_N$. The verifier algorithm Verify requires one modular division, three modular exponentiation in $\mathbb{Z}_N^*$, ℓ number of additions/multiplications in $\mathbb{Z}_{\phi(N)}$, and ℓ number of pseudorandom function evaluations, where $\ell = |C|$ is the number of indices selected during a verification.

6.4.1 Comparison

We compare the proposed scheme POS3 and Ateniese *et al.* [ABC+07, ABC+11b] in Table 6.1 in the setting specified in the below example.

Example 3 *After erasure encoding, the file size is 1GB, block size is* $m = 100$*, and storage overhead due to authentication tags is about 10MB for both schemes. For both schemes, we assume that, during a verification, the challenge query* $\{(i, \nu_i) : i \in C\}$

Table 6.1: Comparison among Ateniese *et al.* [ABC+07, ABC+11b] and POS3 and POS2 w.r.t. a 1GB file. The setting is described in Example 3.

Scheme	Group Element Size	Communication bits	Computation (Data Preprocess)	Computation (Prove)
Ateniese [ABC+11b] [ABC+07]	$\lambda = 1024$	$2\lambda + 160 + 256 = 2464$	2^{23} `exp.` over $\mathbb{Z}_N^*$	$(100 + \ell)$ `exp.` over $\mathbb{Z}_N^*$
POS3	$\lambda = 1024$	$2\lambda + 160 + 256 = 2464$	2^{23} `mul.` over $\mathbb{Z}_{\phi(N)}$	102 `exp.` over $\mathbb{Z}_N^*$
POS2	$\lambda = 1024$	$3\lambda + 240 = 3312$	2^{23} `mul.` over $\mathbb{Z}_p$	100 `exp.` over $\mathbb{Z}_q^*$

is represented by two 80 bits PRF seeds. System parameter ℓ represents the size of set C. All computation times are represented by the corresponding dominant factor. `exp` *and* `mul` *denote the group exponentiation and group multiplication respectively in the corresponding group. Note that one 1024 bits modular exponentiation takes roughly 5 millisecond in a standard PC. See Table 6.1.*

It is worthy to point out that, the most efficient variant scheme E-PDP in Ateniese *et al.* [ABC+07] is suffering from the attack by Shacham and Waters [SW08a]. In Ateniese *et al.* [ABC+07], the main scheme requires the prover to compute the product $\prod_{(i,a_i)\in \mathtt{Chal}} \mathtt{T}_i^{a_i}$ for all tags $\mathtt{T}_i$ selected by the challenge `Chal`. The authors proposed an efficient variant scheme, named E-PDP, by setting all coefficients a_i in the challenge `Chal` as 1, so that in the computation of the aggregated tag, only multiplication is involved and expensive exponentiation is avoided. Shacham and Waters [SW08a] presented an attack on E-PDP, such that the adversary can answer correctly a non-negligible fraction of queries, but there exists no extractor that can recover any data block.

We remark that even compared to the insecure variant E-PDP, our scheme POS3 is still much more efficient in setup and equally efficient in verification.

6.5 Security Analysis of $\mathcal{PDP}$ Scheme POS3

6.5.1 Security Model of $\mathcal{PDP}$

The security game for a $\mathcal{PDP}$ [ABC+07] scheme is the same as the game for a $\mathcal{POR}$ scheme in Section 3.2.2.2 (on page 25), except the following differences:

- To process a verification query, the challenger $\mathcal{C}$ chooses the challenge query in a different way from the $\mathcal{POR}$ game: *$\mathcal{C}$ runs the algorithm* $\mathsf{Challenge}$ *to generate a pair of public-secret tokens* $(\mathsf{pt}, \mathsf{st})$ *and the a challenge query* Chall*, and sends* $(\mathsf{pt}, \mathsf{Chall})$ *to the adversary $\mathcal{A}$ and keeps* st *private. The secret token* st *will be used in the verifier algorithm* Verify*.*

- The **Retrieve** phase is different from the $\mathcal{POR}$ game: *The challenger $\mathcal{C}$ initiates polynomially many $\mathcal{PDP}$ verifications w.r.t. the data file $\mathtt{F}^*$ chosen by $\mathcal{A}$. Suppose the challenger $\mathcal{C}$ asks $\mathcal{A}$ to check all data blocks $\mathtt{F}_i$ of $\mathtt{F}^*$ with index $i \in \mathbf{I}$. The challenger $\mathcal{C}$ extracts file blocks $\{(i, \mathtt{F}'_i) : i \in \mathbf{I}\}$ from $\mathcal{A}$'s storage by applying a PPT knowledge extractor. The adversary $\mathcal{A}$ wins this $\mathcal{PDP}$ security game, if the challenger $\mathcal{C}$ accepts $\mathcal{A}$'s response in the verification during the* **Retrieve** *phase. The challenger $\mathcal{C}$ wins this game if the extracted blocks $\{(i, \mathtt{F}'_i) : i \in \mathbf{I}\}$ are identical to the original $\{(i, \mathtt{F}_i) : i \in \mathbf{I}\}$.*

Definition 9 ([ABC$^+$07]) *A $\mathcal{PDP}$ scheme is sound if for any PPT adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins the above $\mathcal{PDP}$ security game is negligibly close to the probability that $\mathcal{C}$ wins the same security game. That is*

$$\mathsf{Pr}[\mathcal{A} \textit{ wins } \mathcal{PDP} \textit{ game }] \leq \mathsf{Pr}[\mathcal{C} \textit{ wins } \mathcal{PDP} \textit{ game }] + negl. \tag{6.10}$$

6.5.2 Assumptions

The Knowledge of Exponent Assumption (KEA) is introduced by Damgård [Dam92] and subsequently appears in many works [HT98, BP04a, BP04b, Kra05, Den06]. The below is a variant version of KEA in the RSA ring given by Ateniese *et al.* [ABC$^+$07].

Assumption 2 (Knowledge of Exponent Assumption [Dam92, BP04a]) *Let $N = pq$ be a RSA modulus, $g \in \mathbb{Z}_N^*$ and s be an positive integer. For any PPT algorithm $\mathcal{A}$ that takes (N, g, g^s) as input and r as random coin and returns (C, Y) such that $Y = C^s \mod N$, there exists a PPT* extractor *algorithm $\bar{\mathcal{A}}$ which, given (N, g, g^s, r) as input, outputs x such that $g^x = C \mod N$.*

Remark 7

- *Note that the extractor $\bar{\mathcal{A}}$ has access to $\mathcal{A}$'s input (N, g, g^s) and $\mathcal{A}$'s random coin r, thus $\bar{\mathcal{A}}$ can replay step by step the process how $\mathcal{A}$ computes (C, Y) from (N, g, g^s).*
- *This assumption has been shown to hold in generic group by Abe and Fehr [AF07].*

Assumption 3 (Factorization Assumption [RSA78]) *We say an integer N is a RSA modulus, if $N = pq$ and all of $p, q, \frac{p-1}{2}, \frac{q-1}{2}$ are prime numbers and bit-lengths of p and q are equal. Then for any PPT adversary $\mathcal{A}$, the probability that $\mathcal{A}$ can factorize a randomly chosen λ bits RSA modulus, is negligible in λ.*

6.5.3 Security Proof

Theorem 6.1 (POS3 is Sound) *If Knowledge of Exponent Assumption 2 and Factorization Assumption 3 hold and the pseudorandom function family $\{\mathsf{PRF}\}$ is secure, then the proposed scheme* POS3 *is sound.*

The proof below is similar to the proof of Ateniese *et al.* [ABC$^+$07] in the high level: If an adversary $\mathcal{A}$ wins the security game, then some knowledge extractor (as in the assumption KEA) can find a linear combination of data blocks (See Equation (6.5)). Next, each individual block can be obtained by solving a linear equation system. However, the details of the below proof is significantly different from Ateniese *et al.* [ABC$^+$07, ABC$^+$11b].

Like in previous chapters, at first in Lemma 6.2, we prove that the response in a verification of POS3 is unforgeable in a simplified setting, where all decisions (acceptance or rejection) are kept secret from the adversary in the $\mathcal{PDP}$ security game. Recall that we call this simplified setting as "no-feedback" setting.

Lemma 6.2 *Suppose Factorization Assumption 3 holds and the pseudorandom function family $\{\mathsf{PRF}\}$ is secure. Then the proof (ψ_1, ψ_2) in the proposed scheme* POS3 *is unforgeable in the no-feedback setting, where all acceptance or rejection decisions are* *kept secure from the adversary in the $\mathcal{PDP}$ security game. More precisely, let $(\hat{\psi}_1, \hat{\psi}_2)$ denote the adversary $\mathcal{A}$'s response in the* **Retrieve** *phase of the $\mathcal{PDP}$ security game w.r.t.* $\mathsf{POS3}$*, and (ψ_1, ψ_2) denote the corresponding genuine response. The probability*

$$\mathsf{Pr}[\textit{Verifier accepts } (\hat{\psi}_1, \hat{\psi}_2) \quad \wedge \quad (\hat{\psi}_1, \hat{\psi}_2) \neq (\psi_1, \psi_2)] \leq negl(\lambda) \tag{6.11}$$

is negligible.

Proof of Lemma 6.2:

Game 1. The first game is the same as the $\mathcal{PDP}$ security game, except that

- All acceptance or rejection decisions are kept secure from the adversary $\mathcal{A}$. Essentially, the challenger in the $\mathcal{PDP}$ security game does not answer verification queries made by the adversary.
- Adversary $\mathcal{A}$ wins in **Game 1**, if $\mathcal{A}$'s forgery proof $(\hat{\psi}_1, \hat{\psi}_2)$ is accepted and it is different from the genuine proof. Formally, let id, $\{(i, \nu_i) : i \in C\}$ and $(\mathsf{pt}, \mathsf{st})$ denote the file identifier, challenge query, and public-secret tokens respectively in a verification in the **Retrieve** phase of the $\mathcal{PDP}$ security game, let (ψ_1, ψ_2) denote the corresponding genuine proof and (pk, sk) be the public-private key pair. Adversary $\mathcal{A}$ wins in **Game 1**, if

$$\mathsf{Verify}(sk, \mathsf{id}, \mathsf{st}, \{(i, \nu_i) : i \in C\}, \hat{\psi}_1, \hat{\psi}_2) = \texttt{accept} \text{ and } (\hat{\psi}_1, \hat{\psi}_2) \neq (\psi_1, \psi_2). \tag{6.12}$$

Game 2 The second game is the same as **Game 1**, except that the pseudorandom function PRF is replaced by a simulator $\mathsf{PRF}^{\mathsf{Sim}}$, which outputs true randomness over the range of PRF. Precisely, the function $\mathsf{PRF}^{\mathsf{Sim}}$ is evaluated in the following way:

- The challenger keeps a table, which is empty at the very beginning, to store all previous encountered input-output pairs $(v, \mathsf{PRF}^{\mathsf{Sim}}(v))$.
- Given an input v, the challenger lookups the table for v, if there exists an entry (v, u), then return u as output. Otherwise, choose u uniformly randomly from

the range of PRF, insert $(v, \mathsf{PRF}^{\mathsf{Sim}}(v) := u)$ into the table and return u as output.

Game 3 The third game is the same as **Game 2**, except that:

- The range of the function PRF is changed from $\mathbb{Z}_{\phi(N)}$ to $\mathbb{Z}_N$, thus in this game, $\mathsf{PRF}^{\mathsf{Sim}}$ outputs true randomness over $\mathbb{Z}_N$;
- The range of the authentication tag is also changed from $\mathbb{Z}_{\phi(N)}$ to $\mathbb{Z}_N$. More precisely, the Equation (6.4) (on page 99) is replaced by the following equations

$$\sigma_i := \tau \mathsf{F}_i + \mathsf{PRF}_{\mathsf{seed}}(\mathsf{id} \| i) \mod N. \tag{6.13}$$

We remark that in **Game 3**, the challenger is not able to verify adversary's response with algorithm Verify. The challenger does not need to do verification either, since in the no-feedback setting, the challenger will not answer verification queries made by the adversary.

Claim 6.5.1 *If there is a non-negligible difference in a PPT adversary $\mathcal{A}$'s success probability between* **Game 1** *and* **Game 2**, *then there exists another PPT adversary $\mathcal{B}$ that can break the security of the pseudorandom function* PRF. *More precisely,*

$$|\mathsf{Pr}[\mathcal{A} \textit{ wins } \textbf{Game 1}] - \mathsf{Pr}[\mathcal{A} \textit{ wins } \textbf{Game 2}]| \leq N_{\mathsf{PRF}} \cdot \mathsf{Adv}_{\mathcal{B}}^{\mathsf{PRF}},$$

where N_{PRF} is the number of distinct evaluations of pseudorandom function PRF *required and $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{PRF}}$ denotes the probability that $\mathcal{B}$ can distinguish the output of* PRF *from true randomness.*

The above Claim 6.5.1 can be proved using a standard hybrid argument [Gol06]. Here we save the details.

Claim 6.5.2 *For any computationally unbounded adversary $\mathcal{A}$, the probability that $\mathcal{A}$ can find the secret value τ after interacting in* **Game 2**, *is $\frac{1}{\phi(N)}$.*

Proof (Proof Sketch of Claim 6.5.2): In **Game 2**, the pseudorandom function PRF is replaced by a simulator $\mathsf{PRF}^{\mathsf{Sim}}$ which outputs true random numbers in $\mathbb{Z}_{\phi(N)}$, thus the secret value τ is hidden perfectly. Therefore, the probability that an (computationally unbounded) adversary $\mathcal{A}$ can find τ is $\frac{1}{\phi(N)}$. Recall that τ is chosen at random from group $\mathbb{Z}_{\phi(N)}$. □

Claim 6.5.3 *For any PPT adversary $\mathcal{A}$, the probability that $\mathcal{A}$ can factorize N after interacting in* **Game 3** *is negligible.*

Proof (Proof Sketch of Claim 6.5.3): Recall that in **Game 3**, the authentication tag σ_i for each block is a group element chosen at random from $\mathbb{Z}_N$. Suppose a PPT adversary $\mathcal{A}$ can factorize the RSA modulus N after interacting in **Game 3**.

Based on $\mathcal{A}$, we construct a PPT adversary $\mathcal{B}$ to factorize N. Given only the RSA modulus N, the adversary $\mathcal{B}$ can play the role of challenger to setup[2] the $\mathcal{PDP}$ security game w.r.t. scheme $\mathsf{POS3}$, and answer store queries made by the adversary $\mathcal{A}$ by sampling uniform random number from $\mathbb{Z}_N$ as the authentication tag σ_i. Thus,

$$\mathsf{Pr}[\mathcal{A} \text{ factorizes } N \text{ in } \mathbf{Game\ 3}] \leq \mathsf{Pr}[\mathcal{B} \text{ factorizes } N] = \mathsf{Adv}_{\mathcal{B}}^{\text{FACT}}. \tag{6.14}$$

□

Claim 6.5.4 *For any PPT adversary $\mathcal{A}$, the probability that $\mathcal{A}$ can factorize N after interacting in* **Game 2** *is negligible.*

Proof of Claim 6.5.4: We will show that any PPT adversary cannot distinguish between **Game 2** and **Game 3**. As a result, Claim 6.5.3 can imply Claim 6.5.4.

Now we study the *statistical difference* [SV03] between uniform random variables over $\mathbb{Z}_{\phi(N)}$ and over $\mathbb{Z}_N$.

[2]From the input N, $\mathcal{B}$ can generate the public key and simulate the algorithm $\mathsf{DEncode}$. In the no-feedback setting, $\mathcal{B}$ does not need to do verification, so secret key is not necessary.

Let X be a uniform random variable over $\mathbb{Z}_{\phi(N)}$ and Y be a uniform random variable over $\mathbb{Z}_N$. The statistical difference [SV03] between X and Y is

$$\begin{aligned}
\mathsf{SD}(X,Y) &\overset{\text{def}}{=} \frac{1}{2}\sum_a |\mathsf{Pr}[X=a]-\mathsf{Pr}[Y=a]| \\
&= \frac{1}{2}\sum_{a\in\mathbb{Z}_{\phi(N)}} |\mathsf{Pr}[X=a]-\mathsf{Pr}[Y=a]| \;+\; \frac{1}{2}\sum_{a\in\mathbb{Z}_N\setminus\mathbb{Z}_{\phi(N)}} |\mathsf{Pr}[X=a]-\mathsf{Pr}[Y=a]| \\
&= \frac{1}{2}\left(\frac{1}{\phi(N)}-\frac{1}{N}\right)\times\phi(N) \;+\; \frac{1}{2}\left(\frac{1}{N}-0\right)\times(N-\phi(N)) \\
&= 1-\frac{\phi(N)}{N} \\
&= 1-(1-\frac{1}{p})(1-\frac{1}{q}) \\
&= \frac{1}{p}+\frac{1}{q}-\frac{1}{pq}.
\end{aligned}$$

Let N_0 be a positive integer. Let $X_i, i=1,2,\dots,N_0$, be independently and identically distributed uniform random variables over $\mathbb{Z}_{\phi(N)}$, and $Y_i, i=1,2,\dots,N_0$, be independently and identically distributed uniform random variables over $\mathbb{Z}_N$. According to Fact 2.1 and Fact 2.3 of Sahai and Vadhan [SV03], we have

$$\mathsf{SD}((X_1,\dots,X_{N_0}),\ (Y_1,\dots,Y_{N_0})) \le \sum_{i\in[1,N_0]} \mathsf{SD}(X_i,\ Y_i). \tag{6.15}$$

The right hand side of the above Equation (6.15) is

$$\sum_{i\in[1,N_0]} \mathsf{SD}(X_i,\ Y_i) = N_0\times\mathsf{SD}(X,\ Y) = N_0\left(\frac{1}{p}+\frac{1}{q}-\frac{1}{pq}\right). \tag{6.16}$$

Suppose the adversary $\mathcal{A}$ obtains exactly N_{PRF} authentication tags σ_i (σ_i' respectively) for N_{PRF} different indices i's in **Game 2** (**Game 3** respectively). Since σ_i's are independently and identically distributed uniform random variables over $\mathbb{Z}_{\phi(N)}$ and σ_i's are independently and identically distributed uniform random variables over $\mathbb{Z}_N$, the difference of the adversary's views[3] in **Game 2** and **Game 3** is bounded as

[3]Adversary's view is a transcript of all messages received.

below

$$\mathsf{SD}(\text{VIEW}_{\mathcal{A}}^{\textbf{Game 2}},\ \text{VIEW}_{\mathcal{A}}^{\textbf{Game 3}}) \leq N_{\mathsf{PRF}}\left(\frac{1}{p}+\frac{1}{q}-\frac{1}{pq}\right). \tag{6.17}$$

The adversary $\mathcal{A}$ is polynomially bounded, which implies N_{PRF} is polynomially bounded. Therefore, the statistical difference $\mathsf{SD}(\text{VIEW}_{\mathcal{A}}^{\textbf{Game 2}},\ \text{VIEW}_{\mathcal{A}}^{\textbf{Game 3}})$ is negligible in $\lambda \approx \log N \approx 2\log p \approx 2\log q$, and there is no adversary can distinguish between **Game 2** and **Game 3**.

Combining with Claim 6.5.3, we conclude that the probability

$$\mathsf{Pr}[\mathcal{A} \text{ factorizes } N \text{ in } \textbf{Game 2}] \leq \mathsf{Pr}[\mathcal{A} \text{ factorizes } N \text{ in } \textbf{Game 3}] + N_{\mathsf{PRF}}\left(\frac{1}{p}+\frac{1}{q}-\frac{1}{pq}\right)$$

is negligible in $\lambda \approx \log N$. The proof for Claim 6.5.4 is complete. □

Claim 6.5.5 *Without loss of generality, assume $p' < q'$. Then*

$$\mathsf{Pr}[\mathcal{A} \textit{ wins } \textbf{Game 2}] \leq \frac{1}{p'}.$$

Proof of Claim 6.5.5: Let $(\hat{\psi}_1, \hat{\psi}_2)$ denote the adversary $\mathcal{A}$'s forgery output in the **Game 2** and (ψ_1, ψ_2) be the corresponding genuine output which shares the same values $\{(i, \nu_i) : i \in C\}$ with the forgery output. Suppose the adversary $\mathcal{A}$ wins **Game 2**, then $\mathcal{A}$'s forged proof $(\hat{\psi}_1, \hat{\psi}_2)$ is accepted and is different from the genuine output (ψ_1, ψ_2): $(\hat{\psi}_1, \hat{\psi}_2) \neq (\psi_1, \psi_2)$. Since both the forged proof and genuine proof are accepted by the verifier w.r.t. $\{(i, \nu_i) : i \in C\}$ and satisfy the Equation (6.9) (on page 100), we have

$$\left(\hat{\psi}_1\right)^{\tau} = \left(\frac{\hat{\psi}_2}{g^{\sum_{i\in C}\nu_i \mathsf{PRF}_{\mathsf{seed}}(\mathsf{id}\|i)}}\right)^{d} \mod N \tag{6.18}$$

$$(\psi_1)^{\tau} = \left(\frac{\psi_2}{g^{\sum_{i\in C}\nu_i \mathsf{PRF}_{\mathsf{seed}}(\mathsf{id}\|i)}}\right)^{d} \mod N \tag{6.19}$$

Dividing Equation (6.18) with Equation (6.19), we have

$$\left(\frac{\hat{\psi}_1}{\psi_1}\right)^\tau = \left(\frac{\hat{\psi}_2}{\psi_2}\right)^d \mod N \tag{6.20}$$

Recall that the verifier algorithm Verify accepts only if $\psi_1, \hat{\psi}_1 \in \mathcal{QR}_N$. Thus $\frac{\hat{\psi}_1}{\psi_1} \in \mathcal{QR}_N$ is also a quadratic residue. For any element $x \in \mathcal{QR}_N$, $x^{\frac{1}{4}\phi(N)} = 1$, and the multiplicative order of x modulo N will be a factor of $\frac{1}{4}\phi(N) = p'q'$. Let φ denote the multiplicative order of $\frac{\hat{\psi}_1}{\psi_1}$ modulo N. Since $\frac{\hat{\psi}_1}{\psi_1} \neq 1$, φ is at least $\min\{p', q'\} = p'$. Thus a computationally unbounded adversary $\mathcal{B}$ can invoke the adversary $\mathcal{A}$ to obtain the above Equation (6.20) and find the value $(\tau \mod \varphi)$ from Equation (6.20) by solving a discrete log problem with $\frac{\hat{\psi}_1}{\psi_1}$ as base. The probability that $(\tau \mod \varphi) = \tau$ is

$$\mathsf{Pr}[(\tau \mod \varphi) = \tau] = \mathsf{Pr}[\tau \in \mathbb{Z}_\varphi] = \frac{\varphi}{\phi(N)}. \tag{6.21}$$

By Claim 6.5.2, we have the probability

$$\begin{aligned}\mathsf{Pr}[\mathcal{A} \text{ wins } \textbf{Game 2}] &\leq \frac{\mathsf{Pr}[\mathcal{B} \text{ finds } \tau \text{ in } \textbf{Game 2}]}{\mathsf{Pr}[(\tau \mod \varphi) = \tau]} \\ &= \frac{\frac{1}{\phi(N)}}{\frac{\varphi}{\phi(N)}} \\ &= \frac{1}{\varphi} \leq \frac{1}{p'}\end{aligned}$$

is negligible in $\lambda \approx \log N \approx 2 + 2\log p'$. The proof of Claim 6.5.5 is complete. □

Thus, Lemma 6.2 is proved. □

Lemma 6.3 *Suppose Factorization Assumption 3 holds and the pseudorandom function family* $\{\mathsf{PRF}\}$ *is secure. Then the proof* (ψ_1, ψ_2) *in the proposed scheme* $\mathsf{POS3}$ *is unforgeable in the feedback setting, where all acceptance or rejection decisions are provided to the adversary in the* $\mathcal{PDP}$ *security game.*

With Lemma 6.2 present, the proof of the above Lemma 6.3 is essentially identical to the proof of Lemma 4.5 (on page 42). Here we save the details.

Now it is time to prove the main Theorem 6.1 of this chapter.

Proof of Theorem 6.1: Lemma 6.3 states that the response/proof (ψ_1, ψ_2) in the scheme $\mathsf{POS3}$ is unforgeable. Since for random value $d \in \mathbb{Z}^*_{\phi(N)}$, $\mathcal{A}$ can win $\mathcal{PDP}$ security game with non-negligible probability. Then for many different values d_i's, $\mathcal{A}$ can compute $(\psi_{i,1} = g_{d_i}^{\pi_1}, \psi_{i,2} = g^{\pi_2})$ correctly. Let us just consider d_1 and d_2 among these d_i's. Let $c = \frac{d_2}{d_1} \mod \frac{1}{4}\phi(N)$. Given input $(g^{d_1}, g^{d_2} = \left(g^{d_1}\right)^c)$, the adversary $\mathcal{A}$ can output $(g^{d_1\pi_1}, g^{d_2\pi_1} = \left(g^{d_1\pi_1}\right)^c)$. By Knowledge of Exponent Assumption (KEA [Dam92]), there exists a PPT extractor that can find M, such that $g^{d_1\pi_1} = g^{d_1 M} \mod N$.

Case 1: $M \neq \pi_1$ If the two integers M and π_1 are distinct (*Notice that, here we treat M, π_1 as large integer instead of group elements from $\mathbb{Z}_{\phi(N)}$*), then the difference $M - \pi_1$ has to be a multiple of $\frac{1}{4}\phi(N)$, from which the factorization of N can be found using Miller's result [Mil75]. As a result, this case occurs with negligible probability under large integer factorization assumption.

Case 2: $M = \pi_1$ In this case, the extractor finds π_1, as desired. Recall that the large integer $\pi_1 = \sum_{i \in C} \nu_i \mathtt{F}_i$ (Indeed, integer, not group element) is linear equation of file blocks $\mathtt{F}_i$'s. Similar to the proof in Ateniese's PDP [ABC+07, ABC+11b], by choosing independent weights ν_i's in $|C|$ number of executions of the protocol, we obtain $|C|$ independent linear equations in the unknowns $\mathtt{F}_i, i \in C$. Thus these file blocks $\mathtt{F}_i, i \in C$, can be found by solving the linear equation system over integer domain.

Thus, Theorem 6.1 is proved.

□

6.6 Summary

This chapter proposed a Provable Data Possession scheme $\mathsf{POS3}$ using an underlying homomorphic authentication method, which has a similar homomorphic property as the predicate-homomorphic MAC scheme $\mathsf{S2}$. $\mathsf{POS3}$ is very efficient in communication and storage. Its computation complexity is comparable to POS2. Compared to Ateniese *et al.* [ABC+07, ABC+11b], POS3 is much more efficient in computation, and equally efficient in communication and storage. Different from both POS1 and POS2, the verification of POS3 requires a pair of public-secret tokens, which are generated on the fly by the verifier per each verification. Full security proof of POS3 is provided under Knowledge of Exponent Assumption, Large Integer Factorization Assumption and the assumption of existence of secure pseudorandom function.

Part II

Verifiable Outsourced Database:

Authenticating Aggregate Range Query over Multidimensional Outsourced Dataset

Chapter 7

Introduction

Alice has a set **D** of d-dimensional integer points. She chooses a private key, and preprocesses the dataset **D** using her private key to generate some authentication tag **T**. She sends (outsources) **D** and **T** to an untrusted service provider Bob. Then Alice deletes the original copy of dataset **D** and tag **T** from her local storage. Later Alice may issue a query over **D** to Bob, for example, an aggregate query conditional on a multidimensional range selection, and Bob should produce the query result and a proof based on **D** and **T**. Alice wants to authenticate the query result, using only her private key. In Part II of this dissertation, we focus on count query conditional on a multidimensional range selection, that is, counting the number of points within a multidimensional rectangular range. Our solution can be extended to support other types of aggregate queries, like finding minimum coordinate value of points within a multidimensional range, and non-aggregate range selection query, which asks for all points within the query range.

The problem we study fits in the framework of the outsourced database applications [DGMS01, HILM02], which emerged in early 2000s as an example of "Software-as-a-Service" (SaaS). By outsourcing database management, backup services and other IT needs to a professional service provider, companies can reduce expensive cost in purchase of equipments and even more expensive cost in hiring or training qualified IT specialists to maintain the IT services [MNT06].

We are concerned about the communication cost and the storage overhead on Alice/Bob's side. Such requirements exclude the following straightforward approaches: (1) Bob sends back the whole dataset $\mathbf{D}$ with its tag $\mathbf{T}$; (2) Alice keeps a local copy of the dataset; (3) During preprocessing, Alice generates and signs answers to all possible queries. Recently, Gennaro, Gentry and Parno [GGP10] and Chung, Kalai and Vadhan [CKV10] showed that, in general any outsourced/delegated polynomial time function can be verified efficiently using a private verification key in the outsourced computation model, by using fully homomorphic encryption (e.g. Gentry [Gen09]). Nevertheless, it is still meaningful to devise more efficient scheme for small class of functions, without using a fully homomorphic encryption, as mentioned by Gennaro *et al.* [GGP10].

There are many works on authenticating non-aggregate range selection query (e.g. [MND$^+$04, PJRT05, MNT06, CT09, LHKR06, ACK08, MSP09, PZM09, GTT08]). Although there are efficient solutions for 1D and 2D range selection (e.g. Atallah *et al.* [ACK08] for 2D grid dataset), solutions for higher dimension typically rely on geometric partitions which suffer from the "curse of dimensionality", leading to exponential communication overhead. Aggregate range query is arguably more challenging. Only a few works (e.g. [PT08, TYH$^+$09, LHKR10]) are devoted to the authentication of aggregate query, and these works require high communication overhead for high dimensional dataset. Table 7.1 gives a comparison between our result and several previous works.

Table 7.1: Worst case performance of different authentication schemes for aggregate range query or range selection query. This table consists of two parts: the first three rows are for aggregate query; the rest four rows are for range selection query. *Note: (1) The symbol "-" indicates that the authors do not provide such information in their paper. (2) We do not include Pang* et al. *[PT08] and Cheng* et al. *[CT09] in this table, since no concise asymptotic bound are provided. Nevertheless, their performances are limited by their data structure, i.e. KD-tree [PT08] and R-Tree [CT09], which require exponential (in dimension) communication overhead in the worst case. (3) Our scheme supports private key verification, while the other works in this table support public key verification. (4) Our scheme can be extended to provide similar privacy protection like PDAS [TYH+09].*

Scheme	Dimension d	Communication overhead (bits)	Storage overhead	Computation (Verifier Alice)	Computation (Prover Bob)	Query	Techniques
PDAS [TYH+09]	$d = 1$	$O(\|S\| \log N)$	$O(N)$	$O(\|S\| \log N)$	$O(\|S\| + K^2)$	SUM,COUNT	Aggregated commitment + Shamir's Secret-Sharing Scheme
Li *et al.* [LHKR10]	$d \geq 1$	$O(dN + 2^d)$	$\Omega(dN)$	$O(dN + 2^d)$	$\Omega(N^{1-\frac{1}{d}})$	SUM or COUNT or MIN or MAX (*One authentication data structure per query type*)	MHT-like authentication structure for B-Tree/R-Tree
This work	$d \geq 1$	$O(d^2 \log^2 \mathcal{Z})$	$O(dN)$	$O(d^2 \log^2 \mathcal{Z})$†	$O(dN \log \mathcal{Z})$‡	COUNT,MIN,MAX, MEDIAN	(customer designed) functional encryption + **GKEA** based homomorphic tag
Atallah *et al.* [ACK08]	$d = 1, 2$	$O(1)$	$O(N)$	$O(\|S\|)$	$O(1)$	Range Selection	Precomputed prefix sum + **BLS** signature
Martel *et al.* [MND+04]	$d \geq 1$	$O(\log^{d-1} N + \|S\|)$	-	-	-	Range Selection	Authentication Data Structure + Geometry Partition
Chen *et al.* [CMH+08]	$d \geq 1$	$O(\log^d \mathcal{Z})$	$O(N \log^d \mathcal{Z})$	$O(\log^d \mathcal{Z})$	$O(\log^d \mathcal{Z})$	Range Selection	Authentication Tree Structure + Access Control
This work	$d \geq 1$	$O(d^2 \log^2 \mathcal{Z})$	$O(dN)$	$O(d^2 \log^2 \mathcal{Z} + \|S\|)$†	$O(dN \log \mathcal{Z} + \|S\|)$‡	Range Selection	(customer designed) functional encryption + **GKEA** based homomorphic tag

N: The number of tuples in the dataset.
K: The number of servers in PDAS [TYH+09].
†: $O(d^2 \log^2 \mathcal{Z})$ group multiplications.
MHT: Merkle Hash Tree

S: The set of tuples satisfying the query condition.
$\mathcal{Z}$: The domain size of attributes/points in one dimension.
‡: $O(dN \log \mathcal{Z})$ bilinear map operations.

7.1 Our Results

The design of our scheme consists of a few techniques. The first technique exploits Generalized Knowledge of Exponent Assumption (**GKEA**) proposed by Wu and Stinson [WS07], to verify that the result is computed only from data points within the query range. This is achieved by first associating a secret number with each location in the space. Next, a homomorphic tag is computed by Alice from the secret number for each data point and kept in Bob's storage. To authenticate a query, Alice generates and sends to Bob *another* homomorphic tag for each location within the query range, based on the associated secret number and a random nonce. Bob aggregates these two types of tags for all data points within the query range, and sends the resulting aggregated values together with the query result to Alice. The aggregated values can be verified due to homomorphism, and it is difficult to forge the aggregated values using data points outside the query range, under Computational Diffie Hellman assumption and **GKEA**.

However, there are two main drawbacks if the above mentioned technique is employed by itself. Firstly, the validity of the aggregated tags do not rule out "over-counting" (where a data point is used more than once by Bob) and "under-counting" (where a data point is omitted by Bob). To prevent over-counting and under-counting, we further query for data points outside the original query range, and check consistency between proofs and results of these queries.

The second drawback is the high communication overhead required—Alice has to send a tag for *each* location within the query range. In order to lower the communication complexity, we design a functional encryption scheme by exploiting a special property of BBG HIBE scheme [BBG05]. Using this functional encryption scheme, Alice encrypts secret numbers associated with each data point, and sends the resulting ciphertexts to Bob during the setup. For each query, from the query range and the random nonce, Alice can generate a *short* decryption key. From the decryption key, Bob can decrypt those ciphertexts to obtain the tags for data points within the query range as the decrypted values, and learn nothing for data points outside the range, due to the property of our functional encryption scheme.

7.1.1 Contributions

Our main contributions in Part II of this dissertation can be summarized as below:

1. We propose a functional encryption scheme in Chapter 10, by exploiting a special property (we call it "polymorphic property") of the BBG HIBE scheme [BBG05]. Under this functional encryption scheme, given a message Msg and an identity[1] (or attribute) $\vec{\boldsymbol{x}}$, which is a d-dimensional point in domain $[1, \mathcal{Z}]^d$ (Here $\mathcal{Z}$ is an integer), a ciphertext can be generated using the *private*[2] key. A decryption key w.r.t. a d-dimensional rectangular range $\mathbf{R}$ and a random nonce ρ can also be derived from the private key. With this decryption key and the ciphertext for message Msg under identity $\vec{\boldsymbol{x}}$, the decryption algorithm will output $\Omega^{\rho \cdot \mathsf{Msg}}$ iff $\vec{\boldsymbol{x}} \in \mathbf{R}$, where Ω is a part of key of the functional encryption scheme. The size[3] of a private key is in $O(1)$, the size of a ciphertext is in $O(d)$, and the size of a decryption key is in $O(d \log^2 \mathcal{Z})$.
2. We prove that the proposed functional encryption scheme is weak-IND-sID-CPA secure (as defined in Section 10.3), if BBG HIBE scheme [BBG05] is IND-sID-CPA secure (See Theorem 10.2).
3. We propose a scheme for aggregate count query in Chapter 11 by incorporating the functional encryption scheme into the preliminary scheme presented in Chapter 8. The resulting scheme is efficient. For a dataset $\mathbf{D}$ with N points in $[1, \mathcal{Z}]^d$ and a d-dimensional rectangular query range, round complexity is one per query, communication overhead is $O(d^2 \log^2 \mathcal{Z})$ bits per query, and the storage overheads on Alice/Bob's side are $O(1)$ and $O(dN)$, respectively. If the dataset $\mathbf{D}$ is *normalized*[4] [PS85], then $\mathcal{Z} = N$ and $O(d^2 \log^2 \mathcal{Z})$ is sublinear in

[1]In our functional encryption scheme, identity $\vec{\boldsymbol{x}}$ alone is insufficient to encrypt a message.

[2]Unlike [BSW11,O'N10], our functional encryption scheme is a symmetric key encryption system.

[3] Since the private key contains $O(d)$ random elements from $\mathbb{Z}_p^*$ and $O(\ell)$ random elements from $\widetilde{\mathbb{G}}$, its size can be reduced from $O(\ell + d)$ to $O(1)$ (precisely, $O(1)$ number of secret seeds, and each seed with length equal to the security parameter λ), using a pseudorandom function.

[4]For any dataset with size N, one can normalized [PS85] it by sorting the dataset along each dimension, so that the normalized dataset is a subset of $[1, N]^d$. We remark that such normalization will not loss generality: queries over the original dataset can be translated into queries over normalized dataset online by Bob and Alice can verify this translation by checking some authentication tags.

N and polynomial in d. To the best of our knowledge, this is the first solution with worst case communication overhead sublinear in the number of points in the dataset and with polynomial (in (d, N)) storage overhead on server side, without using fully homomorphic encryption scheme [Gen09, vDGHV10]. We compare our result with several previous works in Table 7.1.

4. We prove that the proposed scheme is *correct* and *sound* (Theorem 11.1) under reasonable assumptions (Computational Diffie Hellman assumption, **GKEA** and ℓ-wBDHI assumption [BBG05]). We describe our proof strategy in Section 11.2 and illustrate it by proving that the preliminary scheme in Chapter 8 is *correct* and *sound*. The full proof is in appendix.
5. Our solution for count query leads to efficient solutions that authenticates multidimensional aggregate min/max/median query or non-aggregate range selection query, with communication overhead $O(d^2 \log^2 \mathcal{Z})$. These extensions are described in Chapter 12 and performance is listed in Table 7.1.

7.2 Related work

Researches in secure outsourced database focus on two major aspects: (1) privacy (i.e. protect the data confidentiality against both the service provider and any third party) for example [HILM02, HIM04, MT06, GZ07], and (2) integrity (i.e. authenticate the soundness and completeness of query results returned by the service provider) for example [DGMS01, MND+04, DGMS03, PJRT05, MNT06, PT08, Sio05, CT09, LHKR06, XWYM07, ACK08, YPPK09, MSP09, PZM09, GTT08, TYH+09, LHKR10]. In the latter aspect, a lot of works are conducted for "identity query" [Sio05], i.e. the query result is a subset of the database. Aggregate range query is arguably more challenging and only a few works (for example [PT08, TYH+09, LHKR10]) are devoted to the authentication of aggregate query.

There are roughly four categories of approaches for outsourced database authentication in the literature [DGMS01, MND+04, DGMS03, PJRT05, MNT06, PT08, Sio05, CT09, LHKR06, XWYM07, ACK08, YPPK09, MSP09, PZM09, GTT08]. (1) Cryptographic primitives, like collision-resistant hash, (homomorphic and/or aggregatable) digital signature/commitment [MNT06, HHSY06, TYH$^+$09]. (2) Geometry partition and authenticated data structure [MND$^+$04, CT09, ACK08, MSP09, LHKR06, LHKR10]. For example, Merkle Hash Tree ("MHT" for short; typically for 1D case) and variants, KD-tree with chained signature [PT08], R-Tree with chained signature [CT09], and MHT-like authenticated B-Tree/R-Tree [LHKR10]. (3) Authenticated precomputed partial result, for example, authenticated prefix sum [ACK08, LHKR10] (the static case solution in [LHKR10]) and authenticated partial sum hierarchy [PT08]. (4) Inserting and auditing fake tuples [XWYM07].

To the best of our knowledge, the existing few works (e.g. [PT08,TYH$^+$09,LHKR10]) on authentication of aggregate query either only deal with 1D case, or have communication overhead linear (or even superlinear) w.r.t. the number of data points in the query range, and are suffering from the "curse of dimensionality". Even for multidimensional (non-aggregate) range selection query, the communication overhead is still in $O(\log^{d-1} N + |S|)$ (Martel *et al.* [MND$^+$04], Chen *et al.* [CMH$^+$08]), where S is the set of data points within the query range, N is the number of data points in the dataset, and d is the dimension.

Recently, Gennaro *et al.* [GGP10] and Chung *et al.* [CKV10] proposed methods to authenticate *any* outsourced (or delegated) polynomial time computable function, based on fully homomorphic encryption [Gen09, vDGHV10, Gen10]. They [GGP10, CKV10] also gave a good discussion on why previous techniques (e.g. interactive proofs, probabilistic checkable proof (PCP), and interactive arguments) are insufficient for authenticating outsourced function from the performance point of view. If a function has input size Γ_1 and output size Γ_2, then both Gennaro *et al.* [GGP10] and Chung *et al.* [CKV10] have communication overhead in $\Omega(\Gamma_1 + \Gamma_2)$ to authenticate correctness of the output of this function, where the hidden constant behind the big-Ω notation could be huge. The difference between their solutions and our work may become more clear when authenticating non-aggregate range selection query: Both Gennaro *et al.* [GGP10] and Chung *et al.* [CKV10] will require at least linear communication overhead $\Omega(|S|)$ where S is the set of points within the query range, while our solution (the extension in Section 12.3) still requires $O(d^2 \log^2 \mathcal{Z})$ communication overhead which is independent on size of S.

Shi *et al.* [SBC$^+$07] proposed a predicate encryption scheme called MRQED (Multi-dimensional Range Query over Encrypted Data). Under their scheme, given a message and an identity, which is a d-dimensional point, a ciphertext can be generated. A short decryption key for a d-dimensional rectangular range can be generated from the master secret key. From this decryption key and the ciphertext, the original message can be decrypted, iff the identity point associated with the ciphertext is within the range. There is a subtle but crucial difference between MRQED scheme and our implementation of functional encryption scheme [BSW11, O'N10, LOS$^+$10, OT10]: After a successful decryption, MRQED scheme reveals the original message, whereas our functional encryption scheme reveals only a two-input one-way function value with the original message and a nonce as inputs. As the nonce plays a crucial role in preventing replay attack, it is not suitable to adopt MRQED for our problem. On the other hand, MRQED has its own advantages over our functional encryption scheme, including that MRQED is a public key encryption scheme and has a stronger security model.

Several works in verification of integrity of data stored in remote cloud storage server (e.g. [ABC$^+$07, CX08, SW08a, BJO09a, DVW09, AKK09] and the three schemes POS1, POS2 and POS3 proposed in Part I of this dissertation) also adopted some homomorphic and/or aggregatable verification tags to achieve efficient communication cost.

7.3 Organization

The rest of Part II of this dissertation is organized as follows: Chpater 8 gives a more detailed overview of our main scheme. Chapter 9 presents the problem formulation and security definition. The new functional encryption scheme is constructed in Chapter 10. Our main authentication scheme for count query is described and analyzed in Chapter 11 and its extensions for min/max/median and range selection queries are given in Chapter 12. The full proof of security properties of the functional encryption scheme and the authentication scheme is in Appendix A.

Chapter 8

Overview of Main Scheme

In this chapter, we illustrate our main ideas in two parts: (1) The first part presents a preliminary scheme that authenticates count query. This preliminary scheme is secure (Theorem 11.2) under Computational Diffie Hellman assumption and **GKEA**. However, it requires high communication and computation cost. (2) The second part describes the technique that reduces the communication and computation cost. In particular, the reduction is achieved using a functional encryption scheme which is constructed by exploiting a special property of BBG HIBE scheme [BBG05].

8.1 Preliminary Scheme

Let $\mathbf{D} \subset [1, \mathcal{Z}]^d$ be a set of d-dimensional points. Let G be a cyclic multiplicative group of prime order p and $\{\mathsf{F}_s : [1, \mathcal{Z}]^d \to G\}_{s \in \{0,1\}^\lambda}$ be a pseudorandom function. During setup, Alice chooses at random $\beta \in \mathbb{Z}_p^*$, $\theta \in G$, and $s \in \{0,1\}^\lambda$ as the private key. From the private key, Alice generates a tag value $t_x = (t_{x,1}, t_{x,2}) = (\theta \mathsf{F}_s(x), \mathsf{F}_s(x)^\beta)$ for each data point $x \in \mathbf{D}$. Alice also computes a value $\Delta = \prod_{x \in \mathbf{D}} t_{x,2}$. Next, Alice sends dataset $\mathbf{D}$ and tag values $\mathbf{T} = \{t_x : x \in \mathbf{D}\}$ to Bob and deletes everything except Δ and the private key (β, θ, s) from her storage.

Consider a count query conditional on a range $\mathbf{R} \subset [1, \mathcal{Z}]^d$, which asks for the size of set $\mathbf{D} \cap \mathbf{R}$. Bob is expected to send to Alice a number X as the query result, and a proof to show that indeed $X = |\mathbf{D} \cap \mathbf{R}| \pmod{p}$.

To authenticate this query, Alice chooses two random nonces[1] ρ and $\hat{\rho}$, computes and sends auxiliary messages (called as *challenge-message*) $\Phi = \{\mathsf{F}_s(x)^\rho : x \in \mathbf{R}\}$ and $\hat{\Phi} = \{\mathsf{F}_s(x)^{\hat{\rho}} : x \in \mathbf{R}^\complement\}$ to Bob, where $\mathbf{R}^\complement = \{x \in [1, \mathcal{Z}]^d : x \notin \mathbf{R}\}$ is the complement set of range $\mathbf{R}$. Bob is expected to compute $X = |\mathbf{D} \cap \mathbf{R}|$ and $\hat{X} = |\mathbf{D} \cap \mathbf{R}^\complement|$, and generate the proof $(\Psi_1, \Psi_2, \Psi_3, \hat{\Psi}_1, \hat{\Psi}_2, \hat{\Psi}_3)$ as below:

Step B1: Bob multiplies all tags t_x for point $x \in \mathbf{D} \cap \mathbf{R}$ to obtain Ψ_1, Ψ_2

$$\Psi_1 \leftarrow \prod_{x \in \mathbf{D} \cap \mathbf{R}} t_{x,1} = \theta^X \prod_{x \in \mathbf{D} \cap \mathbf{R}} \mathsf{F}_s(x); \qquad \Psi_2 \leftarrow \prod_{x \in \mathbf{D} \cap \mathbf{R}} t_{x,2} = \prod_{x \in \mathbf{D} \cap \mathbf{R}} \mathsf{F}_s(x)^\beta.$$

Step B2: Bob multiplies all values $\mathsf{F}_s(x)^\rho$ from the challenge-message Φ for point $x \in \mathbf{D} \cap \mathbf{R}$ to obtain Ψ_3:

$$\Psi_3 \leftarrow \prod_{x \in \mathbf{D} \cap \mathbf{R}} \mathsf{F}_s(x)^\rho.$$

Step B3: Bob repeats Step B1 and Step B2 for data points $x \in \mathbf{D} \cap \mathbf{R}^\complement$ using challenge-message $\hat{\Phi}$ to obtain $\hat{\Psi}_1, \hat{\Psi}_2, \hat{\Psi}_3$ correspondingly.

Bob sends back $(X, \Psi_1, \Psi_2, \Psi_3; \hat{X}, \hat{\Psi}_1, \hat{\Psi}_2, \hat{\Psi}_3)$ to Alice, and Alice verifies the returned message using the private key (β, θ, s), the secret random nonces $(\rho, \hat{\rho})$, and value Δ in this way:

Step A1: Is (Ψ_1, Ψ_2) indeed an aggregated multiplication of valid tags?

$$\left(\frac{\Psi_1}{\theta^X}\right)^\beta \stackrel{?}{=} \Psi_2.$$

Step A2: Is Ψ_2 computed using *only* points inside $\mathbf{D} \cap \mathbf{R}$?

$$\Psi_2^\rho \stackrel{?}{=} \Psi_3^\beta.$$

[1]Here the secret random nonces prevent Bob from abusing this challenge-message for other queries.

Step A3: Repeat Step A1 and Step A2 to verify $(\hat{X}, \hat{\Psi}_1, \hat{\Psi}_2, \hat{\Psi}_3)$ using private key and secret random nonce $\hat{\rho}$.

Step A4: Is every point counted for *exactly* once?

$$\Psi_2 \cdot \hat{\Psi}_2 \overset{?}{=} \Delta. \tag{8.1}$$

If all of above verifications succeed, Alice accepts that X is the correct query result.

Remark 8

- *In the computations of Ψ_1, Ψ_2 and Ψ_3, an adversary (playing the role of Bob) may try to multiply tags for some points within $\mathbf{D} \cap \mathbf{R}$ multiple times, and/or ignore some points within $\mathbf{D} \cap \mathbf{R}$. That is, the adversary tries to find integers μ_x's for each point $x \in \mathbf{D}$, treat $(t_{x,1}^{\mu_x},\ t_{x,2}^{\mu_x})$ as the tag of point $x \in \mathbf{D}$ in the computations of Ψ_1 and Ψ_2, treat $\{\mathsf{F}_s(x)^{\rho \cdot \mu_x} : x \in \mathbf{D} \cap \mathbf{R}\}$ as the challenge-message in the computation of Ψ_3, and compute the query result $X = \sum_{x \in \mathbf{D} \cap \mathbf{R}} \mu_x$. Here $\mu_x > 1$ indicates that the point x is* over-counted*, $\mu_x < 1$ indicates that the point x is* under-counted*, and μ_x might take negative integer value. By doing so, the adversary can pass the verifications in Step A1, A2 and A3. However, if such adversary succeeds in passing Step A4, i.e. he can find integers μ_x's, for each $x \in \mathbf{D}$, such that $\prod_{x \in \mathbf{D}} t_{x,2}^{\mu_x} = \Delta = \prod_{x \in \mathbf{D}} t_{x,2}$ and some $\mu_x \neq 1$, then he can solve **DLP** (Discrete Log Problem).*

- *The above attack strategy is "restrictive". A stronger adversary might performance something else to pass the verifications. Fortunately, under **GKEA**, we can show that: (informally) for any efficient adversary, under-counting and/or over-counting are the only ways to pass verifications in Step A1, A2 and A3, with a negligible exception.*

- *In the preliminary scheme, the size of challenge-message is linear w.r.t. $|\mathbf{R}|$, which can be very large, leading to large computation and communication cost.*

- *The second component $t_{x,2} = \mathsf{F}_s(x)^\beta$ in a tag t_x is required to deal with adaptive adversary: An adversary does not gain additional knowledge from adaptive learning, since it can generate challenge-message by itself from $\{\mathsf{F}_s(x)^\beta : x \in \mathbf{D}\}$, and the forged challenge-message is identically distributed as the challenge-message generated by Alice.*

8.2 Deliver challenge-message efficiently and securely

To reduce complexity, Alice needs a way to deliver the information $\Phi = \{\mathsf{F}_s(x)^\rho : x \in \mathbf{R}\}$ (actually the subset $\{\mathsf{F}_s(x)^\rho : x \in \mathbf{D} \cap \mathbf{R}\}$ is sufficient to serve the purpose) to Bob by sending some auxiliary data of much smaller size, and Bob must not know the value of $\mathsf{F}_s(x)^\rho$ for point $x \notin \mathbf{R}$. We design such delivery method by exploiting a special property of existing HIBE scheme.

Polymorphic Property. We observe that some (HIBE) encryption scheme (KeyGen, Enc, Dec), e.g. BBG HIBE scheme [BBG05], satisfies a *polymorphic property*: From a pair of keys $(pk, sk) \in \mathsf{KeyGen}(1^\lambda)$, a plaintext M, an identity id, and a random coin r, one can efficiently find multiple tuples $(pk_j, sk_j, M_j, r_j), 1 \le j \le n$, such that for each $1 \le j \le n$, $(pk_j, sk_j) \in \mathsf{KeyGen}(1^\lambda)$ is a valid key pair and

$$\mathsf{Enc}_{pk}(\mathsf{id}, M; r) = \mathsf{CT} = \mathsf{Enc}_{pk_j}(\mathsf{id}, M_j; r_j).$$

From the opposite point of view, a ciphertext CT can be decrypted into different values M_j's using different decryption keys. We can view these decrypted values M_j's as a function of the original plaintext M which is used to produce the ciphertext CT. Hence, such polymorphic property may lead to a new way to construct functional encryption schemes [BSW11, O'N10, LOS+10, OT10].

Overview of the Delivery Method. Alice can deliver the challenge-message $\{\mathsf{F}_s(x)^\rho : x \in \mathbf{D} \cap \mathbf{R}\}$ in this way: For simplicity, assume the dataset $\mathbf{D} \subset [1, \mathcal{Z}]$ consists of N 1D data points. Each point x in the domain is associated with an identity $\mathsf{ID}(x)$, which corresponds to a leaf node in the identity hierarchy tree of BBG HIBE scheme. In the setup phase, for each data point $x_i \in \mathbf{D}$, Alice computes a ciphertext $\vec{\boldsymbol{c}}_i$, which can be considered as encryption of $M_{i,j}$ under key $(pk_j, sk_j), j = 1, 2, 3, \ldots$ Alice sends these N ciphertexts $\{\vec{\boldsymbol{c}}_i : 1 \leq i \leq N\}$ to Bob at the end of setup phase. Later, for a query range $\mathbf{R}$, Alice chooses a random nonce ρ and derives the delegation key $\vec{\boldsymbol{\delta}}$ w.r.t. the set $S = \{\mathsf{ID}(x) : x \in \mathbf{R}\}$ of identities from the key pair (pk_ρ, sk_ρ), and sends $\vec{\boldsymbol{\delta}}$ as challenge-message to Bob. With this delegation key, Bob is able to decrypt ciphertext $\vec{\boldsymbol{c}}_i$ to obtain $M_{i,\rho}$ if $x_i \in \mathbf{D} \cap \mathbf{R}$. By carefully choosing parameters, we may have $M_{i,\rho} = \mathsf{F}_s(x_i)^\rho$ as desired.

The tree structure of the identity hierarchy of HIBE scheme facilitates short description of the delegation key, and thus the challenge-message, which originally contains $\mathcal{Z}$ subkeys in the worst case, can now be expressed with only $O(\log \mathcal{Z})$ subkeys.

For high dimensional cases, we perform the above procedure for each dimension, and combines the challenge-messages for all dimensions. The security of this method can be reduced to the $\mathsf{IND\text{-}sID\text{-}CPA}$ security of the underlying HIBE scheme.

Chapter 9

Formulation

In this chapter, we formalize the problem and security model, and describe the security assumptions formally.

9.1 Dataset and Query

The dataset $\mathbf{D}$ is a set of N d-dimensional points $\vec{\boldsymbol{x}}_1, \vec{\boldsymbol{x}}_2, \ldots, \vec{\boldsymbol{x}}_N$ from the domain $[1, \mathcal{Z}]^d$ where $\mathcal{Z}$ can be a large integer (e.g. 64 bits integer). Let $\mathbf{R} = [a_1, b_1] \times [a_2, b_2] \times \ldots \times [a_d, b_d] \subseteq [1, \mathcal{Z}]^d$ be a d-dimensional rectangular range. In Part II of this dissertation, we focus on queries that count the number of points in $\mathbf{D} \cap \mathbf{R}$. Let us define function F :

$$F(\mathbf{D}, \mathbf{R}) \overset{\text{def}}{=} |\mathbf{D} \cap \mathbf{R}| = |\mathbf{D} \cap \mathbf{R}| \mod p,$$

where p is a large prime. Note that p is exponential in the security parameter λ and N is polynomial in λ.

We write an integer interval $[1, n] = \{1, 2, 3, \ldots, n\}$ as $[n]$ for abbreviation, where n is some positive integer.

9.2 Security Model

We formulize our problem, as a variant of *Verifiable Computation* proposed by Gennaro *et al.* [GGP10]. Let us view a query on a dataset as the function $F : \mathbb{D} \times \mathbb{R} \to \{0,1\}^*$, where $\mathbb{D}$ is the domain of datasets, $\mathbb{R}$ is the set of all possible queries, and the output of F is represented by a binary string. We define a *remote computing* protocol as follows:

Definition 10 ($\mathcal{RC}$) *A* Remote Computing *($\mathcal{RC}$) protocol for a function $F : \mathbb{D} \times \mathbb{R} \to \{0,1\}^*$, between Alice and Bob, consists of a setup phase and a query phase. The setup phase consists of a key generating algorithm* KGen *and data encoding algorithm* DEnc*; the query phase consists of a pair of interactive algorithms, namely the evaluator* Eval *and the extractor* Ext*. These four algorithms* $(\mathsf{KGen}, \mathsf{DEnc}, \langle \mathsf{Eval}, \mathsf{Ext} \rangle)$ *run in the following way:*

Setup Phase

1. *Given security parameter λ, Alice generates a key K: $K \leftarrow \mathsf{KGen}(1^\lambda)$.*
2. *Alice encodes dataset $\mathbf{D} \in \mathbb{D}$: $(\mathbf{D}_{\mathtt{B}}, \mathbf{D}_{\mathtt{A}}) \leftarrow \mathsf{DEnc}(\mathbf{D}, K)$, then sends $\mathbf{D}_{\mathtt{B}}$ to Bob and keeps $\mathbf{D}_{\mathtt{A}}$.*

Query Phase *The query phase consists of multiple query sessions. In each query session, Alice and Bob interact as below.*

1. *Alice selects a query $\mathbf{R} \in \mathbb{R}$.*
2. *Algorithm $\mathsf{Ext}(\mathbf{D}_{\mathtt{A}}, \mathbf{R}, K)$ on Alice's side, interacts with algorithm $\mathsf{Eval}(\mathbf{D}_{\mathtt{B}})$ on Bob's side to compute $(\zeta, X, \vec{\mathbf{\Psi}}) \leftarrow \langle \mathsf{Eval}(\mathbf{D}_{\mathtt{B}}), \mathsf{Ext}(\mathbf{D}_{\mathtt{A}}, \mathbf{R}, K) \rangle$, where $\zeta \in \{\mathtt{accept}, \mathtt{reject}\}$ and $\vec{\mathbf{\Psi}}$ is the proof of result X.*
   If $\zeta = \mathtt{accept}$, then Alice accepts that X is equal to $F(\mathbf{D}, \mathbf{R})$. Otherwise, Alice rejects.

We are interested in efficient $\mathcal{RC}$ protocol where the sizes of $K, \mathbf{D}_{\mathtt{A}}, \mathbf{D}_{\mathtt{B}}$, and the communication overhead are all small.

One of the main differences between $\mathcal{RC}$ model and Verifiable Computation [GGP10] model is that: $\mathcal{RC}$ allows multiple rounds of communication between Alice and Bob to compute a function value in a query session; in contrast, Verifiable Computation model [GGP10] allows only one round of communication. Although in Part II of this dissertation, both the scheme for count query in Section 11.1 and the extension for range selection query in Section 12.3 requires only one round of communication due to parallelism, our extensions for min/max/median query in Section 12.1 and Section 12.2 require at least two rounds of communication, since dependencies between different rounds prevent parallelism.

We say a $\mathcal{RC}$ protocol is *verifiable*, if the following conditions hold: (1) Alice always accepts, when Bob follows the protocol honestly; (2) Alice rejects with o.h.p. (overwhelming high probability), when Bob returns a wrong result. Here we consider adversaries, i.e. malicious Bob, who are allowed to interact with Alice and learn for polynomial number of query sessions, before launching the attack. During the learning, the adversary may store whatever it has seen or learnt in a state variable.

Definition 11 ($\mathcal{VRC}$) *Let λ be the security parameter. We call a $\mathcal{RC}$ protocol $\mathcal{E} = (\mathsf{KGen}, \mathsf{DEnc}, \langle \mathsf{Eval}, \mathsf{Ext} \rangle)$ w.r.t. function $F : \mathbb{D} \times \mathbb{R} \to \{0,1\}^*$ a* Verifiable Remote Computing *($\mathcal{VRC}$) protocol, if the following two conditions hold:*

- *correctness: for any $\mathbf{D} \in \mathbb{D}$, any $K \leftarrow \mathsf{KGen}(1^\lambda)$ and any $\mathbf{R} \in \mathbb{R}$, it holds that $\langle \mathsf{Eval}(\mathbf{D}_{\mathtt{B}}), \mathsf{Ext}(\mathbf{D}_{\mathtt{A}}, \mathbf{R}, K) \rangle = (\mathtt{accept},\ F(\mathbf{D}, \mathbf{R}),\ \vec{\mathbf{\Psi}})$ for some $\vec{\mathbf{\Psi}}$, where $(\mathbf{D}_{\mathtt{B}}, \mathbf{D}_{\mathtt{A}}) \leftarrow \mathsf{DEnc}(\mathbf{D}, K)$.*
- *soundness: for any adaptive PPT adversary $\mathcal{A}$, the advantage $\mathsf{Adv}_{\mathcal{E},\mathcal{A}}(1^\lambda) \leq negl(\lambda)$,*

where $\mathsf{Adv}_{\mathcal{E},\mathcal{A}}(1^\lambda)$ is defined as below

Experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathcal{E}}(1^\lambda)$

```
$\mathbf{D} \leftarrow \mathcal{A}(\mathsf{view}_{\mathcal{A}}^{\mathcal{E}})$;
$K \leftarrow \mathsf{KGen}(1^\lambda)$;
$(\mathbf{D}_{\mathtt{B}}, \mathbf{D}_{\mathtt{A}}) \leftarrow \mathsf{DEnc}(\mathbf{D}, K)$;
loop until $\mathcal{A}(\mathsf{view}_{\mathcal{A}}^{\mathcal{E}})$ decides to stop
    $\mathbf{R}_i \leftarrow \mathcal{A}(\mathbf{D}_{\mathtt{B}}, \mathsf{view}_{\mathcal{A}}^{\mathcal{E}})$;
    $(\zeta_i, X_i, \vec{\mathbf{\Psi}}_i) \leftarrow \langle \mathcal{A}(\mathbf{D}_{\mathtt{B}}, \mathsf{view}_{\mathcal{A}}^{\mathcal{E}}), \mathsf{Ext}(\mathbf{D}_{\mathtt{A}}, \mathbf{R}_i, K) \rangle$;
$\mathbf{R} \leftarrow \mathcal{A}(\mathbf{D}_{\mathtt{B}}, \mathsf{view}_{\mathcal{A}}^{\mathcal{E}})$;
$(\zeta, X, \vec{\mathbf{\Psi}}) \leftarrow \langle \mathcal{A}(\mathbf{D}_{\mathtt{B}}, \mathsf{view}_{\mathcal{A}}^{\mathcal{E}}), \mathsf{Ext}(\mathbf{D}_{\mathtt{A}}, \mathbf{R}, K) \rangle$;
Output $(\zeta, X, \vec{\mathbf{\Psi}}, \mathsf{view}_{\mathcal{A}}^{\mathcal{E}}, \mathbf{D}, \mathbf{R})$.
```

$$\mathsf{Adv}_{\mathcal{E},\mathcal{A}}(1^\lambda) \overset{\text{def}}{=} \mathsf{Pr}\left[\begin{array}{c} (\zeta, X, \vec{\mathbf{\Psi}}, \mathsf{view}_{\mathcal{A}}^{\mathcal{E}}, \mathbf{D}, \mathbf{R}) \leftarrow \mathsf{Exp}_{\mathcal{A}}^{\mathcal{E}}(1^\lambda) : \\ \zeta = \mathtt{accept} \ \wedge \ X \neq F(\mathbf{D}, \mathbf{R}) \end{array} \right].$$

The probability is taken over all random coins used by related algorithms, negl$(\cdot)$ is some negligible function, and $\mathsf{view}_{\mathcal{A}}^{\mathcal{E}}$ is a state variable[1] *describing all random coins chosen by $\mathcal{A}$ and all messages $\mathcal{A}$ can access during previous interactions with $\mathcal{E}$.*

9.3 Assumptions

Throughout Part II of this dissertation, let p be a λ bits safe prime, and $e : \mathbb{G} \times \mathbb{G} \rightarrow \widetilde{\mathbb{G}}$ be a bilinear map, where $\mathbb{G}$ and $\widetilde{\mathbb{G}}$ are two cyclic multiplicative groups of order p.

Assumption 4 (Computational Diffie Hellman Assumption [DH76]) *For any PPT algorithm $\mathcal{A}$, it holds that*

$$\mathsf{Pr}\left[\mathcal{A}(g, g^a, g^b) = g^{ab}\right] \leq \nu_1(\lambda),$$

where g is chosen at random from $\widetilde{\mathbb{G}}$, a and b are chosen at random from $\mathbb{Z}_p^*$, and $\nu_1(\cdot)$ is some negligible function.

[1]The adaptive adversary $\mathcal{A}$ may keep updating this state variable.

The assumption **GKEA** is an extension of **KEA1** [Dam92, HT98, BP04b, Kra05, Den06] and **KEA3** [BP04a], and proposed by Wu and Stinson [WS07]. Roughly, **GKEA** assumption can be described as below:

> *For any adversary $\mathcal{A}$ that takes input $\{(u_i, u_i^{\beta}) : 1 \le i \le m\}$ and returns (U_1, U_2) with $U_1^{\beta} = U_2$, there exists an "extractor" $\bar{\mathcal{A}}$, which given the same inputs as $\mathcal{A}$ returns $\{\mu_i : 1 \le i \le m\}$, such that $\prod_{i=1}^{m} u_i^{\mu_i} = U_1$.*

Assumption 5 (Generalized KEA [Dam92, BP04a, WS07, Gro10]) *Let $\mathcal{A}$ and $\bar{\mathcal{A}}$ be two algorithms. We define the* **GKEA**-*advantage of $\mathcal{A}$ against $\bar{\mathcal{A}}$ as*

$$\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\mathbf{GKEA}}(\lambda) \overset{\text{def}}{=} \mathsf{Pr}\left[\begin{array}{l} W_m = \{(u_i, u_i^{\beta}) : i \in [m], u_i \xleftarrow{\$} \widetilde{\mathbb{G}}, \beta \xleftarrow{\$} \mathbb{Z}_p^*\} \\ (U_1, U_2) \leftarrow \mathcal{A}(W_m; r); \\ (\mu_1, \mu_2, \ldots, \mu_m) \leftarrow \bar{\mathcal{A}}(W_m; r, \bar{r}) : \\ \qquad U_1^{\beta} = U_2 \ \wedge \ U_1 \neq \prod_{j=1}^{m} u_j^{\mu_j} \end{array}\right], \tag{9.1}$$

where the probability is taken over all random coins used and with m fixed (Here the notation $\xleftarrow{\$}$ denotes uniformly randomly sampling from a set. E.g. the expression $x \xleftarrow{\$} S$ means that x is uniformly randomly chosen from the set S). For any PPT algorithm $\mathcal{A}$ (called as adversary), there exists PPT algorithm $\bar{\mathcal{A}}$ (called as extractor), such that the **GKEA**-*advantage of $\mathcal{A}$ against $\bar{\mathcal{A}}$ is upper bounded by some negligible function $\nu_2(\lambda)$, i.e. $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\mathbf{GKEA}}(\lambda) \le \nu_2(\lambda)$, where m is polynomial in λ.*

Remark 9

- **GKEA** *is a natural extension of* ***KEA1*** *and* ***KEA3****, in the sense that* ***GKEA*** $\Rightarrow$ **KEA3** $\Rightarrow$ **KEA1**. *Abe and Fehr [AF07] proved* ***KEA1*** *and* ***KEA3*** *in* generic group model. *Following their techniques,* ***GKEA*** *can be proved in generic group model.*

- *If $u_i = g^{x^i}$ for each i with some random g and x, then Assumption 5 will become the q-****PKE*** *assumption proposed by Groth [Gro10].*

Furthermore, the (Decision) ℓ-wBDHI Assumption [BBG05] is required for the $\mathsf{IND\text{-}sID\text{-}CPA}$ security of the underlying BBG HIBE scheme.

Chapter 10

Functional Encryption Scheme

Recall that the preliminary scheme in Chapter 8 requires large communication and computation cost. In order to reduce such cost, we construct a functional encryption [BSW11, O'N10] scheme by exploiting the polymorphic property of BBG HIBE scheme [BBG05], following the overview given in Chapter 8.

10.1 Polymorphic Property of BBG HIBE Scheme

We observe that the BBG HIBE scheme [BBG05] satisfies the polymorphic property: An encryption of a message M can be viewed as the encryption of another message $\widehat{M}$ under different key. Precisely, let CT and $\widehat{\mathsf{CT}}$ be defined as follows, we have $\mathsf{CT} = \widehat{\mathsf{CT}}$:

$$\begin{aligned}
\mathsf{CT} &= \mathsf{Encrypt}(\mathsf{params}, \mathsf{id}, M; s) = \left(\Omega^s \cdot M,\ g^s,\ \left(h_1^{I_1} \cdots h_k^{I_k} \cdot g_3\right)^s\right) \\
&\quad \text{under key: } \mathtt{params} = (g, g_1, g_2, g_3, h_1, \ldots, h_\ell, \Omega = e(g_1, g_2)),\ \ \mathtt{master\text{-}key} = g_2^\alpha \\
\widehat{\mathsf{CT}} &= \mathsf{Encrypt}(\widehat{\mathsf{params}}, \mathsf{id}, \widehat{M}; sz) = \left(\Omega^{sz} \cdot \widehat{M},\ \widehat{g}^{sz},\ \left(\widehat{h}_1^{I_1} \cdots \widehat{h}_k^{I_k} \cdot \widehat{g}_3\right)^{sz}\right), \\
&\quad \text{under key: } \widehat{\mathtt{params}} = (\widehat{g}, g_1, g_2, \widehat{g}_3, \widehat{h}_1, \ldots, \widehat{h}_\ell, \Omega = e(g_1, g_2)),\ \ \widehat{\mathtt{master\text{-}key}} = g_2^{\alpha z}
\end{aligned} \tag{10.1}$$

where ℓ is the maximum depth of the HIBE scheme, $k \le \ell$ is the length of identity id, $\widehat{M} = M\Omega^{s(1-z)}$, $\widehat{g} = g^{z^{-1} \bmod p}$, $\widehat{g}_3 = g_3^{z^{-1} \bmod p}$, $\widehat{h}_i = h_i^{z^{-1} \bmod p}$ for $1 \le i \le \ell$ and identity $\mathsf{id} = (I_1, \ldots, I_k) \in \left(\mathbb{Z}_p^*\right)^k$. To be self-contained, the description of this

BBG HIBE scheme is given in Appendix A.1 (on page 174). One can verify the above equality easily.

10.2 Define Identities based on Binary Interval Tree

An identity is a sequence of elements from $\mathbb{Z}_p^*$. To apply HIBE scheme, we intend to construct two mappings to associate identities to integers or integer intervals: (1) $\mathsf{ID}(\cdot)$ maps an integer $x \in [\mathcal{Z}]$ into an identity $\mathsf{ID}(x) \in \left(\mathbb{Z}_p^*\right)^\ell$, where $\ell = \lceil \log \mathcal{Z} \rceil$ is the height of identity hierarchy tree of the BBG HIBE scheme. (2) $\mathsf{IdSet}(\cdot)$ maps an integer interval $[a, b] \subseteq [\mathcal{Z}]$ into a set of $O(\ell)$ identities, where each identity is a sequence of at most ℓ elements from $\mathbb{Z}_p^*$. The two mappings ID and IdSet are required to satisfy the property: For any $x \in [a, b] \subseteq [\mathcal{Z}]$, there is a unique identity id in the set $\mathsf{IdSet}([a, b])$, such that identity id is a prefix of identity $\mathsf{ID}(x)$. If $x \notin [a, b]$, then there is no such identity id in $\mathsf{IdSet}([a, b])$. For each dimension $\iota \in [d]$, we will construct such mappings ID_ι and IdSet_ι using a *binary interval tree* [SBC$^+$07]. The resulting mappings are made public.

Binary Interval Tree. The binary interval tree is constructed as below: First, we build a complete ordered binary tree with 2^ℓ leaf nodes. Next, we associate an integer interval to each tree node in a bottom-up manner: (1) Counting from the leftmost leaf, the j-th leaf is associated with interval $[j, j]$; (2) For any internal node, the associated interval is the union of the two intervals associated to its left and right children respectively. As a result, the interval associated to the root node is $[1, 2^\ell]$. An example of binary interval tree with size 8 is showed in Figure 10.1.

Constructions of Mappings ID_ι and IdSet_ι for dimension ι. Let $\mathcal{H} : \mathbb{Z}_{2^\ell+1} \times \mathbb{Z}_{2^\ell+1} \times [d] \to \mathbb{Z}_p^*$ be a collision resistant hash function. Let $(\mathtt{v}_1, \mathtt{v}_2, \ldots, \mathtt{v}_m)$ be the path from the root node $\mathtt{v}_1$ to the node $\mathtt{v}_m$ in the binary interval tree. We associate to node $\mathtt{v}_m$ the identity $(\mathcal{H}(a_1, b_1, \iota), \ldots, \mathcal{H}(a_m, b_m, \iota)) \in \left(\mathbb{Z}_p^*\right)^m$, where $[a_j, b_j]$ is the interval associated to node $\mathtt{v}_j$, $1 \leq j \leq m$.

For any $x \in [\mathcal{Z}]$, we define $\mathsf{ID}_\iota(x)$ as the identity associated to the x-th leaf node (counting from the left). For any interval $[a, b] \subseteq [\mathcal{Z}]$, we find the minimum set $\{\mathsf{v}_j : \mathsf{v}_j \text{ is a tree node}, 1 \leq j \leq n\}$ such that the intervals associated to v_j's form a partition of $[a, b]$, and define $\mathsf{IdSet}_\iota([a, b])$ as the set $\{\mathsf{id}_j : \mathsf{id}_j$ is the identity associated to node v_j, $1 \leq j \leq n\}$. One can verify that the newly constructed mappings ID_ι and IdSet_ι satisfy the property mentioned in the beginning of Section 10.2. Furthermore, the set $\mathsf{IdSet}_\iota([a, b])$ contains $O(\ell)$ identities and each identity is a sequence of at most ℓ elements from $\mathbb{Z}_p^*$.

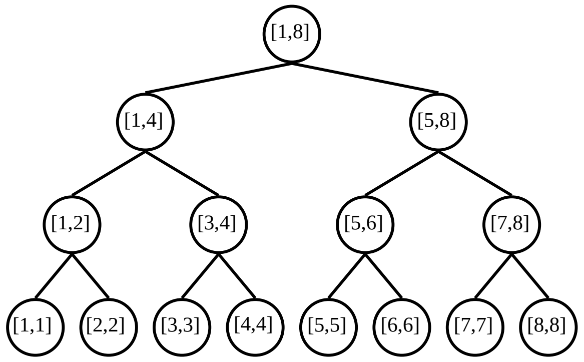


Figure 10.1: Binary Interval Tree with 8 leaf nodes.

10.3 Construction of Functional Encryption Scheme

Let (Setup, KeyGen, Encrypt, Decrypt) be the BBG Hierarchical Identity Based Encryption (HIBE) scheme proposed by Boneh, Boyen and Goh [BBG05] (the description of this scheme is in Appendix A.1 on page 174). Based on this HIBE scheme, we construct a functional encryption scheme $\mathsf{FE} = (f\mathsf{Setup}, f\mathsf{Enc}, f\mathsf{KeyGen}, f\mathsf{Dec}, f\mathsf{Mult})$ as below.

$f\mathsf{Setup}(1^\lambda, d, \mathcal{Z})$: `security parameter` λ`, dimension` d`, maximum integer` $\mathcal{Z}$`; the domain of points is` $[\mathcal{Z}]^d$

1. Let $\ell = \lceil \log \mathcal{Z} \rceil$. Run algorithm $\mathsf{Setup}(\ell, \lambda)$ to obtain bilinear groups $(p, \mathbb{G}, \widetilde{\mathbb{G}}, e)$, public parameter $\mathtt{params} = (g, g_1, g_2, g_3, h_1, \ldots, h_\ell, \Omega = e(g_1, g_2))$ and master private key $\mathtt{master\text{-}key} = g_2^\alpha$, such that p is a λ bits prime, $\mathbb{G}, \widetilde{\mathbb{G}}$ are cyclic multiplicative groups of order p, $e : \mathbb{G} \times \mathbb{G} \to \widetilde{\mathbb{G}}$ is a bilinear map, g is a generator of $\mathbb{G}$, $\alpha \in \mathbb{Z}_p$, $g_1 = g^\alpha \in \mathbb{G}$, and $g_2, g_3, h_1, \ldots, h_\ell \in \mathbb{G}$.

2. Let ID_ι and IdSet_ι, $\iota \in [d]$, be the mappings as in Section 10.2.

3. Choose d random elements $\tau_1, \ldots, \tau_d$ from $\mathbb{Z}_p^*$ and let $\vec{\boldsymbol{\tau}} = (\tau_1, \ldots, \tau_d)$.

4. Let $pk = (p, \mathbb{G}, \widetilde{\mathbb{G}}, e, \Omega)$ and $sk = (pk, \mathtt{params}, \mathtt{master\text{-}key}, \vec{\boldsymbol{\tau}})$. Make ID_ι's and IdSet_ι's public and output (pk, sk).

$f\mathsf{Enc}(\mathsf{Msg}, \vec{\boldsymbol{x}}, sk)$: `message` $\mathsf{Msg} \in \mathbb{Z}_p^*$, d`-dimensional point` $\vec{\boldsymbol{x}}$

1. Treat the d-dimensional point $\vec{\boldsymbol{x}}$ as $(x_1, \ldots, x_d) \in [\mathcal{Z}]^d$; recall that the private key sk is $(pk, \mathsf{params}, \mathsf{master\text{-}key}, \vec{\boldsymbol{\tau}})$, where $\vec{\boldsymbol{\tau}} = (\tau_1, \ldots, \tau_d)$.

2. Choose d random elements $s_1, \ldots, s_d$ from $\mathbb{Z}_p$ with constraint $\mathsf{Msg} = -\sum_{j=1}^d s_j \cdot \tau_j \pmod p$.

3. Choose d random elements $\sigma_1, \ldots, \sigma_d$ from $\widetilde{\mathbb{G}}$ with constraint $\prod_{j=1}^d \sigma_j = \Omega^{-\sum_{j=1}^d s_j}$.

4. For each $j \in [d]$, encrypt σ_j under identity $\mathsf{ID}_j(x_j)$ with random coin s_j to obtain ciphertext $\vec{\boldsymbol{c}}_j$ as follows

$$\vec{\boldsymbol{c}}_j \leftarrow \mathsf{Encrypt}(\mathtt{params},\ \mathsf{ID}_j(x_j),\ \sigma_j;\ s_j). \tag{10.2}$$

5. Output ciphertext $\mathsf{CT} = (\vec{\boldsymbol{c}}_1, \ldots, \vec{\boldsymbol{c}}_d)$.

$f\mathsf{KeyGen}(\mathbf{R}, \rho, sk)$: d`-dimensional rectangular range` $\mathbf{R}$`, function key` $\rho \in \mathbb{Z}_p^*$

1. Treat the d-dimensional rectangular range $\mathbf{R} \subseteq [\mathcal{Z}]^d$ as Cartesian product $\mathbf{A}_1 \times \mathbf{A}_2 \ldots \times \mathbf{A}_d$, where $\mathbf{A}_j \subseteq [\mathcal{Z}]$ for each $j \in [d]$; recall that the private key sk is $(pk, \mathsf{params}, \mathsf{master\text{-}key}, \vec{\boldsymbol{\tau}})$, where $\vec{\boldsymbol{\tau}} = (\tau_1, \ldots, \tau_d)$.

2. For each dimension $j \in [d]$, generate a set δ_j in this way:
   (a) For each identity $\mathsf{id} \in \mathsf{IdSet}_j(\mathbf{A}_j)$, generate the private key d_{id}, using algorithm KeyGen and taking the value $\mathsf{master\text{-}key}^{\rho\tau_j}$ as the master key.
   (b) Set $\delta_j \leftarrow \{d_{\mathsf{id}} : \mathsf{id} \in \mathsf{IdSet}_j(\mathbf{A}_j)\}$.
3. Output delegation key $\vec{\boldsymbol{\delta}} = (\delta_1, \delta_2, \ldots, \delta_d)$.

$f\mathsf{Dec}(\mathsf{CT}, \vec{\boldsymbol{x}}, \mathbf{R}, \vec{\boldsymbol{\delta}}, pk)$: `ciphertext` CT`,` d`-dimensional point` $\vec{\boldsymbol{x}}$`,` d`-dimensional rectangular range` $\mathbf{R}$`, delegation key` $\vec{\boldsymbol{\delta}}$

1. Treat the d-dimensional rectangular range $\mathbf{R} \subseteq [\mathcal{Z}]^d$ as Cartesian product $\mathbf{A}_1 \times \mathbf{A}_2 \ldots \times \mathbf{A}_d$, where $\mathbf{A}_j \subseteq [\mathcal{Z}]$ for each $j \in [d]$. Let us write the ciphertext CT as $(\vec{\boldsymbol{c}}_1, \ldots, \vec{\boldsymbol{c}}_d)$, and the d-dimensional point $\vec{\boldsymbol{x}}$ as $(x_1, \ldots, x_d)$.
2. For each dimension $j \in [d]$, generate $\widetilde{t}_j$ in this way: If $x_j \notin \mathbf{A}_j$, then output $\perp$ and abort. Otherwise, do the followings:
   (a) Find the unique identity $\mathsf{id}^* \in \mathsf{IdSet}_j(\mathbf{A}_j)$ such that id^* is a prefix of identity $\mathsf{ID}_j(x_j)$.
   (b) Parse $\vec{\boldsymbol{\delta}}$ as $(\delta_1, \ldots, \delta_d)$ and find the private key $d_{\mathsf{id}^*} \in \delta_j = \{d_{\mathsf{id}} : \mathsf{id} \in \mathsf{IdSet}_j(\mathbf{A}_j)\}$ for identity id^*.
   (c) Generate the private key d_j for the identity $\mathsf{ID}_j(x_j)$ from private key d_{id^*}, using algorithm KeyGen.
   (d) Decrypt $\vec{\boldsymbol{c}}_j$ using algorithm $\mathsf{Decrypt}$ with decryption key d_j, and denote the decrypted message as $\widetilde{t}_j$.
3. Output $\widetilde{t} = \prod_{1 \le j \le d} \widetilde{t}_j \in \widetilde{\mathbb{G}}$.

$f\mathsf{Mult}(\mathsf{CT}', y, pk)$: `ciphertext` CT'`,` $y \in \widetilde{\mathbb{G}}$

1. Let us write the ciphertext CT' as $(\vec{\boldsymbol{c}}'_1, \ldots, \vec{\boldsymbol{c}}'_d)$.
2. Choose d random elements $\eta_1, \ldots, \eta_d$ from $\widetilde{\mathbb{G}}$ with constraint $\prod_{j=1}^{d} \eta_j = y \in \widetilde{\mathbb{G}}$.

3. For each dimension $j \in [d]$: parse $\vec{\boldsymbol{c}}'_j$ as (A, B, C) and set $\vec{\boldsymbol{c}}_j = (A \cdot \eta_j,\ B,\ C)$.

4. Output ciphertext $\mathsf{CT} = (\vec{\boldsymbol{c}}_1, \ldots, \vec{\boldsymbol{c}}_d)$.
   Note: Both $\vec{\boldsymbol{c}}'_j$ and $\vec{\boldsymbol{c}}_j$ are valid BBG ciphertexts for different plaintexts under the same identity.

10.4 The Constructed Functional Encryption Scheme is Correct and Secure

In this section, we analyze the correctness and security of the newly constructed functional encryption scheme.

10.4.1 Correctness

Let us define a key-ed function family $\{f_\rho : \mathbb{Z}_p^* \to \widetilde{\mathbb{G}}\}_{\rho \in \mathbb{Z}_p^*}$ as below: Let $\Omega \in \widetilde{\mathbb{G}}$ be as in $f\mathsf{Setup}$ of Section 10.3.

$$f_1(\mathsf{Msg}) = \Omega^{\mathsf{Msg}}; \qquad \forall \rho \in \mathbb{Z}_p^*,\ f_\rho(\mathsf{Msg}) = f_1(\mathsf{Msg})^\rho\ \in \widetilde{\mathbb{G}}. \tag{10.3}$$

Lemma 10.1 (FE is correct) *The functional encryption scheme FE described in Section 10.3 satisfies these properties:*

(a) *For any public-private key pair $(pk, sk) \leftarrow f\mathsf{Setup}(1^\lambda, d, \mathcal{Z})$, for any message $\mathsf{Msg} \in \mathbb{Z}_p^*$, for any point $\vec{\boldsymbol{x}} \in [\mathcal{Z}]^d$, for any rectangular range $\mathbf{R} \subseteq [\mathcal{Z}]^d$, for any $\rho \in \mathbb{Z}_p^*$, if $\mathsf{CT} \leftarrow f\mathsf{Enc}(\mathsf{Msg}, \vec{\boldsymbol{x}}, sk)$ and $\vec{\boldsymbol{\delta}} \leftarrow f\mathsf{KeyGen}(\mathbf{R}, \rho, sk)$, then*

$$f\mathsf{Dec}(\mathsf{CT},\ \vec{\boldsymbol{x}},\ \mathbf{R},\ \vec{\boldsymbol{\delta}},\ pk) = \begin{cases} f_\rho(\mathsf{Msg}) & (\textit{if}\ \vec{\boldsymbol{x}} \in \mathbf{R}) \\ \perp & (\textit{otherwise}) \end{cases} \tag{10.4}$$

(b) *For any public-private key pair $(pk, sk) \leftarrow f\mathsf{Setup}(1^\lambda, d, \mathcal{Z})$, for any message $\mathsf{Msg} \in \mathbb{Z}_p^*$, for any point $\vec{\boldsymbol{x}} \in [\mathcal{Z}]^d$, for any rectangular range $\mathbf{R} \subseteq [\mathcal{Z}]^d$, for any $\rho \in \mathbb{Z}_p^*$, for any $y \in \widetilde{\mathbb{G}}$, if $\mathsf{CT} \leftarrow f\mathsf{Enc}(\mathsf{Msg}, \vec{\boldsymbol{x}}, sk)$ and $\vec{\boldsymbol{\delta}} \leftarrow f\mathsf{KeyGen}(\mathbf{R}, \rho, sk)$,* *then*

$$f\mathsf{Dec}(f\mathsf{Mult}(\mathsf{CT}, y, pk),\ \vec{\boldsymbol{x}},\ \mathbf{R},\ \vec{\boldsymbol{\delta}},\ pk) = \begin{cases} y \cdot f_\rho(\mathsf{Msg}) & (\mathit{if}\ \vec{\boldsymbol{x}} \in \mathbf{R}) \\ \perp & (\mathit{otherwise}) \end{cases} \tag{10.5}$$

(The proof is in Appendix A.3.)

10.4.2 Security

We formulize the security requirement of our functional encryption scheme by modifying the IND-sID-CPA security game [BBG05]. The resulting weak-IND-sID-CPA security game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ is defined as below:

Commit: The adversary $\mathcal{A}$ chooses the target point $\vec{\boldsymbol{x}}^*$ from the space $[\mathcal{Z}]^d$ and sends it to the challenger $\mathcal{C}$.
Setup: The challenger $\mathcal{C}$ runs the setup algorithm $f\mathsf{Setup}$ and gives $\mathcal{A}$ the resulting system parameters pk, keeping the secret key sk to itself.
Challenge: The challenger $\mathcal{C}$ chooses two plaintexts $\mathsf{Msg}_0, \mathsf{Msg}_1$ at random from the message space $\mathbb{Z}_p^*$, and chooses a random bit $b \in \{0, 1\}$. $\mathcal{C}$ sets the challenge ciphertext to $\mathsf{CT} = f\mathsf{Enc}(\mathsf{Msg}_b,\ \vec{\boldsymbol{x}}^*,\ sk)$, and sends $(\mathsf{CT}, f_1(\mathsf{Msg}_0), f_1(\mathsf{Msg}_1))$ to the adversary $\mathcal{A}$.
Learning Phase: The adversary $\mathcal{A}$ adaptively issues queries to the challenger $\mathcal{C}$, where each query is one of the following:

- Delegation key query $(\mathbf{R}, \rho)$, where $\vec{\boldsymbol{x}}^* \notin \mathbf{R}$: In response to this query, $\mathcal{C}$ runs algorithm $f\mathsf{KeyGen}(\mathbf{R}, \rho, sk)$ to generate the delegation key $\vec{\boldsymbol{\delta}}$, and sends $\vec{\boldsymbol{\delta}}$ to $\mathcal{A}$.
- Anonymous delegation key query $(\mathbf{R})$: In response to this query, $\mathcal{C}$ chooses ρ at random from the space $\mathbb{Z}_p^*$ and runs algorithm $f\mathsf{KeyGen}(\mathbf{R}, \rho, sk)$ to generate the delegation key $\vec{\boldsymbol{\delta}}$, and sends $\vec{\boldsymbol{\delta}}$ to $\mathcal{A}$.
- Encryption query $(\mathsf{Msg}, \vec{\boldsymbol{x}})$: In response to this query, $\mathcal{C}$ runs $f\mathsf{Enc}(\mathsf{Msg}, \vec{\boldsymbol{x}}, sk)$ to obtain a ciphertext, and sends the ciphertext to $\mathcal{A}$.

Guess: Finally, the adversary $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$ and wins if $b = b'$.

We refer to the above adversary $\mathcal{A}$ as a $\mathsf{weak\text{-}IND\text{-}sID\text{-}CPA}$ adversary. We define the advantage of the adversary $\mathcal{A}$ in attacking the scheme FE as

$$\mathsf{Adv}_{\mathsf{FE},\mathcal{A}}^{\mathsf{weak\text{-}IND\text{-}sID\text{-}CPA}} = \left|\mathsf{Pr}[b = b'] - \frac{1}{2}\right|.$$

Theorem 10.2 *If the BBG HIBE scheme is $\mathsf{IND\text{-}sID\text{-}CPA}$ secure (as defined in [BBG05]), then the functional encryption scheme FE constructed in Section 10.3 is $\mathsf{weak\text{-}IND\text{-}sID\text{-}CPA}$ secure. That is, there is no PPT adversary $\mathcal{A}$ that can win the $\mathsf{weak\text{-}IND\text{-}sID\text{-}CPA}$ game against the scheme FE with non-negligible advantage $\mathsf{Adv}_{\mathsf{FE},\mathcal{A}}^{\mathsf{weak\text{-}IND\text{-}sID\text{-}CPA}}$. (The more detailed statement of this theorem and its proof appear in Appendix A.4 on page 179).*

Most of previous functional encryption schemes (e.g. attribute-based encryption [SW05], and predicate encryption [KSW08]), if not all, allow the decryptor to obtain the original plaintext Msg in "good" case (e.g. if the attribute of plaintext and/or the decryption key satisfy the designated predicate) from a ciphertext of Msg, and nothing otherwise. In contrast, our functional encryption scheme FE only allows the decryptor to obtain $f_1(\mathsf{Msg})^\rho$ in "good" case, from a ciphertext of Msg, where f_1 is a one-way function. Unlike [BSW11, O'N10], our functional encryption scheme is a symmetric key system. Our security formulation is weaker than previous works (e.g. [BSW11, O'N10]), but it is sufficient for our main result in Theorem 11.1.

Chapter 11

Authenticating Aggregate Count Query over Multidimensional Outsourced Dataset

In this chapter, we propose an authentication scheme for aggregate count query in Section 11.1. We analyze the proposed scheme in security aspect in Section 11.2 and in performance aspect in Section 11.3.

11.1 The Main Construction

By incorporating the newly constructed functional encryption scheme FE into the preliminary scheme presented in Chapter 8, we construct a $\mathcal{RC}$ protocol $\mathcal{E} = (\mathsf{KGen}, \mathsf{DEnc}, \langle \mathsf{Eval}, \mathsf{Ext} \rangle)$ as below, to authenticate aggregate count query over multidimensional dataset.

(Alice) $\mathsf{KGen}(1^\lambda)$:

Step 1: Run $f\mathsf{Setup}(1^\lambda)$ to obtain public/private key pair (pk', sk), where $pk' = (p, \mathbb{G}, \widetilde{\mathbb{G}}, e, \Omega)$. Set $pk = (p, \mathbb{G}, \widetilde{\mathbb{G}}, e)$.

Note: e is a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \widetilde{\mathbb{G}}$, $\Omega \in \widetilde{\mathbb{G}}$, and both $\mathbb{G}$ and $\widetilde{\mathbb{G}}$ are multiplicative groups of prime order p.

Step 2: Choose β, γ at random from $\mathbb{Z}_p^*$, and θ at random from $\widetilde{\mathbb{G}}$. Let $\mathcal{K} = (pk', sk, \beta, \gamma, \theta)$.

Step 3: Output $(\mathcal{K}, pk)$.

(Alice) $\mathsf{DEnc}(\mathbf{D}; \mathcal{K})$:

Step 1: Dataset $\mathbf{D} \subset [\mathcal{Z}]^d$ consists of N points $\vec{\boldsymbol{x}}_1, \vec{\boldsymbol{x}}_2, \ldots, \vec{\boldsymbol{x}}_N$. Choose N random elements $W_1, \ldots, W_N$ from $\mathbb{Z}_p^*$ independently and N random elements $v_1, \ldots, v_N$ from $\widetilde{\mathbb{G}}$ independently.

Step 2: For each $i \in [N]$, generate a tag $\vec{\boldsymbol{t}}_i \in \widetilde{\mathbb{G}}^3$:

$$\vec{\boldsymbol{t}}_i \leftarrow \left(\theta v_i,\ v_i^{\beta},\ w_i = f_1(W_i)\right). \tag{11.1}$$

Note: Alice can evaluate functions $\{f_\rho(\cdot)\}_{\rho \in \mathbb{Z}_p^*}$, since Alice has $\Omega \in \widetilde{\mathbb{G}}$.

Step 3: For each $i \in [N]$:

(a) Encrypt message W_i under point $\vec{\boldsymbol{x}}_i$: $\mathsf{CT}'_i \leftarrow f\mathsf{Enc}(W_i, \vec{\boldsymbol{x}}_i, sk)$;

(b) Apply the homomorphic property of FE to attach v_i^{γ} to ciphertext: $\mathsf{CT}_i \leftarrow f\mathsf{Mult}(\mathsf{CT}'_i, v_i^{\gamma}, pk')$.

Step 4: Send $\mathbf{D}_{\mathtt{B}} = \left(\mathbf{D},\ \mathbf{T} = \{\vec{\boldsymbol{t}}_i : i \in [N]\},\ \mathbf{C} = \{\mathsf{CT}_i : i \in [N]\},\ pk\right)$ to Bob, and keep *only* key $\mathcal{K}$ and $\mathbf{D}_{\mathtt{A}} = \left(N, d, \Delta = \prod_{i \in [N]} v_i^{\beta}\right)$ in local storage.

(Alice, Bob) $\mathsf{ProVer} = \langle \mathsf{Eval}(\mathbf{D}_{\mathtt{B}}),\ \mathsf{Ext}(\mathbf{D}_{\mathtt{A}},\ \mathbf{R},\ \mathcal{K}) \rangle$: $\mathbf{D}_{\mathtt{A}} = (N, d, \Delta)$, $\mathbf{D}_{\mathtt{B}} = (\mathbf{D}, \mathbf{T}, \mathbf{C}, pk)$
Precondition: The query range $\mathbf{R} \subset [\mathcal{Z}]^d$ is a rectangular range.

Step 1: Alice partitions the complement range $\mathbf{R}^{\complement}$ into $2d$ rectangular ranges $\{\mathbf{R}_\ell \subset [\mathcal{Z}]^d : \ell \in [1, 2d]\}$, and sets $\mathbf{R}_0 = \mathbf{R}$.

Step 2: For $0 \le \ell \le 2d$, Alice and Bob invokes $\mathsf{CollRes}$ on range $\mathbf{R}_\ell$. Denote the output as $(\zeta_\ell, X_\ell, \Psi_2^{(\ell)})$.

Step 3: Alice sets $\zeta = \texttt{accept}$, if the following equalities hold

$$\forall 0 \le \ell \le 2d, \zeta_\ell \overset{?}{=} \texttt{accept}, \qquad \prod_{0 \le \ell \le 2d} \Psi_2^{(\ell)} \overset{?}{\equiv} \Delta; \tag{11.2}$$

otherwise sets $\zeta = \texttt{reject}$. Alice outputs (ζ, X_0, Δ).

(Alice, Bob) $\mathsf{CollRes} = \left\langle \widetilde{\mathsf{Eval}}(\mathbf{D}_{\mathtt{B}}),\ \widetilde{\mathsf{Ext}}(\mathbf{D}_{\mathtt{A}},\ \mathbf{R},\ \mathcal{K}) \right\rangle$: $\mathbf{D}_{\mathtt{A}} = (N, d, \Delta)$, $\mathbf{D}_{\mathtt{B}} = (\mathbf{D}, \mathbf{T}, \mathbf{C}, pk)$
Precondition. The query range $\mathbf{R} \subset [\mathcal{Z}]^d$ is a rectangular range.

Step A1: (Alice's first step) Alice chooses a random nonce ρ from $\mathbb{Z}_p^*$ and runs algorithm $f\mathsf{KeyGen}$ to generate a delegation key $\vec{\boldsymbol{\delta}}$ w.r.t. $(\mathbf{R}, \rho)$: $\vec{\boldsymbol{\delta}} \leftarrow f\mathsf{KeyGen}(\mathbf{R}, \rho, sk)$. Alice sends $(\mathbf{R}, \vec{\boldsymbol{\delta}})$ to Bob, where $\vec{\boldsymbol{\delta}}$ will be treated as the challenge-message.

Step B1: (Bob's first step) Bob computes the query result X and proof $(\Psi_1, \Psi_2, \Psi_3, \Psi_4)$ as follows

$$X \leftarrow |\mathbf{D} \cap \mathbf{R}|\,; \quad (\Psi_1, \Psi_2, \Psi_3) \leftarrow \bigotimes_{\vec{\boldsymbol{x}}_i \in \mathbf{D} \cap \mathbf{R}} \vec{\boldsymbol{t}}_i; \quad \Psi_4 \leftarrow \prod_{\vec{\boldsymbol{x}}_i \in \mathbf{D} \cap \mathbf{R}} f\mathsf{Dec}(\mathsf{CT}_i, \vec{\boldsymbol{x}}_i, \mathbf{R}, \vec{\boldsymbol{\delta}}, pk). \tag{11.3}$$

Bob sends $(X, \Psi_1, \Psi_2, \Psi_3, \Psi_4)$ to Alice.
Note: For $\vec{\boldsymbol{x}}_i \in \mathbf{D} \cap \mathbf{R}$, $f\mathsf{Dec}(\mathsf{CT}_i, \vec{\boldsymbol{x}}_i, \mathbf{R}, \vec{\boldsymbol{\delta}}, pk)$ is supposed to output $v_i^\gamma f_1(W_i)^\rho = v_i^\gamma w_i^\rho$; the operator $\bigotimes$ denotes component-wise multiplication of vectors of the same dimension.

Step A2: (Alice's second step) Let $\Lambda \leftarrow \frac{\Psi_1}{\theta^X}$. Alice sets $\zeta = \texttt{accept}$, if the following equalities hold

$$\Lambda^\beta \overset{?}{=} \Psi_2, \qquad \Lambda^\gamma \Psi_3^\rho \overset{?}{=} \Psi_4. \tag{11.4}$$

Otherwise sets $\zeta = \texttt{reject}$. Alice outputs (ζ, X, Ψ_2).

Remark 10

1. *To understand the verifications in the interactive algorithm* $\mathsf{CollRes}$*, one may consider a homomorphic tag function* Tag *defined as below: Let* y *be the input,* $K = (\beta, \gamma, \theta)$ *be the key, and* v, w *be random coins.*

$$\mathsf{Tag}_K(y; v, w) = (\theta^y v,\ v^\beta,\ w,\ v^\gamma w^\rho);$$
$$\prod_i \mathsf{Tag}_K(y_i; v_i, w_i) = \mathsf{Tag}_K\left(\sum_i y_i;\ \prod_i v_i,\ \prod_i w_i\right).$$

   Note that the first three component of $\mathsf{Tag}_K(1; v_i, w_i)$ *are just the three components of vector* $\vec{\boldsymbol{t}}_i$ *generated in Equation* (11.1)*, and the fourth component* $v^\gamma w^\rho$ *is the output of* $f\mathsf{Dec}(\mathsf{CT}_i, \vec{\boldsymbol{x}}_i, \mathbf{R}, \vec{\boldsymbol{\delta}}, pk)$ *w.r.t. the random nonce* ρ *(i.e.* $\vec{\boldsymbol{\delta}} \leftarrow f\mathsf{KeyGen}(\mathbf{R}, \rho, sk)$ *). In the algorithm* $\mathsf{CollRes}$*, Bob computes the product of* Tag *values of data points within the query range as the proof of the query result* X*. Next, Alice verifies whether the proof* $(\Psi_1, \Psi_2, \Psi_3, \Psi_4)$ *is a valid* Tag *value for* X*, with key* K *and without knowing the values of random coins* v_j*'s and* w_j*'s.*

2. *A simple alternative construction of* Tag *is as follows, as in the preliminary scheme in Chapter 8:*

$$\mathsf{Tag}'_K(y; v) = (\theta^y v,\ v^\beta,\ v^\rho),$$

   where v^ρ *is the output of* $f\mathsf{Dec}$*. However, we encounter difficulty in proving the security of the constructed scheme if we adopt* Tag'*. Thus we introduce a new random coin* w *and change* Tag' *to* Tag*. It is not clear whether such additional component plays a crucial role in achieving security or simply helps in simplifying the proof. The role of the additional random coin* w_i *is as follows: In our proof, given the first two components* $\{(\theta^{y_i} v_i, v_i^\beta) : i \in [N]\}$ *of all* *tag values, a simulator can simulate Alice in our scheme. Next, the simulator invokes a malicious Bob to interact with Alice to produce a forgery. For the alternative construction with* Tag'*, to simulate* DEnc*, the simulator has to find a ciphertext for some message* $W_i \in \mathbb{Z}_p^*$*, such that* $f_1(W_i) = \Omega^{W_i} = v_i^{\beta}$*, which could be infeasible due to* ***DLP*** *(Discrete Log Problem). In our construction with* Tag*, since the additional term* w_i *is independent on the first two components of tag* $\vec{\boldsymbol{t}}_i$*, the simulator can choose* W_i *freely, and generate* $w_i \leftarrow f_1(W_i)$ *and the ciphertext* $\mathsf{CT}_i \leftarrow f\mathsf{Mult}\left(f\mathsf{Enc}\left(W_i, \vec{\boldsymbol{x}}_i, sk\right),\ v_i^{\beta\cdot\gamma'},\ pk\right)$ *where* $\gamma' \in \mathbb{Z}_p^*$ *is randomly chosen. Consequently, if* $\vec{\boldsymbol{x}}_i \in \mathbf{R}$*, then by Lemma 10.1,* $f\mathsf{Dec}\left(\mathsf{CT}_i, \vec{\boldsymbol{x}}_i, \mathbf{R}, f\mathsf{KeyGen}(\mathbf{R}, \rho, sk), pk\right) = v_i^{\beta\gamma'} f_\rho(W_i) = v_i^{\gamma} w_i^{\rho}$ *as desired (taking* γ *as* $\beta\gamma'$*). Thus the simulation of* DEnc *can be done.*

3. *To ensure completeness and prevent over-counting or under-counting, we need to run* $\mathsf{CollRes}$ *on the complement query range* $\mathbf{R}^{\complement}$*. Since our functional encryption scheme* FE *only supports high dimensional* rectangular *ranges, we have to divide range* $\mathbf{R}^{\complement}$ *into multiple high dimensional rectangular ranges, and then run* $\mathsf{CollRes}$ *on each of them.*

4. *The* $(2d+1)$ *invocations of* $\mathsf{CollRes}$ *can be executed in parallel. As a result, the round complexity of our scheme is exactly 1.*

5. *In Step 2 of* ProVer*, in the extreme case that* $\mathbf{R}_\ell = \emptyset$*, Alice can save the execution of* $\mathsf{CollRes}$ *on range* $\mathbf{R}_\ell$*, since Alice can predict the genuine result* $(\zeta_\ell = \mathtt{accept}, X_\ell = 0, \Psi_1 = \Psi_2 = \Psi_3 = \Psi_4 = 1 \in \widetilde{\mathbb{G}})$*.*

6. *The complement set of rectangular range* $[a_1, b_1] \times [a_2, b_2] \times \ldots \times [a_d, b_d] \subset [\mathcal{Z}]^d$ *can be partitioned into* $2d$ *number of rectangular ranges:* $[a_1, b_1] \times [a_2, b_2] \times \ldots \times [1, a_\ell - 1]$ *and* $[a_1, b_1] \times [a_2, b_2] \times \ldots \times [b_\ell + 1, \mathcal{Z}]$*,* $\ell \in [d]$*.*

11.2 Security Analysis

11.2.1 Our main theorem

Theorem 11.1 (Main Theorem of Part II) *Suppose Computational Diffie-Hellman Assumption 4 and* **GKEA** *Assumption 5 hold, and* BBG *[BBG05] HIBE scheme is* IND-sID-CPA *secure. Then the* $\mathcal{RC}$ *protocol* $\mathcal{E} = (\mathsf{KGen}, \mathsf{DEnc}, \mathsf{ProVer})$ *constructed in Section 11.1 is* $\mathcal{VRC}$ *w.r.t. aggregate count function* $F(\cdot,\cdot)$*, where* $F(\cdot,\cdot)$ *is defined in Section 9.1 (on page 127) and* $\mathcal{VRC}$ *(Verifiable Remote Computing protocol) is defined in Definition 11 (on page 129). Namely,* $\mathcal{E}$ *is* correct *and* sound *w.r.t. multidimensional aggregate count query.*

The full proof is in appendix. In this section, we will brief the proof strategy for the main theorem, and prove the security of the preliminary scheme given in Chapter 8 as an illustration of our proof strategy.

11.2.2 Overview of Proof of Main Theorem

To process a query, our scheme (particularly the algorithm ProVer) invokes $(2d + 1)$ instances of interactive algorithm CollRes. In each instance of CollRes, Bob is supposed to return a 5-tuple $(X, \Psi_1, \Psi_2, \Psi_3, \Psi_4)$ where X is the query result and $(\Psi_1, \Psi_2, \Psi_3, \Psi_4)$ is the proof, and Alice will verify whether the proof is valid w.r.t. the query result. Furthermore, after all of $(2d+1)$ invocations, Alice will perform one additional verification (equation (11.2)) to ensure completeness and prevent over-counting or under-counting. In order to fool Alice with a wrong query result, an adversary has to provide a valid 5-tuple for each invocation of CollRes and pass the equation (11.2). Therefore, an adversary against $\mathcal{E} = (\mathsf{KGen}, \mathsf{DEnc}, \mathsf{ProVer})$ is also an adversary against $\widetilde{\mathcal{E}} = (\mathsf{KGen}, \mathsf{DEnc}, \mathsf{CollRes})$.

We consider various types of PPT adversaries against $\mathcal{E}$ or $\widetilde{\mathcal{E}}$, which interacts with Alice by playing the role of Bob and intends to output a wrong query result and a forged but valid proof:

- Type I adversary: This adversary is not confined in any way in its attack strategy and produces a 5-tuple $(X, \Psi_1, \Psi_2, \Psi_3, \Psi_4)$ on a query range $\mathbf{R}$.

- Type II adversary: A restricted adversary which can produce the same forgery[1] from the same input as Type I adversary, and can find N integers[2] μ_i's, $1 \leq i \leq N$, such that $\Psi_2 = \prod_{i \in [N]} \left(v_i^{\beta}\right)^{\mu_i}$, where β and v_i's are as in Section 11.1.

- Type III adversary: The same as Type II adversary, with additional constraint: $\mu_i = 0$ for each $\vec{\boldsymbol{x}}_i \in \mathbf{D} \cap \mathbf{R}^{\complement}$.

- Type IV adversary: The same as Type III adversary, with additional constraint: $\mu_i = 1$ for each $\vec{\boldsymbol{x}}_i \in \mathbf{D} \cap \mathbf{R}$.

Note that the Type II (or Type III, Type IV) adversary *explicitly* outputs $\{\mu_1, \ldots, \mu_N\}$, and *implicitly* outputs $(X, \Psi_1, \Psi_2, \Psi_3, \Psi_4)$ which is exactly the output of the corresponding Type I adversary. This is similar to the relationship between **KEA** extractor and **KEA** adversary.

Basically, our proof framework is like this:

- Lemma A.2 (on page 188): The existence of Type I adversary implies the existence of Type II adversary, under **GKEA** Assumption 5, where Type I adversary is a counterpart of adversary $\mathcal{A}$ in **GKEA** and Type II adversary is a counterpart of the extractor $\bar{\mathcal{A}}$ in **GKEA**.

- Theorem A.3 (on page 192): If there exists a Type II adversary which is not in Type III, then there exists a PPT algorithm to break the weak-IND-sID-CPA security of the functional encryption scheme FE.

- Theorem A.4 (on page 200): If there exists a Type III adversary which is not in Type IV, then there exists a PPT algorithm to break Discrete Log Problem.

- (Part of)Theorem 11.1 (on page 204): If there exists a Type IV adversary which breaks our scheme, then there exists a PPT algorithm to break Assumption 4.

[1]This is possible, if the Type II adversary just invokes Type I adversary as a subroutine using the same random coin.

[2]Note that μ_i can take negative integer value, and $\mu_i > 1$ ($\mu_i < 1$, respectively) corresponds to the case of over-counting (under-counting, respectively) point $\vec{\boldsymbol{x}}_i$.

Informally, by combining all together, Theorem 11.1 states that if there exists a Type I adversary which outputs result X and a valid proof, then X has to be equal to the correct query result with o.h.p, under related computational assumptions. Note that Lemma A.2 and Theorem A.3 focus on the partial scheme $\widetilde{\mathcal{E}}$ and Theorem A.4 and Theorem 11.1 focus on the whole scheme $\mathcal{E}$.

The structure of our proof or the relationships among all assumptions, lemmas and theorems are shown as below. Note that the proof of correctness (Lemma 10.1) does not rely on any computational assumption.

$$\left.\begin{array}{r} \left.\begin{array}{r} \left.\begin{array}{r} \textbf{Assumption } 4 \\ \mathsf{BBG} \text{ is } \mathsf{IND\text{-}sID\text{-}CPA} \text{ secure [BBG05]} \Rightarrow \textbf{Theorem } 10.2 \\ \textbf{Assumption } 5 \Rightarrow \textbf{Lemma } A.1 \Rightarrow \textbf{Lemma } A.2 \end{array}\right\} \Rightarrow \textbf{Theorem } A.3 \\ \textbf{Assumption } 4 \Rightarrow \textbf{DLP Assumption} \end{array}\right\} \Rightarrow \textbf{Theorem } A.4 \\ \textbf{Assumption } 4 \\ \textbf{Lemma } 10.1 \end{array}\right\} \Rightarrow \textbf{Theorem } 11.1$$

11.2.3 The Preliminary Scheme is Secure

In this subsection, we prove that the preliminary scheme described in Chapter 8 is secure, following the proof framework in Section 11.2.2. This proof sketch serves as an illustration of our proof strategy and as a warm up of our full proof for the main scheme in appendix.

Theorem 11.2 *Suppose Assumption 4 and Assumption 5 hold for the cyclic multiplicative group G of order p and $\mathsf{F}_s(\cdot)$ is a random oracle. The preliminary scheme described in Chapter 8 is a $\mathcal{VRC}$ w.r.t. function $F(\cdot,\cdot)$ as defined in Section 9.1, under Definition 11. Namely, the preliminary scheme is* correct *and* sound *w.r.t. the aggregate count function F.*

Proof sketch of Theorem 11.2: The correctness part is straightforward. We just focus on soundness.

Part I: *The existence of Type I adversary implies the existence of Type II adversary, under* ***GKEA*** *Assumption 5.* Suppose there exists Type I adversary $\mathcal{B}$ against the preliminary scheme. We try to construct a Type II adversary $\bar{\mathcal{B}}$ based on **GKEA**

Assumption. We follow the proof framework for the statement that **KEA3** implies **KEA1** by Bellare *et al.* [BP04a]. First, we construct a **GKEA** adversary $\mathcal{A}_1$ based on the Type I adversary $\mathcal{B}$:

Construction of **GKEA** adversary $\mathcal{A}_1$ based on the Type I adversary $\mathcal{B}$

1. The input is $\{(u_i, u_i^\beta) : u_i \in G, 1 \le i \le m\}$, where $\beta \in \mathbb{Z}_p$ is unknown.
2. Choose two independent random elements $R_1, R_2 \in G$. There exist some unknown $\theta, v_0 \in G$, such that $R_1 = \theta v_0, R_2 = v_0^\beta$.
3. Let $\mathbf{D} = \{x_1, x_2, \ldots, x_{m+1}\} \subset [\mathcal{Z}]^d$ be the dataset. Let $u_{m+1} = 1$. Define function F_s: For any $x_i \in \mathbf{D}$, $\mathsf{F}_s(x_i) = u_i v_0$; for any $x \in [\mathcal{Z}]^d \setminus \mathbf{D}$, choose $z_x \in \mathbb{Z}_p^*$ at random and set $\mathsf{F}_s(x) = u_1^{z_x}$. *Note that $\mathsf{F}_s(x)^\beta$ still can be computed, although β is unknown.*
4. Invoke the preliminary scheme (Alice's part) with parameters β, θ and function F_s. *Note that tag $t_i = (\theta \mathsf{F}_s(x_i), \mathsf{F}_s(x_i)^\beta) = (u_i R_1, u_i^\beta R_2)$ still can be computed without knowing the values of $\theta, \beta, \mathsf{F}_s(x_i)$.*
5. Invoke the adversary $\mathcal{B}$ (Bob's part) to interact with Alice. For any query $\mathbf{R}$ made by $\mathcal{B}$, generate challenge-message from $\{\mathsf{F}_s(x)^\beta\}$ in this way: choose $\rho' \in \mathbb{Z}_p^*$ at random, and send $\{\mathsf{F}_s(x)^{\beta\rho'} : x \in \mathbf{R}\}$. *Note the actual random nonce $\rho = \beta\rho'$ is unknown.*
6. Obtain output $(X, \Psi_1, \Psi_2, \Psi_3)$ from $\mathcal{B}$, and output $\left(\frac{\Psi_1}{R_1^X}, \frac{\Psi_2}{R_2^X}\right)$.

If the adversary $\mathcal{B}$'s output $(X, \Psi_1, \Psi_2, \Psi_3)$ can pass Alice's verification step 1, i.e. $\left(\frac{\Psi_1}{\theta^X}\right)^\beta = \Psi_2$, then the **GKEA** adversary $\mathcal{A}_1$'s output is valid:

$$\left(\frac{\Psi_1}{R_1^X}\right)^\beta = \frac{\Psi_1^\beta}{\theta^{X\beta} v_0^{X\beta}} = \frac{\Psi_2}{v_0^{X\beta}} = \frac{\Psi_2}{R_2^X}.$$

By **GKEA** Assumption, there exists an extractor $\bar{\mathcal{A}}_1$, which outputs $\{\mu_i : 1 \le i \le m\}$ from the same input[3] of $\mathcal{A}$, such that $\frac{\Psi_2}{R_2^X} = \prod_{i=1}^m u_i^{\beta\mu_i}$. Then we can construct an adversary $\mathcal{B}_2$ based on $\bar{\mathcal{A}}_1$ which just outputs $\{\mu_i : 1 \le i \le m+1\}$, where

[3]Including the random coin.

$\mu_{m+1} = X - \sum_{i=1}^{m} \mu_i \mod p$. We conclude that $\mathcal{B}_2$ is a Type II adversary against the preliminary scheme, since

$$\prod_{i=1}^{m+1} \mathsf{F}_s(x_i)^{\beta\mu_i} = \mathsf{F}_s(x_{m+1})^{\beta\mu_{m+1}} \prod_{i=1}^{m} \mathsf{F}_s(x_i)^{\beta\mu_i} = R_2^{X-\sum_{i=1}^{m}\mu_i} \prod_{i=1}^{m} \left(u_i^{\beta} R_2\right)^{\mu_i} = R_2^X \prod_{i=1}^{m} u_i^{\beta\mu_i} = \Psi_2.$$

Part II: *If there exists a Type II adversary which is not in Type III, then there exists a PPT algorithm to break Computational Diffie Hellman Assumption 4.*

Construction of adversary $\mathcal{A}_2$ against Computational Diffie Hellman Problem

1. The input is $(v, v^a, u) \in G^3$ where $a \in \mathbb{Z}_p$. The goal is to find u^a.
2. Define function F_s: For each $x_i \in \mathbf{D}$, choose z_i at random from $\mathbb{Z}_p$ and set $\mathsf{F}_s(x_i) = v^{z_i} \in G$. For each $x_i \in [\mathcal{Z}]^d \backslash \mathbf{D}$, choose z_i at random from $\mathbb{Z}_p$ and set $\mathsf{F}_s(x_i) = v^{z_i} \in G$.
3. Choose i^* from $[N]$ at random and redefine $\mathsf{F}_s(x_{i^*})$: $\mathsf{F}_s(x_{i^*}) = u$.
4. Invoke the preliminary scheme (Alice's part) with function F_s and invoke the Type II adversary (Bob's part) to interact with Alice. The adversary's adaptive queries can be answered in the same way as in Step 5 of algorithm $\mathcal{A}_1$.
5. Let $\mathbf{R}$ be the adversary's challenging query range. If $x_{i^*} \in \mathbf{R}$, abort and fail. Otherwise, generate challenge-message with random nonce $\rho = a$: the value $\mathsf{F}_s(x_i)^a = (v^a)^{z_i}$ for $x_i \in \mathbf{R}$ can be computed, although a is unknown.
6. Let $(X, \Psi_1, \Psi_2, \Psi_3, \mu_1, \ldots, \mu_N)$ be the output of adversary. If $\mu_{i^*} \neq 0$, then compute φ as below and output $\varphi^{\mu_{i^*}^{-1}}$ *(This is the success case)*.

$$\varphi \leftarrow \frac{\Psi_3}{\prod_{1\le i\le N}^{i\neq i^*} \mathsf{F}_s(x_i)^{a\mu_i}}$$

Otherwise, abort and fail.

Let $S_{\#} = \{i : \mu_i \neq 0, x_i \in \mathbf{D} \cap \mathbf{R}^{\complement}\}$. Since the adversary is not in Type III, $S_{\#} \neq \emptyset$. It is easy to verify that, in the success case, i.e. when the adversary's output pass Alice's verifications and $\Psi_2 = \prod_{i=1}^{N} \mathsf{F}_s(x_i)^{\beta\mu_i}$ and $i^* \in S_{\#}$, then the output $\varphi^{\mu_{i^*}^{-1}} = \mathsf{F}_s(x_{i^*})^a = u^a$. Since the index i^* is uniformly random in $[N]$ and tags for all N points are identically distributed, there is non-negligible probability that the success case will be reached, and thus the value of u^a can be found.

Part III: *If there exists a Type III adversary which is not in Type IV, then there exists a PPT algorithm to break Discrete Log Assumption.*

Construction of adversary $\mathcal{A}_3$ against Discrete Log Problem

1. The input is $(v, v^a) \in G^2$. The goal is to find $a \in \mathbb{Z}_p$.
2. Define function F_s: For each $x_i \in \mathbf{D}$, choose y_i, z_i at random from $\mathbb{Z}_p$ and set $\mathsf{F}_s(x_i) = (v^a)^{y_i} \cdot v^{z_i} \in G$; otherwise, set $\mathsf{F}_s(x)$ to a random number in G.
3. Invoke the preliminary scheme (Alice's part) with function F_s and invoke the Type III adversary (Bob's part) to interact with Alice. The adversary's adaptive queries can be answered in the same way as in the preliminary scheme. *Note the simulator has all of private key.*
4. Let $\mathbf{R}$ be the challenging query range. Let $(X, \Psi_1, \Psi_2, \Psi_3, \{\mu_i : x_i \in \mathbf{D} \cap \mathbf{R}\})$ be the output of adversary for range $\mathbf{R}$ and let $(\hat{X}, \hat{\Psi}_1, \hat{\Psi}_2, \hat{\Psi}_3, \{\mu_i : x_i \in \mathbf{D} \cap \mathbf{R}^{\complement}\})$ be the output of adversary for the complement range $\mathbf{R}^{\complement}$.
5. If the adversary succeeds, we have
$$\prod_{i=1}^{N} \mathsf{F}_s(x_i)^{\beta\mu_i} = \prod_{x_i \in \mathbf{D}\cap\mathbf{R}} \mathsf{F}_s(x_i)^{\beta\mu_i} \prod_{x_i \in \mathbf{D}\cap\mathbf{R}^{\complement}} \mathsf{F}_s(x_i)^{\beta\mu_i} = \Psi_2 \cdot \hat{\Psi}_2 = \Delta = \prod_{i=1}^{N} \mathsf{F}_s(x_i)^{\beta}$$
6. Since the adversary is not Type IV, there exists some i, such that $\mu_i \neq 1$. Consequently, a univariable equation in unknown a of order 1 can be formed from the above equation. Solve this equation to get root a' and output a'.

Note that Computational Diffie Hellman Assumption 4 implies Discrete Log Assumption.

Part IV: *If there exists a Type IV adversary which breaks our scheme, then there exists a PPT algorithm to break Assumption 4.* Given input (u, u^β, v^β), we can construct an algorithm to find v. Choose a random number R. There exists some θ, such that $R = \theta v$. Similar as the construction of **GKEA** adversary $\mathcal{A}_1$ in Part I, from input $(u, u^\beta, \theta v, v^\beta)$, we can simulate the preliminary scheme (Alice's part). Let $\mathbf{R}$ be the challenging query range. let $(X', \Psi'_1, \Psi'_2, \Psi'_3, \{\mu_i : x_i \in \mathbf{D} \cap \mathbf{R}\})$ be an output of a Type IV adversary on query $\mathbf{R}$ and let $(X, \Psi_1, \Psi_2, \Psi_3, \{\mu_i : x_i \in \mathbf{D} \cap \mathbf{R}\})$ be the output of an honest Bob on query $\mathbf{R}$. If the Type IV adversary succeeds, then $\Psi'_2 = \prod_{x_i \in \mathbf{D} \cap \mathbf{R}} \mathsf{F}_s(x_i)^{\beta\mu_i}$, where $\mu_i = 1, 1 \leq i \leq N$, and $\left(\frac{\Psi'_1}{\theta^{X'}}\right)^\beta = \Psi'_2$. On the other hand, the output from an honest Bob also passes Alice's verifications: $\left(\frac{\Psi_1}{\theta^X}\right)^\beta = \Psi_2$ and $\Psi_2 = \prod_{x_i \in \mathbf{D} \cap \mathbf{R}} \mathsf{F}_s(x_i)^\beta = \Psi'_2$. Combining the above equations, we have $\left(\frac{\Psi'_1}{\Psi_1}\right)^{(X-X')^{-1}} = \theta$. As a result, we find the value of v: $v = \frac{R}{\theta}$.

Combining the results in Part I, II, III and IV, we conclude that no efficient adversary against the preliminary scheme can output a wrong result and forged a valid proof. In other words, the preliminary scheme is sound. □

11.3 Performance

In the setup phase, the computation complexity on Alice's side is $O(dN \log \mathcal{Z})$ and the dominant step is Step 3 of DEnc in Section 11.1. In the query phase, the communication overhead (in term of bits) per query is $O(d^2 \log^2 \mathcal{Z})$: (1) In $\mathsf{CollRes}$, the communication overhead is dominated by the size of challenge-message $\vec{\boldsymbol{\delta}}$, which is in $O(d \log^2 \mathcal{Z})$, i.e. $O(\log \mathcal{Z})$ decryption keys for each dimension, and each decryption key of size $O(\log \mathcal{Z})$; (2) There are $O(d)$ invocations of $\mathsf{CollRes}$ to process one query. Computation complexity on Bob's side is $O(dN \log \mathcal{Z})$ (bilinear map operations): (1) In $\mathsf{CollRes}$, $O(d|\mathbf{D} \cap \mathbf{R}| \log \mathcal{Z})$ computation is required for query range $\mathbf{R}$ and the dominant computation step is Step B1 of $\mathsf{CollRes}$ in Section 11.1; (2) In total, $\sum_{\ell=0}^{2d} O(d|\mathbf{D} \cap \mathbf{R}_\ell| \log \mathcal{Z}) = O(dN \log \mathcal{Z})$, where $\{\mathbf{R}_\ell : \ell \in [0, 2d]\}$ is a partition of the domain $[\mathcal{Z}]^d$. The computation complexity per query on Alice's side is $O(d^2 \log^2 \mathcal{Z})$

(group multiplications). The dominant computation step is Step A1 of $\mathsf{CollRes}$ in Section 11.1. The storage overhead on Bob's side, is $O(dN)$. The storage cost on Alice's side, i.e. the size of key and $\mathbf{D}_{\mathtt{A}}$, is $O(d+\ell)$, which can be reduced to $O(1)$ (precisely $O(1)$ number of seeds and each seed with length equal to the security parameter λ) using a pseudorandom function.

Chapter 12

Authenticating Other Types of Queries—Min,Max,Median and Range-Selection Queries

In this chapter, we apply the authentication scheme for count query proposed in the previous Chapter 11 as a blackbox, in order to authenticate other types of queries, including aggregate min/max/median and non-aggregate range-selection queries. The extended schemes require only $O(d^2 \log^2 \mathcal{Z})$ communication overhead for each type of queries which are supported.

To distinguish different queries, in this chapter, we denote an aggregate count query with multidimensional rectangular range $\mathbf{R}$ with notation $\textsc{Count}(\mathbf{R})$.

12.1 Min and Max

$\textsc{Min}$ and $\textsc{Max}$ queries can be authenticated in a similar way. We only elaborate the authentication method for $\textsc{Min}$ query as below.

A min query $\textsc{Min}(\mathbf{R}, \iota)$ with query range $\mathbf{R}$ and dimension $\iota \in [d]$, asks for the minimum coordinate value along the ι-th dimension among all data points within $\mathbf{D} \cap \mathbf{R}$, i.e. $\min\{\vec{\boldsymbol{x}}[\iota] : \vec{\boldsymbol{x}} \in \mathbf{D} \cap \mathbf{R}\}$. We find that $\textsc{Min}$ query can be converted to $\textsc{Count}$ query. The conversion is based on the below Proposition 1:

Proposition 1 *For any finite set S of numbers,*

$$c = \min S \qquad \Leftrightarrow \qquad c \in S \ \wedge \ |S| = |\{x : x \in S \wedge x \geq c\}|. \tag{12.1}$$

Suppose Alice asks Bob for the minimum coordinate value along the ι-th dimension of points within range $\mathbf{R}$. Bob returns a data point $\vec{\boldsymbol{x}}$, such that $\vec{\boldsymbol{x}}[\iota]$ is the minimum in the set S of coordinate values along the ι-th dimension of all points within range $\mathbf{R}$ (i.e. $S = \{\vec{\boldsymbol{x}}[\iota] : \vec{\boldsymbol{x}} \in \mathbf{R} \cap \mathbf{D}\}$). Meanwhile, Bob also sends a proof to show that $\vec{\boldsymbol{x}} \in \mathbf{D}$. Then Alice issues two COUNT queries to Bob: (1) COUNT($\mathbf{R}$), i.e. the size of set S; (2) COUNT $\left(\mathbf{R} \bigcap \left([\mathcal{Z}]^{\iota-1} \times [c, \mathcal{Z}] \times [\mathcal{Z}]^{d-\iota}\right)\right)$ where $c = \vec{\boldsymbol{x}}[\iota]$, i.e. the size of set $\{x : x \in S \wedge x \geq c\}$. Bob is expected to return the two count numbers with proofs following the scheme in previous Chapter 11. Alice believes c is the minimum value if all proofs are valid and the two count nubmers are equal. The interactive algorithm between Alice and Bob is showed as below.

Authenticating Query MIN($\mathbf{R}, \iota$)

1. Alice sends $(\mathbf{R}, \iota)$ to Bob.
2. Bob finds $\vec{\boldsymbol{x}}^* = \arg\min_{\vec{\boldsymbol{x}} \in \mathbf{D} \cap \mathbf{R}} \vec{\boldsymbol{x}}[\iota]$ and sends $\vec{\boldsymbol{x}}^*$ to Alice.
3. Alice issues a count query COUNT($\{\vec{\boldsymbol{x}}^*\}$) with range $\{\vec{\boldsymbol{x}}^*\}$ to Bob and gets authenticated query result N_0.
4. Alice sets $c = \vec{\boldsymbol{x}}^*[\iota]$ and finds the range $\mathbf{R}_c = \mathbf{R} \cap \left([\mathcal{Z}]^{\iota-1} \times [c, \mathcal{Z}] \times [\mathcal{Z}]^{d-\iota}\right)$.
5. Alice issues two count queries COUNT($\mathbf{R}$) and COUNT($\mathbf{R}_c$) to Bob and gets authenticated results N_1 and N_2.
6. Alice accepts c as the minimum, if all verifications succeed and $N_0 \geq 1$ and $N_1 = N_2$.

Proposition 1 and our main Theorem 11.1 in Part II of this dissertation, imply the following Corollary 12.1:

Corollary 12.1 *The above extended scheme is a $\mathcal{VRC}$ (Verifiable Remote Computing protocol) w.r.t.* MIN *query, i.e. it is* correct *and* sound *to authenticate* MIN *query.*

12.2 Median

MEDIAN can also be converted into COUNT. Quartile or percentile queries can be handled in a similar way.

Proposition 2 *Let S be a finite set of numbers.*

$$c \text{ is the median in set } S \quad \Leftrightarrow$$
$$c \in S \quad \wedge \quad |\{x : x \in S \wedge x \leq c\}| \geq \lceil \tfrac{|S|}{2} \rceil \quad \wedge \quad |\{x : x \in S \wedge x \geq c\}| \geq \lceil \tfrac{|S|}{2} \rceil \tag{12.2}$$

Suppose Alice asks Bob for the median coordinate value along the ι-th dimension of points within range $\mathbf{R}$. Bob returns a data point $\vec{\boldsymbol{x}}$, such that $\vec{\boldsymbol{x}}[\iota]$ is the median in the set S of coordinate values along the ι-th dimension of all points within range $\mathbf{R}$ (i.e. $S = \{\vec{\boldsymbol{x}}[\iota] : \vec{\boldsymbol{x}} \in \mathbf{D} \cap \mathbf{R}\}$). Meanwhile, Bob also sends a proof to show that $\vec{\boldsymbol{x}} \in \mathbf{D}$. Then Alice issues three COUNT queries to Bob: (1) COUNT($\mathbf{R}$), i.e. the size of set S; (2) COUNT $\left(\mathbf{R} \bigcap \left([\mathcal{Z}]^{\iota-1} \times [c, \mathcal{Z}] \times [\mathcal{Z}]^{d-\iota}\right)\right)$ where $c = \vec{\boldsymbol{x}}[\iota]$, i.e. the size of set $\{x : x \in S \wedge x \geq c\}$; (3) COUNT $\left(\mathbf{R} \bigcap \left([\mathcal{Z}]^{\iota-1} \times [1, c] \times [\mathcal{Z}]^{d-\iota}\right)\right)$ i.e. the size of set $\{x : x \in S \wedge x \leq c\}$. Bob is expected to return the three count numbers N_1, N_2 and N_3 with proofs following the scheme in previous Chapter 11. Alice believes c is the median value if all proofs are valid and $N_2 \geq \lceil \frac{N_1}{2} \rceil$ and $N_3 \geq \lceil \frac{N_1}{2} \rceil$.

The interactive algorithm between Alice and Bob is showned as below. Note that when the size of S is even, there are two medians. For simplicity of presentation of the algorithm, we request Bob to return either one of the two medians, instead of both.

Authenticating Query MEDIAN($\mathbf{R}, \iota$)

1. Alice sends $(\mathbf{R}, \iota)$ to Bob.
2. Bob finds $\vec{\boldsymbol{x}}^*$ such that $\vec{\boldsymbol{x}}^*[\iota]$ is a median among $\{\vec{\boldsymbol{x}}[\iota] : \vec{\boldsymbol{x}} \in \mathbf{D}\}$ and sends $\vec{\boldsymbol{x}}^*$ to Alice.
3. Alice issues a count query COUNT($\{\vec{\boldsymbol{x}}^*\}$) with range $\{\vec{\boldsymbol{x}}^*\}$ to Bob and gets authenticated query result N_0.

4. Alice sets $c = \vec{\boldsymbol{x}}^*[\iota]$ and finds the range $\mathbf{R}_c^+ = \mathbf{R} \cap \big([\mathcal{Z}]^{\iota-1} \times [c, \mathcal{Z}] \times [\mathcal{Z}]^{d-\iota}\big)$ and range $\mathbf{R}_c^- = \mathbf{R} \cap \big([\mathcal{Z}]^{\iota-1} \times [1, c] \times [\mathcal{Z}]^{d-\iota}\big)$.
5. Alice issues three count queries $\textsc{Count}(\mathbf{R})$, $\textsc{Count}(\mathbf{R}_c^+)$ and $\textsc{Count}(\mathbf{R}_c^-)$ to Bob and gets authenticated results N_1, N_2 and N_3.
6. Alice accepts c as the median, if all verifications succeed and $N_0 \geq 1$ and $N_2 \geq \lceil \frac{N_1}{2} \rceil$ and $N_3 \geq \lceil \frac{N_1}{2} \rceil$.

Proposition 2 and Theorem 11.1 for count query, imply the following Corollary 12.2:

Corollary 12.2 *The above extended scheme is a $\mathcal{VRC}$ (Verifiable Remote Computing protocol) w.r.t.* MEDIAN *query, i.e. it is* correct *and* sound *to authenticate* MEDIAN *query.*

12.3 Range Selection

With the help of an aggregate signature scheme (e.g. BLS signature [BLS04,BGLS03]), it is straightforward to extend our scheme for count query to authenticate range selection query with multidimensional rectangular range $\mathbf{R}$, which is denoted with $\textsc{RangeSelect}(\mathbf{R})$ and asks for the set $\{\vec{\boldsymbol{x}} : \vec{\boldsymbol{x}} \in \mathbf{D} \cap \mathbf{R}\}$. To the best of our knowledge, this is the first solution that authenticates multidimensional range selection query with sublinear communication overhead in the worst case and polynomial storage on Bob's side.

We assume the dataset $\mathbf{D}$ is a set of *distinct* points. The authentication scheme for range selection query is as follows:

Authenticating Query $\textsc{RangeSelect}(\mathbf{R})$

1. In the setup, Alice generates a signature $\mathsf{Sig}(\vec{\boldsymbol{x}})$ for each data point $\vec{\boldsymbol{x}} \in \mathbf{D}$ using an aggregate signature scheme, and sends all signatures to Bob.

2. To answer a range selection query with range $\mathbf{R}$, Bob finds the set $S = \{\vec{\boldsymbol{x}} : \vec{\boldsymbol{x}} \in \mathbf{D} \cap \mathbf{R}\}$ and computes an aggregated signature $\mathsf{Sig}(S)$ for set S from signatures $\mathsf{Sig}(\vec{\boldsymbol{x}})$'s for

point $\vec{\boldsymbol{x}} \in \mathbf{D} \cap \mathbf{R}$, using the aggregate signature scheme. Bob sends $(S, \mathsf{Sig}(S))$ to Alice.

3. Alice verifies: (1) Is S a set of distinct points? (2) Is S a subset of query range $\mathbf{R}$? (3) Is $\mathsf{Sig}(S)$ a valid signature for S?

4. Alice issues a count query with range $\mathbf{R}$ to Bob using our scheme presented in Section 11.1 and gets authenticated result N_0.

5. Alice verifies whether $|S| = N_0$.

6. Alice accepts S as the query result, if all verifications succeed.

We remark that the extra aggregate signature for each data point is actually unnecessary, since our authentication tag can take the role of aggregate signature in this particular application.

Corollary 12.3 *The above extended scheme is a $\mathcal{VRC}$ (Verifiable Remote Computing protocol) w.r.t.* RangeSelect *query, i.e. it is* correct *and* sound *to authenticate* RangeSelect *query.*

Chapter 13

Conclusion

In this dissertation, we studied two problems in verifiable cloud computing: proofs of storage and verifiable outsourced database. We have constructed efficient solutions to these problems, by devising new homomorphic cryptographic methods.

Proofs of Storage.

In this first part of this dissertation, we designed three efficient solutions POS1, POS2 and POS3 to proofs of storage problem, based on some underlying linearly homomorphic authentication methods. Each of these three solutions enables a user Alice to remotely and reliably verify the integrity of her data files stored in a cloud storage server Bob, without trusting in Bob and without retrieving the files. All of these three solutions require only $\mathcal{O}(\lambda)$ communication, storage and computation cost on Alice's side, independent on the size of the file stored in the cloud storage. Furthermore, in both POS2 and POS3, the storage overhead (due to error erasure encoding and authentication information) is only a fraction (e.g. 2% or 3%) of the original file size, and the latency for one verification is within one second independent on the file size, confirmed by empirical study and analysis. We provided full security proofs for all of three solutions. The proposed predicate-homomorphic MAC scheme may have independent interests.

Verifiable Outsourced Database.

In the second part, we proposed a new functional encryption scheme. This functional encryption scheme allows a third party, with a delegation key which is generated on the fly, to compute a pre-determined two-input one-way function value from a ciphertext, where the first input is the corresponding plaintext and the second input is secretly embedded in the delegation key, without knowing the value of of the plaintext.

We applied the proposed functional encryption scheme to construct a scheme to authenticate aggregate range query over static multidimensional outsourced dataset, and the communication complexity (in term of bits) is $O(d^2 \log^2 \mathcal{Z})$ (d is the dimension and each data point is in domain $[\mathcal{Z}]^d$). Our solution for aggregate count query leads to solutions for aggregate min/max/median and non-aggregate range selection queries with similar complexities.

The proposed functional encryption scheme and the idea of implementing functional encryption by exploiting polymorphic property of existing encryption schemes may have independent interests.

Bibliography

[AB09] Shweta Agrawal and Dan Boneh. Homomorphic MACs: MAC-Based Integrity for Network Coding. In *ACNS '09: International Conference on Applied Cryptography and Network Security*, pages 292–305, 2009.

[ABC$^+$07] Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary Peterson, and Dawn Song. Provable data possession at untrusted stores. In *CCS '07: ACM conference on Computer and communications security*, pages 598–609, 2007.

[ABC$^+$11a] Jae Ahn, Dan Boneh, Jan Camenisch, Susan Hohenberger, abhi shelat, and Brent Waters. Computing on Authenticated Data. Cryptology ePrint Archive, Report 2011/096, 2011. `http://eprint.iacr.org/`.

[ABC$^+$11b] Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring, Osama Khan, Lea Kissner, Zachary Peterson, and Dawn Song. Remote data checking using provable data possession. *ACM Transactions on Information and System Security*, 14:12:1–12:34, 2011.

[ACK08] Mikhail Atallah, YounSun Cho, and Ashish Kundu. Efficient Data Authentication in an Environment of Untrusted Third-Party Distributors. In *ICDE '08: IEEE International Conference on Data Engineering*, pages 696–704, 2008.

[ADPMT08] Giuseppe Ateniese, Roberto Di Pietro, Luigi Mancini, and Gene Tsudik. Scalable and efficient provable data possession. In *SecureComm '08:*

International conference on Security and privacy in communication netowrks, pages 9:1–9:10, 2008.

[AF07] Masayuki Abe and Serge Fehr. Perfect NIZK with adaptive soundness. In *TCC '07: Theory of Cryptography Conference*, pages 118–136, 2007.

[AKK09] Giuseppe Ateniese, Seny Kamara, and Jonathan Katz. Proofs of Storage from Homomorphic Identification Protocols. In *ASIACRYPT '09: International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, pages 319–333, 2009.

[Ama] AmazonForum. Major Outage for Amazon S3 and EC2. `https://forums.aws.amazon.com/thread.jspa?threadID=19714&start=15&tstart=0`.

[ATS04] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys*, 36(4):335–371, 2004.

[BB04] Dan Boneh and Xavier Boyen. Short Signatures Without Random Oracles. In *EUROCRYPT '04: Annual International Conference on Advances in Cryptology*, pages 56–73, 2004.

[BB08] Dan Boneh and Xavier Boyen. Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups. *Journal of Cryptology*, 21:149–177, 2008.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical Identity Based Encryption with Constant Size Ciphertext. In *EUROCRYPT '05: Annual International Conference on Advances in Cryptology*, pages 440–456, 2005.

[BBST02] Christopher Batten, Kenneth Barr, Arvind Saraf, and Stanley Trepetin. pStore: A Secure Peer-to-Peer Backup System. Technical Report MIT-LCS-TM-632, MIT, 2002.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37:156–189, 1988.

[BCK96] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying Hash Functions for Message Authentication. In *CRYPTO '96: Annual International Cryptology Conference on Advances in Cryptology*, pages 1–15, 1996.

[BEG$^+$91] Manuel Blum, Will Evans, Peter Gemmell, Sampath Kannan, and Moni Naor. Checking the correctness of memories. In *FOCS '91: Annual Symposium on Foundations of Computer Science*, pages 90–99, 1991.

[BFKW09] Dan Boneh, David Freeman, Jonathan Katz, and Brent Waters. Signing a Linear Subspace: Signature Schemes for Network Coding. In *PKC '09: International Conference on Practice and Theory in Public Key Cryptography*, pages 68–87, 2009.

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In *EUROCRYPT '03: Annual International Conference on Advances in Cryptology*, pages 416–432, 2003.

[BGV11] Siavosh Benabbas, Rosario Gennaro, and Yevgeniy Vahlis. Verifiable Delegation of Computation over Large Datasets. In *CRYPTO '11: Annual International Cryptology Conference on Advances in Cryptology*, pages 111–131, 2011.

[BH05] Boaz Barak and Shai Halevi. A model and architecture for pseudorandom generation with applications to `/dev/random`. In *CCS '05: Proceedings of the 12th ACM conference on Computer and communications security*, pages 203–212, 2005.

[BJO09a] Kevin Bowers, Ari Juels, and Alina Oprea. HAIL: a high-availability and integrity layer for cloud storage. In *CCS '09: ACM conference on Computer and communications security*, pages 187–198, 2009.

[BJO09b] Kevin Bowers, Ari Juels, and Alina Oprea. Proofs of retrievability: theory and implementation. In *CCSW '09: ACM workshop on Cloud computing security*, pages 43–54, 2009.

[BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham. Short Signatures from the Weil Pairing. *Journal of Cryptology*, 17(4):297–319, 2004.

[Blu81] Manuel Blum. Coin Flipping by Telephone. In *CRYPTO '81: Annual International Cryptology Conference on Advances in Cryptology*, pages 11–15, 1981.

[BP04a] Mihir Bellare and Adriana Palacio. The Knowledge-of-Exponent Assumptions and 3-Round Zero-Knowledge Protocols. In *CRYPTO '04: Annual International Cryptology Conference on Advances in Cryptology*, pages 273–289, 2004.

[BP04b] Mihir Bellare and Adriana Palacio. Towards Plaintext-Aware Public-Key Encryption Without Random Oracles. In *ASIACRYPT '04: International Conference on the Theory and Application of Cryptology and Information Security*, pages 48–62, 2004.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional Encryption: Definitions and Challenges. In *TCC '11: Theory of Cryptography Conference*, pages 253–273, 2011.

[Bus] BusinesSinsider. Amazon's Cloud Crash Disaster Permanently Destroyed Many Customers' Data. `http://www.businessinsider.com/amazon-lost-data-2011-4`.

[BW07] Dan Boneh and Brent Waters. Conjunctive, Subset, and Range Queries on Encrypted Data. In *TCC '07: Theory of Cryptography Conference*, pages 535–554, 2007.

[Che52] Herman Chernoff. A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the sum of Observations. *Annals of Mathematical Statistics*, 23(4):493507, 1952.

[CKBA08] Reza Curtmola, Osama Khan, Randal Burns, and Giuseppe Ateniese. MR-PDP: Multiple-Replica Provable Data Possession. In *ICDCS '08: International Conference on Distributed Computing Systems*, pages 411–420, 2008.

[CKV10] Kai-Min Chung, Yael Kalai, and Salil Vadhan. Improved Delegation of Computation Using Fully Homomorphic Encryption. In *CRYPTO '10: Annual International Cryptology Conference on Advances in Cryptology*, pages 483–501, 2010.

[CMH$^+$08] Hong Chen, Xiaonan Ma, Windsor Hsu, Ninghui Li, and Qihua Wang. Access Control Friendly Query Verification for Outsourced Data Publishing. In *ESORICS '08: European Symposium on Research in Computer Security*, pages 177–191, 2008.

[CT09] Weiwei Cheng and Kian-Lee Tan. Query assurance verification for outsourced multi-dimensional databases. *Journal of Computer Security*, 17(1):101–126, 2009.

[CX08] Ee-Chien Chang and Jia Xu. Remote Integrity Check with Dishonest Storage Server. In *ESORICS '08: European Symposium on Research in Computer Security*, pages 223–237, 2008.

[Dam92] Ivan Damgård. Towards Practical Public Key Systems Secure Against Chosen Ciphertext Attacks. In *CRYPTO '91: Annual International Cryptology Conference on Advances in Cryptology*, pages 445–456, 1992.

[Den06] Alexander W. Dent. The Cramer-Shoup Encryption Scheme Is Plaintext Aware in the Standard Model. In *EUROCRYPT '06: Annual International Conference on Advances in Cryptology*, pages 289–307, 2006.

[DGMS01] Premkumar Devanbu, Michael Gertz, Charles Martel, and Stuart Stubblebine. Authentic Third-party Data Publication. In *Proceedings of the IFIP TC11/ WG11.3 Fourteenth Annual Working Conference on Database Security*, pages 101–112, 2001.

[DGMS03] Premkumar Devanbu, Michael Gertz, Charles Martel, and Stuart Stubblebine. Authentic data publication over the internet. *Journal of Computer Security*, 11(3):291–314, 2003.

[DH76] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[Dif09] Whitfield Diffie. How Secure Is Cloud Computing? (An Interview), 2009. `http://www.technologyreview.com/computing/23951/`.

[DNRV09] Cynthia Dwork, Moni Naor, Guy Rothblum, and Vinod Vaikuntanathan. How Efficient Can Memory Checking Be?. In *TCC '09: Theory of Cryptography Conference*, pages 503–520, 2009.

[DQS03] Yves Deswarte, Jean-Jacques Quisquater, and Ayda Saïdane. Remote Integrity Checking: How to Trust Files Stored on Untrusted Servers . In *Conference on Integrity and Internal Control in Information Systems*, pages 1–11, 2003.

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard.* 2002.

[Dro11] Dropbox. Dropbox forums on data loss topic, 2011. `http://forums.dropbox.com/tags.php?tag=data-loss`.

[DVW09] Yevgeniy Dodis, Salil Vadhan, and Daniel Wichs. Proofs of Retrievability via Hardness Amplification. In *TCC '09: Theory of Cryptography Conference on Theory of Cryptography*, pages 109–127, 2009.

[EKPT09] Chris Erway, Alptekin Küpçü, Charalampos Papamanthou, and Roberto Tamassia. Dynamic provable data possession. In *CCS '09:*

ACM conference on Computer and communications security, pages 213–222, 2009.

[FB06] Décio Filho and Paulo Barreto. Demonstrating data possession and uncheatable data transfer. Cryptology ePrint Archive, Report 2006/150, 2006. `http://eprint.iacr.org/`.

[Gen09] Craig Gentry. Fully Homomorphic Encryption using Ideal Lattices. In *STOC '09: ACM Symposium on Theory of Computing*, pages 169–178, 2009.

[Gen10] Craig Gentry. Toward Basing Fully Homomorphic Encryption on Worst-Case Hardness. In *CRYPTO '10: Annual International Cryptology Conference on Advances in Cryptology*, pages 116–137, 2010.

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers. In *CRYPTO '10: Annual International Cryptology Conference on Advances in Cryptology*, pages 465–482, 2010.

[GMP] GMP. The GNU Multiple Precision Arithmetic Library. `http://www.gmplib.org/`.

[Gol06] Oded Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools.* Cambridge University Press, New York, NY, USA, 2006.

[Goo11] Google. Gmail back soon for everyone, 2011. `http://gmailblog.blogspot.com/2011/02/gmail-back-soon-for-everyone.html`.

[Gro10] Jens Groth. Short Pairing-Based Non-interactive Zero-Knowledge Arguments. In *ASIACRYPT '10: International Conference on the Theory and Application of Cryptology and Information Security*, pages 321–340, 2010.

[GTT08] Michael Goodrich, Roberto Tamassia, and Nikos Triandopoulos. Super-Efficient Verification of Dynamic Outsourced Databases. In *CT-RSA*

'08: The Cryptographer's Track at the RSA Conference on Topics in Cryptology, pages 407–424, 2008.

[GZ07] Tingjian Ge and Stanley Zdonik. Answering Aggregation Queries in a Secure System Model. In *VLDB '07: International Conference on Very Large Data Bases*, pages 519–530, 2007.

[HHSY06] Stuart Haber, William Horne, Tomas Sander, and Danfeng Yao. Privacy-Preserving Verification of Aggregate Queries on Outsourced Databases. Technical report, HP Laboratories, 2006. HPL-2006-128.

[HILM02] Hakan Hacigümüs, Balakrishna Iyer, Chen Li, and Sharad Mehrotra. Executing SQL over Encrypted Data in the Database Service Provider Model. In *SIGMOD '02: ACM SIGMOD International conference on Management of data*, pages 216–227, 2002.

[HIM04] Hakan Hacigümüs, Balakrishna Iyer, and Sharad Mehrotra. Efficient Execution of Aggregation Queries over Encrypted Relational Databases. In *DASFAA '04: Database Systems for Advanced Applications*, pages 125–136, 2004.

[HT98] Satoshi Hada and Toshiaki Tanaka. On the Existence of 3-Round Zero-Knowledge Protocols. In *CRYPTO '98: Annual International Cryptology Conference on Advances in Cryptology*, pages 408–423, 1998.

[JK07] Ari Juels and Burton Kaliski, Jr. Pors: proofs of retrievability for large files. In *CCS '07: ACM conference on Computer and communications security*, pages 584–597, 2007.

[Kra05] Hugo Krawczyk. HMQV: A High-Performance Secure Diffie-Hellman Protocol. In *CRYPTO '05: Annual International Cryptology Conference on Advances in Cryptology*, pages 546–566, 2005.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products.

In *EUROCRYPT '08: Annual International Conference on Advances in Cryptology*, pages 146–162, 2008.

[KZG10] Aniket Kate, Gregory Zaverucha, and Ian Goldberg. Constant-Size Commitments to Polynomials and Their Applications. In *ASIACRYPT '10: International Conference on the Theory and Application of Cryptology and Information Security*, pages 177–194, 2010.

[LD06] Jinyang Li and Frank Dabek. F2F: Reliable Storage in Open Networks. In *IPTPS '06: International Workshop on Peer-to-Peer Systems*, 2006.

[LHKR06] Feifei Li, Marios Hadjieleftheriou, George Kollios, and Leonid Reyzin. Dynamic authenticated index structures for outsourced databases. In *SIGMOD '06: ACM SIGMOD International conference on Management of data*, pages 121–132, 2006.

[LHKR10] Feifei Li, Marios Hadjieleftheriou, George Kollios, and Leonid Reyzin. Authenticated index structures for aggregation queries. *ACM Transactions on Information and System Security*, 13:32:1–32:35, 2010.

[LOS$^+$10] Allison Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption. In *EUROCRYPT '10: Annual International Conference on Advances in Cryptology*, pages 62–91, 2010.

[LTT10] Anna Lysyanskaya, Roberto Tamassia, and Nikos Triandopoulos. Authenticated error-correcting codes with applications to multicast authentication. *ACM Transactions on Information and System Security*, 13:17:1–17:34, 2010.

[Mer80] Ralph Merkle. Protocols for Public Key Cryptosystems. In *SP '80: IEEE Symposium on Security and Privacy*, page 122, 1980.

[Mic11] Microsoft. Hotmail email access issue now resolved, 2011. `http://windowsteamblog.com/windows_live/b/windowslive/archive/2011/01/03/hotmail-email-access-issue-now-resolved.aspx`.

[Mil75] Gary Miller. Riemann's hypothesis and tests for primality. In *STOC '75: ACM Symposium on Theory of Computing*, pages 234–239, 1975.

[MND$^+$04] Charles Martel, Glen Nuckolls, Premkumar Devanbu, Michael Gertz, April Kwong, and Stuart Stubblebine. A General Model for Authenticated Data Structures. *Algorithmica*, 39(1):21–41, 2004.

[MNT06] Einar Mykletun, Maithili Narasimha, and Gene Tsudik. Authentication and Integrity in Outsourced Databases. *Trans. Storage*, 2(2):107–138, 2006.

[MS58] N. Macon and A. Spitzbart. Inverses of Vandermonde Matrices. *The American Mathematical Monthly*, 65(2):95–100, 1958.

[MSP09] Kyriakos Mouratidis, Dimitris Sacharidis, and Hweehwa Pang. Partially materialized digest scheme: an efficient verification method for outsourced databases. *The VLDB Journal*, 18(1):363–381, 2009.

[MT06] Einar Mykletun and Gene Tsudik. Aggregation Queries in the Database-As-a-Service Model. In *IFIP WG 11.3 Working Conference on Data and Applications Security*, pages 89–103, 2006.

[NIS02] NIST. National Institute of Standards and Technology. Secure hash standard (SHS). FIPS 180-2, August 2002.

[NR05] Moni Naor and Guy Rothblum. The complexity of online memory checking. In *FOCS '05: Symposium on Foundations of Computer Science*, pages 573–584, 2005.

[NR09] Moni Naor and Guy Rothblum. The complexity of online memory checking. *Journal of the ACM*, 56:2:1–2:46, 2009.

[O'N10] Adam O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. `http://eprint.iacr.org/`.

[Ope] OpenSSL. OpenSSL Project. `http://www.openssl.org/`.

[OT10] Tatsuaki Okamoto and Katsuyuki Takashima. Fully Secure Functional Encryption with General Relations from the Decisional Linear Assumption. In *CRYPTO '10: Annual International Cryptology Conference on Advances in Cryptology*, pages 191–208, 2010.

[PB61] W. Peterson and D. Brown. Cyclic Codes for Error Detection. *Proceedings of the IRE*, 49(1):228–235, 1961.

[PJRT05] HweeHwa Pang, Arpit Jain, Krithi Ramamritham, and Kian-Lee Tan. Verifying completeness of relational query results in data publishing. In *SIGMOD '05: ACM SIGMOD International conference on Management of data*, pages 407–418, 2005.

[PS85] Franco Preparata and Michael Shamos. *Computational geometry: an introduction.* Springer-Verlag New York, Inc., 1985.

[PT08] Hweehwa Pang and Kian-Lee Tan. Verifying Completeness of Relational Query Answers from Online Servers. *ACM Transactions on Information and System Security*, 11(2):1–50, 2008.

[PZM09] HweeHwa Pang, Jilian Zhang, and Kyriakos Mouratidis. Scalable Verification for Outsourced Dynamic Databases. *Proc. VLDB Endow.*, 2:802–813, 2009.

[RS60] Irving Reed and Gustave Solomon. Polynomial Codes over Certain Finite Fields. *Journal of the Society for Industrial and Applied Mathematics (SIAM)*, 8(2):300–304, 1960.

[RSA78] Ronald Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[SBC$^+$07] Elaine Shi, John Bethencourt, Hubert Chan, Dawn Song, and Adrian Perrig. Multi-Dimensional Range Query over Encrypted Data. In *SP '07: IEEE Symposium on Security and Privacy*, pages 350–364, 2007.

[SCG$^+$03] Edward Suh, Dwaine Clarke, Blaise Gassend, Marten van Dijk, and Srinivas Devadas. Efficient Memory Integrity Verification and Encryption for Secure Processors. In *MICRO '03: Annual IEEE/ACM International Symposium on Microarchitecture*, pages 339–350, 2003.

[Sio05] Radu Sion. Query Execution Assurance for Outsourced Databases. In *VLDB '05: International Conference on Very Large Data Bases*, pages 601–612, 2005.

[SV03] Amit Sahai and Salil Vadhan. A Complete Problem for Statistical Zero Knowledge. *Journal of the ACM*, 50:196–249, 2003.

[SW05] Amit Sahai and Brent Waters. Fuzzy Identity-Based Encryption. In *EUROCRYPT '05: Annual International Conference on Advances in Cryptology*, pages 457–473, 2005.

[SW08a] Hovav Shacham and Brent Waters. Compact Proofs of Retrievability. In *ASIACRYPT '08: International Conference on the Theory and Application of Cryptology and Information Security*, pages 90–107, 2008.

[SW08b] Elaine Shi and Brent Waters. Delegating Capabilities in Predicate Encryption Systems. In *ICALP '08: International colloquium on Automata, Languages and Programming, Part II*, pages 560–578, 2008.

[Twi] Twitter. Tweetdeck. `http://money.cnn.com/2012/03/30/technology/tweetdeck-bug-twitter/`.

[TYH$^+$09] Brian Thompson, Danfeng Yao, Stuart Haber, William Horne, and Tomas Sander. Privacy-Preserving Computation and Verification of Aggregate Queries on Outsourced Databases. In *PETS '09: Privacy Enhancing Technologies Symposium*, 2009.

[vDGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully Homomorphic Encryption over the Integers. In *EUROCRYPT '10: Annual International Conference on Advances in Cryptology*, pages 24–43, 2010.

[Wik11] Wikipedia. 2009 Sidekick data loss, 2011. `http://en.wikipedia.org/wiki/2009_Sidekick_data_loss`.

[wir] wired.com. Dropbox left user accounts unlocked for 4 hours sunday. `http://www.wired.com/threatlevel/2011/06/dropbox/`.

[WS07] Jiang Wu and Doug Stinson. An Efficient Identification Protocol and the Knowledge-of-Exponent Assumption. Cryptology ePrint Archive, Report 2007/479, 2007. `http://eprint.iacr.org/`.

[WWL$^+$09] Qian Wang, Cong Wang, Jin Li, Kui Ren, and Wenjing Lou. Enabling public verifiability and data dynamics for storage security in cloud computing. In *ESORICS'09: European conference on Research in computer security*, pages 355–370, 2009.

[WWRL10] Cong Wang, Qian Wang, Kui Ren, and Wenjing Lou. Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing. In *INFOCOM '10: Annual IEEE International Conference on Computer Communications*, pages 525–533, 2010.

[XC12] Jia Xu and Ee-Chien Chang. Towards Efficient Proofs of Retrievability. In *AsiaCCS '12: ACM Symposium on Information, Computer and Communications Security*, 2012.

[XWYM07] Min Xie, Haixun Wang, Jian Yin, and Xiaofeng Meng. Integrity auditing of outsourced data. In *VLDB '07: International conference on Very large data bases*, pages 782–793, 2007.

[YJ11] Kan Yang and Xiaohua Jia. Data storage auditing service in cloud computing: challenges, methods and opportunities. *(will appear in) Journal of World Wide Web*, 2011.

[YPPK09] Yin Yang, Dimitris Papadias, Stavros Papadopoulos, and Panos Kalnis. Authenticated join processing in outsourced databases. In *SIGMOD '09: ACM SIGMOD International conference on Management of data*, pages 5–18, 2009.

[Zhe97] Yuliang Zheng. Digital Signcryption or How to Achieve Cost(Signature & Encryption) $\ll$ Cost(Signature) + Cost(Encryption). In *CRYPTO '97: Annual International Cryptology Conference on Advances in Cryptology*, pages 165–179, 1997.

Appendix A

Security Proof

A.1 BBG HIBE

We restate the BBG HIBE scheme proposed by Boneh *et al.* [BBG05], to make this paper self-contained. Let p be a λ bits safe prime, and $e : \mathbb{G} \times \mathbb{G} \to \widetilde{\mathbb{G}}$ be a bilinear map, where the orders of $\mathbb{G}$ and $\widetilde{\mathbb{G}}$ are both p. The HIBE scheme contains four algorithms ($\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}$), which are described as follows.

$\mathsf{Setup}(\ell)$

To generate system parameters for an HIBE of maximum depth ℓ, select a random generator $g \in \mathbb{G}$, a random $\alpha \in \mathbb{Z}_p$, and set $g_1 = g^\alpha$. Next, pick random elements $g_2, g_3, h_1, \ldots, h_\ell \in \mathbb{G}$. The public parameters and the master key are

$$\mathsf{params} = (g, g_1, g_2, g_3, h_1, \ldots, h_\ell, \Omega = e(g_1, g_2)), \quad \mathsf{master\text{-}key} = g_2^\alpha.$$

$\mathsf{KeyGen}(d_{\mathsf{id}|k-1}, \mathsf{id})$

To generate a private key d_{id} for an identity $\mathsf{id} = (I_1, \ldots, I_k) \in \left(\mathbb{Z}_p^*\right)^k$ of depth $k \leq \ell$, using the master secret key $\mathsf{master\text{-}key}$, pick a random $r \in \mathbb{Z}_p$ and output

$$d_{\mathsf{id}} = \left(g_2^\alpha \cdot \left(h_1^{I_1} \ldots h_k^{I_k} \cdot g_3\right)^r, \ g^r, \ h_{k+1}^r, \ldots, h_\ell^r\right) \in \mathbb{G}^{2+\ell-k}$$

The private key for id can be generated incrementally, given a private key for the parent identity $\mathsf{id}_{|k-1} = (I_1, \ldots, I_{k-1}) \in \left(\mathbb{Z}_p^*\right)^{k-1}$. Let

$$d_{\mathsf{id}|k-1} = \left(g_2^{\alpha} \cdot \left(h_1^{I_1} \ldots h_{k-1}^{I_{k-1}} \cdot g_3\right)^{r'},\ g^{r'},\ h_k^{r'}, \ldots, h_\ell^{r'}\right) = (K_0, K_1, W_k, \ldots, W_\ell)$$

be the private key for $\mathsf{id}_{|k-1}$. To generate d_{id}, pick a random $t \in \mathbb{Z}_p$ and output

$$d_{\mathsf{id}} = \left(K_0 \cdot W_k^{I_k} \cdot \left(h_1^{I_1} \ldots h_k^{I_k} \cdot g_3\right)^t,\ K_1 \cdot g^t,\ W_{k+1} \cdot h_{k+1}^t, \ldots, W_\ell \cdot h_\ell^t\right).$$

This private key is a properly distributed private key for $\mathsf{id} = (I_1, \ldots, I_k)$ for $r = r' + t \in \mathbb{Z}_p$.

$\mathsf{Encrypt}(\mathsf{params}, \mathsf{id}, M; s)$

To encrypt a message $M \in \widetilde{\mathbb{G}}$ under the public key $\mathsf{id} = (I_1, \ldots, I_k) \in \left(\mathbb{Z}_p^*\right)^k$, pick a random $s \in \mathbb{Z}_p$ and output

$$\mathsf{CT} = \left(\Omega^s \cdot M,\ g^s,\ \left(h_1^{I_1} \ldots h_k^{I_k} \cdot g_3\right)^s\right) \in \widetilde{\mathbb{G}} \times \mathbb{G}^2. \tag{A.1}$$

$\mathsf{Decrypt}(d_{\mathsf{id}}, \mathsf{CT})$

Consider an identity $\mathsf{id} = (I_1, \ldots, I_k)$. To decrypt a given ciphertext $\mathsf{CT} = (A, B, C)$ using the private key $d_{\mathsf{id}} = (K_0, K_1, W_{k+1}, \ldots, W_\ell)$, output

$$A \cdot \frac{e(K_1, C)}{e(B, K_0)}.$$

For a valid ciphertext, we have

$$\frac{e(K_1, C)}{e(B, K_0)} = \frac{e\left(g^r, \left(h_1^{I_1} \ldots h_k^{I_k} \cdot g_3\right)^s\right)}{e\left(g^s, g_2^{\alpha}\left(h_1^{I_1} \ldots h_k^{I_k} \cdot g_3\right)^r\right)} = \frac{1}{e(g^s, g_2^{\alpha})} = \frac{1}{e(g_1, g_2)^s} = \frac{1}{\Omega^s}. \tag{A.2}$$

A.2 Two Propositions

Some analysis in our proof is based on the following propositions (*We do not claim the discovery of Proposition 3 or Proposition 4.*)

Proposition 3 *If event A implies event B, then* $\mathsf{Pr}[A] \leq \mathsf{Pr}[B]$.

Proof : Since $A \Rightarrow B$, we have $\mathsf{Pr}[\neg A \vee B] = 1$ and $\mathsf{Pr}[A \wedge \neg B] = 0$. Therefore,

$$\mathsf{Pr}[A] = \mathsf{Pr}[A \wedge \neg B] + \mathsf{Pr}[A \wedge B] = 0 + \mathsf{Pr}[A \wedge B] = \mathsf{Pr}[A|B]\mathsf{Pr}[B] \leq \mathsf{Pr}[B].$$

□

Proposition 4 *For any n events $A_1, \ldots, A_n$, it always holds that* $\mathsf{Pr}[\bigwedge_{1\leq i\leq n} A_i] \geq 1 - \sum_{i=1}^{n} \mathsf{Pr}[\neg A_i]$.

Proof :

$$\mathsf{Pr}[\bigwedge_{1\leq i\leq n} A_i] = 1 - \mathsf{Pr}[\bigvee_{1\leq i\leq n} \neg A_i] \geq 1 - \sum_{i=1}^{n} \mathsf{Pr}[\neg A_i].$$

□

A.3 Proof of Lemma 10.1

Lemma 10.1 *The functional encryption scheme* FE *described in Section 10.3 satisfies these properties:*

(a) *For any* $(pk, sk) \leftarrow f\mathsf{Setup}(1^\lambda, d, \mathcal{Z})$, *for any message* $\mathsf{Msg} \in \mathbb{Z}_p^*$, *for any point* $\vec{\boldsymbol{x}} \in [\mathcal{Z}]^d$, *for any rectangular range* $\mathbf{R} \subseteq [\mathcal{Z}]^d$, *if* $\mathsf{CT} \leftarrow f\mathsf{Enc}(\mathsf{Msg}, \vec{\boldsymbol{x}}, sk)$ *and* $\vec{\boldsymbol{\delta}} \leftarrow f\mathsf{KeyGen}(\mathbf{R}, \rho, sk)$, *then*

$$f\mathsf{Dec}(\mathsf{CT},\ \vec{\boldsymbol{x}},\ \mathbf{R},\ \vec{\boldsymbol{\delta}},\ pk) = \begin{cases} f_\rho(\mathsf{Msg}) & (\textit{if } \vec{\boldsymbol{x}} \in \mathbf{R}) \\ \perp & (\textit{otherwise}) \end{cases} \qquad \text{(A.3)}$$

(b) *For any* $(pk, sk) \leftarrow f\mathsf{Setup}(1^\lambda, d, \mathcal{Z})$*, for any message* $\mathsf{Msg} \in \mathbb{Z}_p^*$*, for any point* $\vec{\boldsymbol{x}} \in [\mathcal{Z}]^d$*, for any rectangular range* $\mathbf{R} \subseteq [\mathcal{Z}]^d$*, for any* $y \in \widetilde{\mathbb{G}}$*, if* $\mathsf{CT} \leftarrow f\mathsf{Enc}(\mathsf{Msg}, \vec{\boldsymbol{x}}, sk)$ *and* $\vec{\boldsymbol{\delta}} \leftarrow f\mathsf{KeyGen}(\mathbf{R}, \rho, sk)$*, then*

$$f\mathsf{Dec}(f\mathsf{Mult}(\mathsf{CT}, y, pk),\ \vec{\boldsymbol{x}},\ \mathbf{R},\ \vec{\boldsymbol{\delta}},\ pk) = \begin{cases} y \cdot f_\rho(\mathsf{Msg}) & (\textit{if } \vec{\boldsymbol{x}} \in \mathbf{R}) \\ \perp & (\textit{otherwise}) \end{cases} \tag{A.4}$$

Proof of Lemma 10.1: We observe that the BBG HIBE scheme [BBG05] satisfies the polymorphic property: An encryption of a message M can be viewed as the encryption of another message $\widehat{M}$ under different key. Precisely, let CT and $\widehat{\mathsf{CT}}$ be defined as follows, we have $\mathsf{CT} = \widehat{\mathsf{CT}}$:

$$\begin{aligned} \mathsf{CT} &= \mathsf{Encrypt}(\mathsf{params}, \mathsf{id}, M; s) = \left(\Omega^s \cdot M,\ g^s,\ \left(h_1^{I_1} \cdots h_k^{I_k} \cdot g_3\right)^s\ \right) \\ &\quad \text{under key: } \mathtt{params} = (g, g_1, g_2, g_3, h_1, \ldots, h_\ell, \Omega = e(g_1, g_2)),\ \ \mathtt{master\text{-}key} = g_2^\alpha \\ \widehat{\mathsf{CT}} &= \mathsf{Encrypt}(\widehat{\mathsf{params}}, \mathsf{id}, \widehat{M}; sz) = \left(\Omega^{sz} \cdot \widehat{M},\ \widehat{g}^{sz},\ \left(\widehat{h}_1^{I_1} \cdots \widehat{h}_k^{I_k} \cdot \widehat{g}_3\right)^{sz}\ \right), \\ &\quad \text{under key: } \widehat{\mathtt{params}} = (\widehat{g}, g_1, g_2, \widehat{g}_3, \widehat{h}_1, \ldots, \widehat{h}_\ell, \Omega = e(g_1, g_2)),\ \ \widehat{\mathtt{master\text{-}key}} = g_2^{\alpha z} \end{aligned} \tag{A.5}$$

where identity $\mathsf{id} = (I_1, \ldots, I_k) \in \left(\mathbb{Z}_p^*\right)^k$, $\widehat{M} = M\Omega^{s(1-z)}$, $\widehat{g} = g^{z^{-1} \bmod p}$, $\widehat{g}_3 = g_3^{z^{-1} \bmod p}$ and $\widehat{h}_i = h_i^{z^{-1} \bmod p}$ for $1 \leq i \leq \ell$. To be self-contained, the description of this BBG HIBE scheme [BBG05] is given in Appendix A.1. One can verify the above equality easily.

Proof of Lemma 1(a):

Let $(pk, sk) \leftarrow f\mathsf{Setup}(1^\lambda)$, message $\mathsf{Msg} \in \mathbb{Z}_p^*$, point $\vec{\boldsymbol{x}} \in [\mathcal{Z}]^d$, $\mathbf{R}$ be a d-dimensional rectangular range, and $\rho \in \mathbb{Z}_p^*$. Let $\mathsf{CT} \leftarrow f\mathsf{Enc}(\mathsf{Msg}, \vec{\boldsymbol{x}}, sk)$, $\vec{\boldsymbol{\delta}} \leftarrow f\mathsf{KeyGen}(\mathbf{R}, \rho, sk)$, and $y \in \widetilde{\mathbb{G}}$.

We consider dimension $j \in [d]$ and apply the polymorphic property of BBG scheme (equation (A.5)): Take $M = \sigma_j, s = s_j$ and $z = \rho\tau_j$. Then $\widehat{M} = M\Omega^{s(1-z)} = \sigma_j\Omega^{s_j(1-\tau_j\rho)}$.

In case $\vec{\boldsymbol{x}} \in \mathbf{R}$. If $\vec{\boldsymbol{x}} \in \mathbf{R}$, then the HIBE decryption will succeed in the process of $f\mathsf{Dec}$ (Section 10.3). Note that during decryption for dimension j, we use decryption key derived from $\mathsf{master\text{-}key}^{\rho\tau_j}$. Let $\widetilde{t_j}$ be as in Step 2(d) of $f\mathsf{Dec}$ for decrypting ciphertext CT. We have

$$\widetilde{t_j} = \widehat{M} = \sigma_j \Omega^{s_j(1-\tau_j\rho)}, j \in [d]. \tag{A.6}$$

Combining all d dimensions, and applying the two equalities (see algorithm $f\mathsf{Enc}$ in Section 10.3) $\mathsf{Msg} = -\sum_{j=1}^{d} s_j\tau_j \mod p$ and $\prod_{j=1}^{d}\sigma_j = \Omega^{-\sum_{j=1}^{d} s_j}$ we have,

$$\begin{aligned}
f\mathsf{Dec}(\mathsf{CT},\ \vec{\boldsymbol{x}},\ \mathbf{R},\ \vec{\boldsymbol{\delta}},\ pk) = \widetilde{t} = \prod_{j=1}^{d}\widetilde{t_j} &= \prod_{j=1}^{d}\left(\sigma_j\Omega^{s_j(1-\tau_j\rho)}\right)\\
&= \prod_{j=1}^{d}\sigma_j \ \cdot\ \prod_{j=1}^{d}\ \Omega^{s_j}\ \cdot\ \left(\prod_{j=1}^{d}\Omega^{-s_j\tau_j}\right)^{\rho}\\
&= \Omega^{-\sum_{j=1}^{d} s_j}\cdot\prod_{j=1}^{d}\ \Omega^{s_j}\cdot\left(\Omega^{\mathsf{Msg}}\right)^{\rho}\\
&= \Omega^{\rho\mathsf{Msg}}\\
&= f_\rho(\mathsf{Msg}).
\end{aligned}$$

In case $\vec{\boldsymbol{x}} \notin \mathbf{R}$. Let $\mathbf{R} = \mathbf{A}_1 \times \mathbf{A}_2 \ldots \times \mathbf{A}_d$ as in Step 1 of $f\mathsf{Dec}$. If $\vec{\boldsymbol{x}} \notin \mathbf{R}$, then for some dimension $j \in [d]$, $\vec{\boldsymbol{x}}[j] \notin \mathbf{A}_j$, and $f\mathsf{Dec}$ will output $\perp$ (Step 2 of $f\mathsf{Dec}$ in Section 10.3).

Proof of Lemma 1(b):

In case $\vec{\boldsymbol{x}} \in \mathbf{R}$. Check the decryption algorithm $\mathsf{Decrypt}$ of BBG HIBE (See Appendix A.1), it is easy to verify that: For any $\eta \in \widetilde{\mathbb{G}}$, if $\mathsf{Decrypt}(d_{\mathsf{id}}, (A, B, C))$ outputs M, then $\mathsf{Decrypt}(d_{\mathsf{id}}, (A\cdot\eta, B, C))$ will output $\eta \cdot M$.

Let $\eta_1, \ldots, \eta_d$ be as in Step 2 of $f\mathsf{Mult}$ in Section 10.3. Similar as the argument for Lemma 10.1(a), for dimension $j \in [d]$, we have ($\widetilde{t_j}$ is as in equation (A.6) and $\widetilde{t'_j}$

is the counterpart of $\widetilde{t_j}$ for decrypting ciphertext $f\mathsf{Mult}(\mathsf{CT}, y, pk)$)

$$\widetilde{t_j'} = \eta_j \widehat{M} = \eta_j \widetilde{t_j}.$$

Combining all d dimensions and applying the equation $\prod_{j=1}^{d} \eta_j = y$ (See Step 2 of $f\mathsf{Mult}$) and the result in Lemma 10.1(a), we have

$$\begin{aligned} f\mathsf{Dec}(f\mathsf{Mult}(\mathsf{CT}, y, pk),\ \vec{\boldsymbol{x}},\ \mathbf{R},\ \vec{\boldsymbol{\delta}},\ pk) &= \prod_{j=1}^{d} \widetilde{t_j'} = \prod_{j=1}^{d} \left(\eta_j \widetilde{t_j}\right) \\ &= \left(\prod_{j=1}^{d} \eta_j\right) \cdot f\mathsf{Dec}(\mathsf{CT},\ \vec{\boldsymbol{x}},\ \mathbf{R},\ \vec{\boldsymbol{\delta}},\ pk) \\ &= y \cdot f_\rho(\mathsf{Msg}). \end{aligned}$$

In case $\vec{\boldsymbol{x}} \notin \mathbf{R}$. The same argument for the case $\vec{\boldsymbol{x}} \notin \mathbf{R}$ of Lemma 10.1(a) applies. □

A.4 Proof of Theorem 10.2

Theorem 10.2 *Suppose there exists a weak-IND-sID-CPA adversary $\mathcal{A}_{\mathsf{FE}}$, which runs in time t_{FE} and has non-negligible advantage ϵ against the functional encryption scheme FE with one chosen delegation key query and N_{aq} chosen anonymous delegation key queries and N_{enc} chosen encryption queries. Then there exists an IND-sID-CPA adversary $\mathcal{A}_{\mathsf{BBG}}$, which has advantage $\frac{\epsilon}{2d}$ against the BBG HIBE scheme [BBG05] with $O(d\ell)$ chosen private key queries and zero chosen decryption query, and runs in time $t_{\mathsf{FE}} + O(d\ell \cdot t_{max} \cdot (N_{aq} + N_{enc}))$, where t_{max} is the maximum time for a random sampling (within a space of size at most p), a BBG encryption* Encrypt*, or a BBG key generation* KeyGen*.*

Proof :

The proof idea.

Let $\mathcal{A}_{\mathsf{FE}}$ be the $\mathsf{weak\text{-}IND\text{-}sID\text{-}CPA}$ adversary against the functional encryption scheme FE as in Theorem 10.2. We try to construct an $\mathsf{IND\text{-}sID\text{-}CPA}$ adversary $\mathcal{A}_{\mathsf{BBG}}$ against BBG based on $\mathcal{A}_{\mathsf{FE}}$: Choose two random messages m_0 and m_1, and send them to the BBG challenger. After receiving the challenge ciphertext CT for message m_b where $b \in \{0,1\}$, guess $b = 0$ and construct a FE challenge $(f_1(\mathsf{Msg}_0), f_1(\mathsf{Msg}_1), \mathsf{CT}_{\mathsf{FE}})$ based on the BBG challenge CT. If the adversary $\mathcal{A}_{\mathsf{FE}}$ wins the $\mathsf{weak\text{-}IND\text{-}sID\text{-}CPA}$ game, then output a guess $b' = 0$; otherwise output a guess $b' = 1$.

We argue that if indeed $b = 0$, then the forged FE challenge is valid, and the hypothesis is applicable: $\mathcal{A}_{\mathsf{FE}}$ wins with probability $1/2 + \epsilon$. If $b = 1$, the forged FE challenge is invalid, we cannot apply the hypothesis. However, in this case the forged FE challenge is independent on the value of b. Hence, in case of $b = 1$, $\mathcal{A}_{\mathsf{FE}}$ wins with probability exactly $1/2$.

Recall that the BBG HIBE scheme is $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ and the functional encryption scheme FE is $(f\mathsf{Setup}, f\mathsf{Enc}, f\mathsf{KeyGen}, f\mathsf{Dec}, f\mathsf{Mult})$. Now let us construct the $\mathsf{IND\text{-}sID\text{-}CPA}$ adversary $\mathcal{A}_{\mathsf{BBG}}$ against BBG. $\mathcal{A}_{\mathsf{BBG}}$ will simulate the $\mathsf{weak\text{-}IND\text{-}sID\text{-}CPA}$ game where $\mathcal{A}_{\mathsf{BBG}}$ takes the role of challenger and invokes $\mathcal{A}_{\mathsf{FE}}$ in the hypothesis as the adversary.

Construction of $\mathsf{IND\text{-}sID\text{-}CPA}$ adversary $\mathcal{A}_{\mathsf{BBG}}$ against BBG HIBE scheme based on $\mathcal{A}_{\mathsf{FE}}$

BBG Commit :

FE Commit : Adversary $\mathcal{A}_{\mathsf{FE}}$ chooses a random point $\vec{\boldsymbol{x}}^* = (x_1, \ldots, x_d) \in [\mathcal{Z}]^d$. $\mathcal{A}_{\mathsf{FE}}$ sends $\vec{\boldsymbol{x}}^*$ to FE challenger $\mathcal{A}_{\mathsf{BBG}}$ as the target identity.

BBG adversary $\mathcal{A}_{\mathsf{BBG}}$ chooses $\xi \in [d]$ at random and sends target identity $\mathsf{id}^* = \mathsf{ID}_\xi(x_\xi) \in \left(\mathbb{Z}_p^*\right)^\ell$ to BBG challenger $\mathcal{C}_{\mathsf{BBG}}$.

BBG Setup : BBG challenger $\mathcal{C}_{\mathsf{BBG}}$ runs setup algorithm Setup, and give $\mathcal{A}_{\mathsf{BBG}}$ the resulting system parameter params, keeping the $\mathsf{master\text{-}key}$ private.

BBG Phase 1 : Adversary $\mathcal{A}_{\mathsf{BBG}}$ does nothing.

BBG Challenge : Adversary $\mathcal{A}_{\mathsf{BBG}}$ chooses m_0, m_1 at random from the plaintext space $\widetilde{\mathbb{G}}$, and sends (m_0, m_1) to the challenger $\mathcal{C}_{\mathsf{BBG}}$. $\mathcal{C}_{\mathsf{BBG}}$ picks a random bit $b \in \{0,1\}$ and sends the challenge ciphertext $\mathsf{CT} = \mathsf{Encrypt}(\mathsf{params}, \mathsf{id}^*, m_b; s)$ to $\mathcal{A}_{\mathsf{BBG}}$.

BBG Phase 2 :

FE Setup : $\mathcal{A}_{\mathsf{BBG}}$ chooses d random elements $\tau_1, \ldots, \tau_d$ from $\mathbb{Z}_p^*$ and let $\vec{\boldsymbol{\tau}} = (\tau_1, \ldots, \tau_d)$. Let $(p, \mathbb{G}, \widetilde{\mathbb{G}}, e, \Omega)$ be a part of params, where p is a prime, both $\mathbb{G}$ and $\widetilde{\mathbb{G}}$ are cyclic multiplicative group of order p, $e : \mathbb{G} \times \mathbb{G} \to \widetilde{\mathbb{G}}$ is a bilinear map, and $\Omega \in \widetilde{\mathbb{G}}$. Let $pk = (p, \mathbb{G}, \widetilde{\mathbb{G}}, e, \Omega)$ and $sk = (pk, \mathsf{params}, \mathsf{master\text{-}key}, \vec{\boldsymbol{\tau}})$. $\mathcal{A}_{\mathsf{BBG}}$ sends pk to $\mathcal{A}_{\mathsf{FE}}$.

Note: $\mathcal{A}_{\mathsf{BBG}}$ *does not know* $\mathsf{master\text{-}key}$.

FE Challenge : The FE challenger $\mathcal{A}_{\mathsf{BBG}}$ chooses a random bit $a \in \{0,1\}$ and a random message Msg_{1-a} from the message space $\mathbb{Z}_p^*$. $\mathcal{A}_{\mathsf{BBG}}$ will decide Msg_a and generate the challenge ciphertext $\mathsf{CT}_{\mathsf{FE}}$ in this way:

1. Parse the BBG challenge ciphertext as $\mathsf{CT} = (A, B, C)$, where $A = \Omega^s m_b$.
2. Choose $(d-1)$ random elements $s_1, \ldots, s_{\xi-1}, s_{\xi+1}, \ldots, s_d$ (i.e. excluding s_ξ) from $\mathbb{Z}_p^*$.
3. Choose d random elements $\sigma_1, \ldots, \sigma_d$ from $\widetilde{\mathbb{G}}$ with constraint

$$\prod_{j=1}^{d} \sigma_j = (\Omega^s m_b)^{-1} m_0 \cdot \Omega^{-\sum_{\substack{1 \le j \le d \\ j \ne \xi}} s_j}$$

4. For each $j \in [d]$ and $j \neq \xi$, encrypt σ_j under identity $\mathsf{ID}_j(x_j)$ with random coin s_j to obtain ciphertext $\vec{\boldsymbol{c}}_j$ as follows

$$\vec{\boldsymbol{c}}_j \leftarrow \mathsf{Encrypt}(\mathtt{params},\ \mathsf{ID}_j(x_j),\ \sigma_j;\ s_j). \tag{A.7}$$

5. Define $\vec{\boldsymbol{c}}_\xi$ based on the BBG challenge ciphertext $\mathsf{CT} = (\Omega^s m_b, B, C)$:

$$\vec{\boldsymbol{c}}_\xi = (\Omega^s m_b \cdot m_0^{-1} \cdot \sigma_\xi,\ B,\ C).$$

6. Define $\mathsf{Msg}_a = -\sum_{j\in[d]} s_j \cdot \tau_j \pmod{p}$, where unknown $s_\xi \in \mathbb{Z}_p$ is defined by $\Omega^{s_\xi} = \Omega^s m_b \cdot m_0^{-1}$. Although the value Msg_a is unknown since s_ξ is unknown, $\mathcal{A}_{\mathsf{BBG}}$ can still compute $f_1(\mathsf{Msg}_a)$:

$$f_1(\mathsf{Msg}_a) = \Omega^{\mathsf{Msg}_a} = \Big((\Omega^s m_b)^{-1} \cdot m_0\Big)^{\tau_\xi} \cdot \Omega^{-\sum_{\substack{1\le j\le d\\ j\neq\xi}} s_j\cdot\tau_j}$$

$\mathcal{A}_{\mathsf{BBG}}$ computes $f_1(\mathsf{Msg}_{1-a}) = \Omega^{\mathsf{Msg}_{1-a}}$.

7. Set the challenge ciphertext to $\mathsf{CT}_{\mathsf{FE}} = (\vec{\boldsymbol{c}}_1, \ldots, \vec{\boldsymbol{c}}_d)$, and send $(\mathsf{CT}_{\mathsf{FE}}, f_1(\mathsf{Msg}_0), f_1(\mathsf{Msg}_1))$ to $\mathcal{A}_{\mathsf{FE}}$.

FE Learning Phase :

1. $\mathcal{A}_{\mathsf{FE}}$ issues a *delegation key query* $(\mathbf{R}, \rho)$, where $\vec{\boldsymbol{x}}^* \notin \mathbf{R}$ and $\mathbf{R} = \mathbf{A}_1 \times \mathbf{A}_2 \ldots \times \mathbf{A}_d \subseteq [\mathcal{Z}]^d$: If $x_\xi \in \mathbf{A}_\xi$, then $\mathcal{A}_{\mathsf{BBG}}$ aborts and outputs a random bit $b' \in \{0,1\}$ (Denote this event as $\mathbf{E}_1$). Otherwise, simulate the procedure of $f\mathsf{KeyGen}$:
   (a) The private key is $sk = (pk, \mathsf{params}, \mathsf{master\text{-}key}, \vec{\boldsymbol{\tau}} = (\tau_1, \ldots, \tau_d))$, where $\mathcal{A}_{\mathsf{BBG}}$ has only pk, params and $\vec{\boldsymbol{\tau}}$, and does not know $\mathsf{master\text{-}key}$ (which is kept securely by the BBG challenger $\mathcal{C}_{\mathsf{BBG}}$).
   (b) For each $j \in [d]$, generate a set δ_j in this way:
   - For each identity $\mathsf{id} \in \mathsf{IdSet}_j(\mathbf{A}_j)$, issue a private key query with identity id to BBG challenger $\mathcal{C}_{\mathsf{BBG}}$ and get reply d_{id}.
     Note: The BBG private key query (id*) is valid, i.e.* $\mathsf{id} \neq \mathsf{id}^*$ *and* id *is not a prefix of* id^**. This is implied by the following two properties satisfied by our constructions of* ID_ι *and* IdSet_ι *in Section 10.2: (1) For any* $i, j \in [d], x, y \in [\mathcal{Z}]$*, if* $\mathsf{ID}_i(x)$ *and* $\mathsf{ID}_j(y)$ *share a non-empty prefix, then* $i = j$*; (2) For any* $\in [a,b] \subseteq [\mathcal{Z}]$*, iff* $x \in [a,b]$*, there exits an identity* id *in the set* $\mathsf{IdSet}_j([a,b])$*, such that* id *is a prefix of identity* $\mathsf{ID}_j(x)$*.*
   - For each identity $\mathsf{id} \in \mathsf{IdSet}_j(\mathbf{A}_j)$, parse the key d_{id} as $(K_0, K_1, \Upsilon_k, \ldots, \Upsilon_\ell)$ and set $d'_{\mathsf{id}} = (K_0^{\rho\tau_j}, K_1^{\rho\tau_j}, \Upsilon_k^{\rho\tau_j}, \ldots, \Upsilon_\ell^{\rho\tau_j})$.
   - Set $\delta_j \leftarrow \{d'_{\mathsf{id}} : \mathsf{id} \in \mathsf{IdSet}_j(\mathbf{A}_j)\}$.

   (c) Send $\vec{\boldsymbol{\delta}} = (\delta_1, \delta_2, \ldots, \delta_d)$ to $\mathcal{A}_{\mathsf{FE}}$ as the delegation key w.r.t. $(\mathbf{R}, \rho)$.

   Note: $\mathcal{A}_{\mathsf{FE}}$ *can make at most one delegation key query.*

2. $\mathcal{A}_{\mathsf{FE}}$ issues an *anonymous delegation key query* ($\mathbf{R}$): Choose a random element $Z \in \widetilde{\mathbb{G}}$. For each anonymous delegation key query ($\mathbf{R}$), choose $\rho \in \mathbb{Z}_p^*$ at random, run the algorithm $f\mathsf{KeyGen}(\mathbf{R}, \rho, sk')$, where $sk' = (pk, \mathsf{params}, Z, \vec{\boldsymbol{\tau}})$ (i.e. taking Z as the master key), and get output $\vec{\boldsymbol{\delta}}$. Send $\vec{\boldsymbol{\delta}}$ to $\mathcal{A}_{\mathsf{FE}}$ as the delegation key w.r.t. $\mathbf{R}$.

   Note: (1) $\mathcal{A}_{\mathsf{BBG}}$ can answer anonymous delegation key query without *the help of BBG challenger $\mathcal{C}_{\mathsf{BBG}}$. (2) There exists an unknown ω, such that $Z = \mathsf{master\text{-}key}^{\omega}$. The generated delegation key $\vec{\boldsymbol{\delta}}$ corresponds to range $\mathbf{R}$ and (unknown) function key $\rho\omega$, where $\rho\omega$ is uniformly distributed in $\mathbb{Z}_p^*$ as desired.*

3. $\mathcal{A}_{\mathsf{FE}}$ issues an encryption key query ($\mathsf{Msg}, \vec{\boldsymbol{x}}$): Run the encryption algorithm: $\mathsf{C} \leftarrow f\mathsf{Enc}(\mathsf{Msg}, \vec{\boldsymbol{x}}, sk)$ and send the resulting ciphertext C to $\mathcal{A}_{\mathsf{FE}}$ as the reply.

   Note: $\mathcal{A}_{\mathsf{BBG}}$ can run algorithm $f\mathsf{Enc}$, since it requires only $pk, \mathsf{params}, \vec{\boldsymbol{\tau}}$.

FE Guess : Adversary $\mathcal{A}_{\mathsf{FE}}$ outputs a bit $a' \in \{0, 1\}$.

BBG Guess : If $a = a'$, adversary $\mathcal{A}_{\mathsf{BBG}}$ outputs $b' = 0$. Otherwise, $\mathcal{A}_{\mathsf{BBG}}$ outputs $b' = 1$.

The constructed BBG adversary $\mathcal{A}_{\mathsf{BBG}}$ made $O(d\ell)$ private key query and zero decryption query to the BBG challenger $\mathcal{C}_{\mathsf{BBG}}$. Let $\mathsf{id}^* = \mathsf{ID}_\xi(x_\xi) = (I_1, \ldots, I_\ell) \in \left(\mathbb{Z}_p^*\right)^\ell$. Recall that the two BBG ciphertexts CT and $\vec{\boldsymbol{c}}_\xi$ are

$$\mathsf{CT} = (A,\ B,\ C) = \left(\Omega^s \cdot m_b,\ g^s,\ \left(h_1^{I_1} \ldots h_\ell^{I_\ell} \cdot g_3\right)^s\right) \in \widetilde{\mathbb{G}} \times \mathbb{G}^2$$
$$\vec{\boldsymbol{c}}_\xi = (\Omega^{s_\xi} \cdot \sigma_\xi,\ B,\ C) = \left(\Omega^{s_\xi} \cdot \sigma_\xi,\ g^s,\ \left(h_1^{I_1} \ldots h_\ell^{I_\ell} \cdot g_3\right)^s\right) \in \widetilde{\mathbb{G}} \times \mathbb{G}^2$$

where $\Omega^{s_\xi} = \Omega^s m_b \cdot m_0^{-1}$. If $b = 0$, then $s_\xi = s$ and $\vec{\boldsymbol{c}}_\xi$ is a valid BBG encryption of σ_ξ under identity $\mathsf{ID}_\xi(x_\xi)$ with random coin s_ξ. Consequently, the FE scheme simulated by $\mathcal{A}_{\mathsf{BBG}}$ is *identical* to a real one from the view of $\mathcal{A}_{\mathsf{FE}}$ (even if $\mathcal{A}_{\mathsf{FE}}$ is computationally unbounded). If $b = 1$, then s_ξ is independent on s. As a result, in the FE scheme simulated by $\mathcal{A}_{\mathsf{BBG}}$, the challenging ciphertext $\mathsf{CT}_{\mathsf{FE}}$ is *independent* on the value of Msg_a. Note that adversary $\mathcal{A}_{\mathsf{FE}}$ does not know $m_0, m_1, \mathsf{params}$.

Let the d-dimensional range $\mathbf{R} = \mathbf{A}_1 \times \ldots \times \mathbf{A}_d$. Define set $S_{\#}$:

$$S_{\#} = \{j \in [d] : \vec{\boldsymbol{x}}^*[j] \notin \mathbf{A}_j\}$$

Since $\vec{\boldsymbol{x}}^* \notin \mathbf{R}$, $S_{\#}$ is not empty and $|S_{\#}| \geq 1$. We have

$$\mathsf{Pr}[\neg\mathbf{E}_1] = \mathsf{Pr}[\xi \in S_{\#}] = \frac{|S_{\#}|}{d} \geq \frac{1}{d}.$$

Note that adversary $\mathcal{A}_{\mathsf{BBG}}$ has two terminal cases: (1) If event $\mathbf{E}_1$ occurs, $\mathcal{A}_{\mathsf{BBG}}$ outputs a random bit $b' \in \{0,1\}$. (2) If event $\mathbf{E}_1$ does not occur, $\mathcal{A}_{\mathsf{BBG}}$ outputs $b' = 0$ iff $\mathcal{A}_{\mathsf{FE}}$ outputs $a' = a$.

In case of $\mathbf{E}_1$: Conditional on event $\mathbf{E}_1$, $\mathsf{Pr}[b = b'] = 1/2$.

In case of $\neg\mathbf{E}_1$: Suppose event $\mathbf{E}_1$ does not occur. Then $b' = 0 \Leftrightarrow a = a'$ and $b' = 1 \Leftrightarrow a \neq a'$. Applying the Proposition 3, we have $\mathsf{Pr}[b' = 0|b = 0] = \mathsf{Pr}[a = a'|b = 0]$ and $\mathsf{Pr}[b' = 1|b = 1] = \mathsf{Pr}[a \neq a'|b = 1]$.

As a result, conditional on event $\neg\mathbf{E}_1$,

$$\begin{aligned}
\mathsf{Pr}[b = b'] &= \mathsf{Pr}[b = b' = 0] + \mathsf{Pr}[b = b' = 1] \\
&= \mathsf{Pr}[b = 0]\mathsf{Pr}[b' = 0|b = 0] + \mathsf{Pr}[b = 1]\mathsf{Pr}[b' = 1|b = 1] \\
&= \mathsf{Pr}[b = 0]\mathsf{Pr}[a = a'|b = 0] + \mathsf{Pr}[b = 1]\mathsf{Pr}[a \neq a'|b = 1] \\
&= \frac{1}{2} \times \left(\frac{1}{2} + \epsilon\right) + \frac{1}{2} \times \frac{1}{2} \\
&= \frac{1}{2} + \frac{1}{2}\epsilon
\end{aligned}$$

Combining the two cases ($\mathbf{E}_1$ and $\neg\mathbf{E}_1$), we obtain the advantage of $\mathcal{A}_{\mathsf{BBG}}$ against BBG scheme in the $\mathsf{IND\text{-}sID\text{-}CPA}$ game:

$$\mathsf{Adv}_{\mathsf{BBG}}^{\mathcal{A}_{\mathsf{BBG}}} + \frac{1}{2} = \mathsf{Pr}[\mathbf{E}_1] \times \frac{1}{2} + \mathsf{Pr}[\neg\mathbf{E}_1] \times \left(\frac{1}{2} + \frac{1}{2}\epsilon\right) = \frac{1}{2} + \frac{\epsilon}{2}\mathsf{Pr}[\neg\mathbf{E}_1] \geq \frac{1}{2} + \frac{\epsilon}{2d}$$

The adversary $\mathcal{A}_{\mathsf{BBG}}$ wins the game with probability at least $1/2 + \epsilon/(2d)$ using $O(d\ell)$

private key queries and running in time $t_{\mathsf{FE}} + O(d\ell \cdot t_{max} \cdot (N_{aq} + N_{enc}))$, where t_{max} is the maximum time for random sampling (within a space of size at most p), BBG encryption $\mathsf{Encrypt}$, or BBG key generation KeyGen, and N_{aq} (N_{enc}, respectively) is the number of anonymous delegation key queries (encryption key queries, respectively) made by $\mathcal{A}_{\mathsf{FE}}$. □

A.5 A valid proof should be generated from points within dataset D

The notion that a valid proof is essentially generated from points (and their tags) within the dataset $\mathbf{D}$, is formulized by Lemma A.2. We prove Lemma A.2 in two steps: first we show that the **GKEA** Assumption 5 implies Lemma A.1 (which states that an alternative form of **GKEA** problem is hard); then we derive Lemma A.2 from Lemma A.1.

We remark that both the proof of Lemma A.1 in Appendix A.5.1 and proof of Lemma A.2 in Appendix A.5.2 follow the proof framework for statement that **KEA3** implies **KEA** in [BP04a]. Their proof can be outlined as follows: Given any adversary algorithm $\mathcal{A}_{\mathbf{KEA}}$, construct an adversary algorithm $\mathcal{A}_{\mathbf{KEA3}}$. Then applying **KEA3** Assumption, there exists an extractor algorithm $\bar{\mathcal{A}}_{\mathbf{KEA3}}$. Based on $\bar{\mathcal{A}}_{\mathbf{KEA3}}$, construct extractor algorithm $\bar{\mathcal{A}}_{\mathbf{KEA}}$ for $\mathcal{A}_{\mathbf{KEA}}$. The key point is how to convert the input/output between $\bar{\mathcal{A}}_{\mathbf{KEA}}$ ($\mathcal{A}_{\mathbf{KEA}}$, respectively) and $\bar{\mathcal{A}}_{\mathbf{KEA3}}$ ($\mathcal{A}_{\mathbf{KEA3}}$, respectively).

A.5.1 Lemma A.1 and Proof

Lemma A.1 *Suppose Assumption 5 holds. For any PPT algorithm $\mathcal{A}$, there exists a PPT algorithm $\bar{\mathcal{A}}$, such that*

$$\mathsf{Adv}^{\textbf{Lem A.1}}_{\mathcal{A},\bar{\mathcal{A}}}(\lambda) \overset{\text{def}}{=} \Pr\left[\begin{array}{l} S_{m+1} \leftarrow \{(\theta v_i, v_i^{\beta}) : i \in [m+1], v_i \overset{\$}{\leftarrow} \widetilde{\mathbb{G}}, \theta \overset{\$}{\leftarrow} \widetilde{\mathbb{G}}, \beta \overset{\$}{\leftarrow} \mathbb{Z}_p^*\} \\ (\Psi_1, \Psi_2, X) \leftarrow \mathcal{A}(S_{m+1}; r); \\ (\Psi_1, \Psi_2, X, \mu_1, \mu_2, \ldots, \mu_m, \mu_{m+1}) \leftarrow \bar{\mathcal{A}}(S_{m+1}; r, \bar{r}) : \\ \qquad \Psi_2 = \left(\frac{\Psi_1}{\theta^X}\right)^{\beta} \ \wedge \ \Psi_2 \neq \prod_{j=1}^{m+1} (v_j^{\beta})^{\mu_j} \end{array}\right]$$
$$\leq \nu_2(\lambda), \tag{A.8}$$

where the probability is taken over all random coins used, m is polynomial in λ and function $\nu_2(\cdot)$ is as in Assumption 5.

Proof of Lemma A.1: Let adversary $\mathcal{A}$ be as in Lemma A.1. We construct a **GKEA** adversary $\mathcal{A}_1$ based on an adversary $\mathcal{A}$.

Construction of **GKEA** adversary $\mathcal{A}_1$: Based on an adversary $\mathcal{A}$

1. The input is $W_m = \{(u_i, u_i^{\beta}) \in \widetilde{\mathbb{G}}^2 : 1 \leq i \leq m\}$ and the random coin is r_1.
2. Choose two independent random elements $R_1, R_2 \in \widetilde{\mathbb{G}}^2$ based on the random coin r_1. There exists some unknown $\theta, v_{m+1} \in \widetilde{\mathbb{G}}$, such that $R_1 = \theta v_{m+1}$ and $R_2 = v_{m+1}^{\beta}$.
3. For each $1 \leq i \leq m$, define $v_i = u_i v_{m+1}$ and compute $\theta v_i = u_i(\theta v_{m+1}) = u_i R_1$ and $v_i^{\beta} = (u_i v_{m+1})^{\beta} = u_i^{\beta} R_2$. Let $S_{m+1} = \{(\theta v_i, v_i^{\beta}) : 1 \leq i \leq m+1\}$.
4. Invoke the adversary $\mathcal{A}$ with random coin r derived from r_1: $(\Psi_1, \Psi_2, X) \leftarrow \mathcal{A}(S_{m+1}; r)$.
5. Output $(U_1 = \frac{\Psi_1}{R_1^X}, U_2 = \frac{\Psi_2}{R_2^X})$.

Since $\left(\frac{\Psi_1}{R_1^X}\right)^{\beta} = \frac{\Psi_1^{\beta}}{\theta^{X\beta} v_{m+1}^{X\beta}}$ and $\frac{\Psi_2}{R_2^X} = \frac{\Psi_2}{v_{m+1}^{X\beta}}$, we have

$$\left(\frac{\Psi_1}{\theta^X}\right)^{\beta} = \Psi_2 \qquad \Leftrightarrow \qquad \left(\frac{\Psi_1}{R_1^X}\right)^{\beta} = \frac{\Psi_2}{R_2^X}. \tag{A.9}$$

According to Assumption 5, there exists an extractor $\bar{\mathcal{A}}_1$ for the adversary $\mathcal{A}_1$, such that $\mathsf{Adv}^{\mathbf{GKEA}}_{\mathcal{A}_1,\bar{\mathcal{A}}_1}$ is negligible. Now we construct an extractor $\bar{\mathcal{A}}$ for $\mathcal{A}$ based on $\bar{\mathcal{A}}_1$.

Construction of extractor $\bar{\mathcal{A}}$ for $\mathcal{A}$: Based on **GKEA** extractor $\bar{\mathcal{A}}_1$

1. The input is $S_{m+1} = \{(\theta v_i, v_i^\beta) : 1 \le i \le m+1\}$. The random coin is $(r, \bar{r})$.
2. For each $1 \le i \le m$, set $u_i = \frac{\theta v_i}{\theta v_{m+1}}$ and compute $u_i^\beta = \frac{v_i^\beta}{v_{m+1}^\beta}$. Let $W_m = \{(u_i, u_i^\beta) : 1 \le i \le m\}$.
3. Invoke adversary $\mathcal{A}_1$ with random coin $r_1 = (\theta v_{m+1}, v_{m+1}^\beta, r)$: $(U_1 = \frac{\Psi_1}{R_1^X}, U_2 = \frac{\Psi_2}{R_2^X}) \leftarrow \mathcal{A}_1(W_m; r_1)$.
   Note: We can represent the random coin r_1 used by $\mathcal{A}_1$ as (R_1, R_2, r).
4. Invoke extractor $\bar{\mathcal{A}}_1$ with random coin $(r_1, \bar{r}_1 = \bar{r})$: $(\mu_1, \ldots, \mu_m) \leftarrow \bar{\mathcal{A}}_1(W_m; r_1, \bar{r}_1)$.
5. Define $\mu_{m+1} = X - \sum_{i=1}^m \mu_i$. Output $(\mu_1, \ldots, \mu_m, \mu_{m+1})$.

If $U_2 = \prod_{i=1}^m u_i^{\beta\mu_i}$, then we have

$$\prod_{i=1}^{m+1} v_i^{\beta\mu_i} = v_{m+1}^{\beta\mu_{m+1}} \prod_{i=1}^{m} v_i^{\beta\mu_i} = v_{m+1}^{\beta(X-\sum_{i=1}^m \mu_i)} \prod_{i=1}^{m} u_i^{\beta\mu_i} \prod_{i=1}^{m} v_{m+1}^{\beta\mu_i} = v_{m+1}^{\beta X} U_2 = R_2^X U_2 = \Psi_2.$$

That is,

$$U_2 = \prod_{i=1}^{m} u_i^{\beta\mu_i} \qquad \Rightarrow \qquad \prod_{i=1}^{m+1} v_i^{\beta\mu_i} = \Psi_2. \tag{A.10}$$

If $U_1^\beta = U_2 \;\Rightarrow\; U_2 = \prod_{i=1}^m u_i^{\beta\mu_i}$, combining with equation (A.9) and equation (A.10), we have

$$\left(\frac{\Psi_1}{\theta^X}\right)^\beta = \Psi_2 \Rightarrow U_1^\beta = U_2 \Rightarrow U_2 = \prod_{i=1}^{m} u_i^{\beta\mu_i} \Rightarrow \Psi_2 = \prod_{i=1}^{m+1} v_i^{\beta\mu_i}.$$

As a result,

$$\left(U_1^\beta = U_2 \Rightarrow U_2 = \prod_{i=1}^{m} u_i^{\beta\mu_i}\right) \Rightarrow \left(\left(\frac{\Psi_1}{\theta^X}\right)^\beta = \Psi_2 \Rightarrow \Psi_2 = \prod_{i=1}^{m+1} v_i^{\beta\mu_i}\right)$$

Note that the implications in equation (A.9) and equation (A.10) are *always* true, while the implication that $U_1^\beta = U_2 \Rightarrow U_2 = \prod_{i=1}^{m} u_i^{\beta\mu_i}$ is true only with certain probability.

Applying the Proposition 3 in Appendix A.2, we have

$$\begin{aligned}\mathsf{Pr}\left[U_1^\beta = U_2 \Rightarrow U_2 = \prod_{i=1}^{m} u_i^{\beta\mu_i}\right] &= 1 - \mathsf{Adv}_{\mathcal{A}_1,\bar{\mathcal{A}}_1}^{\mathbf{GKEA}} \\ &\leq \mathsf{Pr}\left[\left(\frac{\Psi_1}{\theta^X}\right)^\beta = \Psi_2 \Rightarrow \Psi_2 = \prod_{i=1}^{m+1} v_i^{\beta\mu_i}\right] \\ &= 1 - \mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\mathbf{Lem\ A.1}}.\end{aligned}$$

Hence,

$$\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\mathbf{Lem\ A.1}} \leq \mathsf{Adv}_{\mathcal{A}_1,\bar{\mathcal{A}}_1}^{\mathbf{GKEA}} \leq \nu_2.$$

□

A.5.2 Lemma A.2 and Proof

Lemma A.2 *Suppose Assumption 5 holds. For any PPT algorithm $\mathcal{A}$, there exists a PPT algorithm $\bar{\mathcal{A}}$, such that $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\mathbf{Lem\ A.2}}(1^\lambda) \leq \nu_2(\lambda)$, where the advantage $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\mathbf{Lem\ A.2}}$ of $\mathcal{A}$ against $\bar{\mathcal{A}}$ w.r.t. scheme $\widetilde{\mathcal{E}} = (\mathsf{KGen}, \mathsf{DEnc}, \mathsf{CollRes})$ is defined as*

$$\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\mathbf{Lem\ A.2}}(1^\lambda) \overset{\text{def}}{=} 1 - \mathsf{Pr}\left[\begin{array}{c} (\zeta, X, \Psi_2, \mathsf{view}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}, \mathbf{D}, \mathbf{R}) \leftarrow \mathsf{Exp}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}(1^\lambda); \\ \{\mu_i : i \in [N]\} \leftarrow \bar{\mathcal{A}}(\mathsf{view}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}) : \\ \zeta = \mathtt{accept} \ \Rightarrow \ \Psi_2 = \prod_{i\in[N]} \left(v_i^\beta\right)^{\mu_i} \end{array}\right],$$

where v_i^β is the second component of tag $\vec{\boldsymbol{t}}_i$ for data point $\vec{\boldsymbol{x}}_i \in \mathbf{D}$ (See Step 2 of DEnc *in Section 11.1).*

Proof of Lemma A.2: Let $\mathcal{A}$ be any PPT adversary against scheme $\widetilde{\mathcal{E}} = (\mathsf{KGen}, \mathsf{DEnc}, \mathsf{CollRes})$. We construct a PPT adversary $\mathcal{B}$ against Lemma A.1 based on $\mathcal{A}$.

Adversary $\mathcal{B}$ against Lemma A.1: Based on $\mathcal{A}$

1. The input is $S_m = \{(\theta v_j, v_j^\beta) \in \widetilde{\mathbb{G}}^2 : j \in [m]\}$. The random coin used in this algorithm is r.
2. Simulate Alice in the experiment $\mathsf{Exp}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$.
   (a) Invoke adversary $\mathcal{A}$ with a random coin derived from r. $\mathcal{A}$ chooses a set $\mathbf{D} = \{\vec{\boldsymbol{x}}_i \in [\mathcal{Z}]^d : i \in [N]\}$ of $N = m$ d-dimensional data points. *Note: If $N > m$, $\mathcal{B}$ can generate more tuples $(\theta v_j, v_j^\beta)$ for $j = m+1, \ldots, N$ from S_m. For simplicity, we just assume $N = m$.*
   (b) Simulate KGen:
      i. Invoke $f\mathsf{Setup}(1^\lambda)$ to obtain public/private key pair (pk, sk).
      ii. Choose γ' at random from $\mathbb{Z}_p^*$. $\mathcal{B}$ does not know the values of θ, β.
   (c) Simulate DEnc:
      i. Choose N random elements $W_1, \ldots, W_N$ from $\mathbb{Z}_p^*$.
      ii. For each $i \in [N]$, compute a tag $\vec{\boldsymbol{t}}_i = \Big(\theta v_i, v_i^\beta, f_1(W_i)\Big)$.
      iii. For each $i \in [N]$, encrypt message W_i under point $\vec{\boldsymbol{x}}_i$: $\mathsf{CT}'_i \leftarrow f\mathsf{Enc}(W_i, \vec{\boldsymbol{x}}_i, sk)$; attach $v_i^{\beta\gamma'}$ to ciphertext by applying the homomorphic property of the functional encryption scheme: $\mathsf{CT}_i \leftarrow f\mathsf{Mult}(\mathsf{CT}'_i, v_i^{\beta\gamma'}, pk)$.
      iv. Send $\mathbf{D}_{\mathsf{B}} = \{\mathbf{D}, \mathbf{T} = \{\vec{\boldsymbol{t}}_i : i \in [N]\}, \mathbf{C} = \{\mathsf{CT}_i : i \in [N]\}, pk\}$ to adversary $\mathcal{A}$.
   (d) Simulate $\mathsf{CollRes}$: For each query range $\mathbf{R}_j$ chosen by $\mathcal{A}$ during $\mathcal{A}$'s learning phase and challenging phase
      i. (Step A1) Choose random nonce $\rho \in \mathbb{Z}_p^*$, generate delegation key $\vec{\boldsymbol{\delta}} \leftarrow f\mathsf{KeyGen}(\mathbf{R}_j, \rho, sk)$, and send $(\mathbf{R}_j, \vec{\boldsymbol{\delta}})$ to adversary $\mathcal{A}$.
      ii. (Step A2) Do not perform verifications, after receiving reply from $\mathcal{A}$.
3. Receive output $(X, \Psi_1, \Psi_2, \Psi_3, \Psi_4)$ for the challenging query range $\mathbf{R}$ from $\mathcal{A}$. Output (X, Ψ_1, Ψ_2).

Remarks on Algorithm $\mathcal{B}$.

1. Adversary $\mathcal{B}$ has the private key sk, and can generate the challenge-message in the same way as in $\mathsf{CollRes}$ in Section 11.1.

2. Adversary $\mathcal{B}$ has no knowledge of β or θ, and consequently cannot perform verification as in $\mathsf{CollRes}$.

3. The view of adversary $\mathcal{A}$ after interacting with the simulated scheme is identically distributed with the the view $\mathsf{view}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$ of $\mathcal{A}$ after interacting with the real scheme in the experiment $\mathsf{Exp}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$.

   (a) KGen: The simulator $\mathcal{B}$ generates key (pk, sk) in the same way as the real scheme. The unknown secret key $\beta \in \mathbb{Z}_p^*, \theta \in \widetilde{\mathbb{G}}$ and $\gamma = (\beta\gamma')^{-1} \in \mathbb{Z}_p^*$ are independently and uniformly randomly distributed over corresponding domains.

   (b) DEnc: The dataset $\mathbf{D}$ is generated in the same way as in real experiment. The tags $\mathbf{T}$ are identically distributed as in real experiment. The ciphertexts $\mathbf{C}$ are generated in the same way as in real experiment. As a result, $\mathbf{D}_{\mathsf{B}} = \{\mathbf{D}, \mathbf{T}, \mathbf{C}\}$ that $\mathcal{A}$ received is identically distributed as in real experiment.

   (c) $\mathsf{CollRes}$

      i. Step A1: The simulator just follows the procedure in Section 11.1 with key sk and random nonce ρ to execute this step.

      ii. Step A2: Since the simulator does not know the values of secret key (β, γ, θ), it cannot perform the verifications. However, according to our scheme, the accept/reject decisions are always kept secret from Bob (or adversary).

4. If $\mathcal{A}$ is a successful adversary, then its output will indeed pass all verifications with non-negligible probability, although the simulator cannot perform the actual verifications and yet cannot know whether $\mathcal{A}$ succeeds or not in each single attack instance.

From the random coin r, $\mathcal{B}$ can simulate the experiment $\mathsf{Exp}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$ and produce a view view_r which is identically distributed as the view $\mathsf{view}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$ produced by a real experiment. In the other direction, information theoretically, the random coin r can be recovered from the adversary's view view_r, considering r as the collection of all (true) random bits flipped in the simulation. Consequently, we can view view_r as an alternative representation of random coin r.

By Lemma A.1, there exists a PPT algorithm $\bar{\mathcal{B}}$ such that $\mathsf{Adv}_{\mathcal{B},\bar{\mathcal{B}}}^{\mathbf{Lem\ A.1}} \leq \nu_2$. We construct an extractor $\bar{\mathcal{A}}$ for adversary $\mathcal{A}$ based on $\bar{\mathcal{B}}$.

Extractor $\bar{\mathcal{A}}$: Based on $\bar{\mathcal{B}}$

1. The input is $\mathsf{view}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$, a state variable describing all random coins chosen and all message accessed by $\mathcal{A}$ during interactions with $\widetilde{\mathcal{E}}$ in the experiment $\mathsf{Exp}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$.
2. Recover $\mathbf{D}, \mathbf{T}, \mathbf{C}$ from $\mathsf{view}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$ and construct a set $S_N \leftarrow \{(\theta v_i, v_i^{\beta}) : i \in [N]\}$, where θv_i and v_i^{β} are the first two components of tag $\vec{\boldsymbol{t}}_i \in \mathbf{T}$.
3. Invoke $\mathcal{B}$ on input S_N with random coin $\mathsf{view}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$. $\mathcal{B}$ extracts information from $\mathsf{view}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$ and replays[1] the interaction between Alice and Bob (i.e. the adversary $\mathcal{A}$) in the experiment $\mathsf{Exp}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$. Denote the output of experiment as $(\zeta, X, \vec{\boldsymbol{\Psi}}, \mathsf{view}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}, \mathbf{D}, \mathbf{R})$. Recover the reply of $\mathcal{A}$ on the challenging query $\mathbf{R}$ from $\mathsf{view}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$, and denote it with $(X, \Psi_1, \Psi_2, \Psi_3, \Psi_4)$. $\mathcal{B}$ outputs (X, Ψ_1, Ψ_2).
4. Let $\bar{\mathcal{B}}$ be the extractor such that $\mathsf{Adv}_{\mathcal{B},\bar{\mathcal{B}}}^{\mathbf{Lem\ A.1}} \leq \nu_2$. Invoke $\bar{\mathcal{B}}$ on input S_N using random coin $\mathsf{view}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$, and obtain output $(\mu_1, \ldots, \mu_N)$ from $\bar{\mathcal{B}}$. Output $(\Psi_1, \Psi_2, X, \mu_1, \ldots, \mu_N)$.

Note that according to the algorithm $\mathsf{CollRes}$, $\zeta = \mathtt{accept}$ implies that the verification is passed and $\Psi_2 = \left(\frac{\Psi_1}{\theta^X}\right)^{\beta}$ (the first equality test in equation (11.4)). We have

$$\zeta = \mathtt{accept} \ \wedge \ \Psi_2 \neq \prod_{i \in [N]} (v_i^{\beta})^{\mu_i} \quad \Rightarrow \quad \Psi_2 = \left(\frac{\Psi_1}{\theta^X}\right)^{\beta} \ \wedge \ \Psi_2 \neq \prod_{i \in [N]} (v_i^{\beta})^{\mu_i}$$

[1]Since $\mathcal{A}$ is invoked with the same random coin recovered from $\mathsf{view}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$, its behaviors become deterministic.

Hence, by applying Proposition 3, we have

$$\begin{aligned}\mathsf{Adv}^{\textbf{Lem A.2}}_{\mathcal{A},\bar{\mathcal{A}}} &= \Pr\left[\zeta = \texttt{accept} \ \wedge \ \Psi_2 \neq \prod_{i\in[N]} (v_i^{\beta})^{\mu_i}\right] \\ \leq \mathsf{Adv}^{\textbf{Lem A.1}}_{\mathcal{B},\bar{\mathcal{B}}} &= \Pr\left[\Psi_2 = \left(\frac{\Psi_1}{\theta^X}\right)^{\beta} \ \wedge \ \Psi_2 \neq \prod_{i\in[N]} (v_i^{\beta})^{\mu_i}\right] \leq \nu_2.\end{aligned}$$

□

A.6 A valid proof should be generated from points within intersection $\mathbf{D} \cap \mathbf{R}$

Theorem A.3 *Suppose Assumption 4 and Assumption 5 hold, and* FE *scheme constructed in Section 10.3 is* weak-IND-sID-CPA *secure. For any PPT algorithm $\mathcal{A}$, there exists PPT algorithm $\bar{\mathcal{A}}$, such that both $\mathsf{Adv}^{\textbf{Lem A.2}}_{\mathcal{A},\bar{\mathcal{A}}}$ and $\mathsf{Adv}^{\textbf{Thm A.3}}_{\mathcal{A},\bar{\mathcal{A}}}$ are negligible, where the advantage $\mathsf{Adv}^{\textbf{Thm A.3}}_{\mathcal{A},\bar{\mathcal{A}}}$ of $\mathcal{A}$ against $\bar{\mathcal{A}}$ w.r.t. scheme $\widetilde{\mathcal{E}} = (\mathsf{KGen}, \mathsf{DEnc}, \mathsf{CollRes})$ is defined as*

$$\mathsf{Adv}^{\textbf{Thm A.3}}_{\mathcal{A},\bar{\mathcal{A}}}(1^{\lambda}) \overset{\text{def}}{=} 1 - \Pr\left[\begin{array}{c}(\zeta, X, \Psi_2, \mathsf{view}^{\widetilde{\mathcal{E}}}_{\mathcal{A}}, \mathbf{D}, \mathbf{R}) \leftarrow \mathsf{Exp}^{\widetilde{\mathcal{E}}}_{\mathcal{A}}(1^{\lambda}); \\ (\{\mu_i : i \in [N]\}) \leftarrow \bar{\mathcal{A}}(\mathsf{view}^{\widetilde{\mathcal{E}}}_{\mathcal{A}}) : \\ \zeta = \texttt{accept} \wedge \Psi_2 = \prod_{i\in[N]} \left(v_i^{\beta}\right)^{\mu_i} \ \Rightarrow \\ \forall \vec{\boldsymbol{x}}_i \in \mathbf{D} \cap \mathbf{R}^{\complement}, \mu_i = 0\end{array}\right],$$

where v_i^{β} is the second component of tag $\vec{\boldsymbol{t}}_i$ for data point $\vec{\boldsymbol{x}}_i \in \mathbf{D}$ (See Step 2 of DEnc *in Section 11.1).*

Proof of Theorem A.3:

The idea of proof. For any PPT algorithm $\mathcal{A}$, applying Lemma A.2, let $\bar{\mathcal{A}}$ be the PPT algorithm such that $\mathsf{Adv}^{\textbf{Lem A.2}}_{\mathcal{A},\bar{\mathcal{A}}}$ is negligible. Using proof of contradiction, assume that $\mathsf{Adv}^{\textbf{Thm A.3}}_{\mathcal{A},\bar{\mathcal{A}}}$ is non-negligible (**Hypothesis!**). Based on $\mathcal{A}$ and $\bar{\mathcal{A}}$, we construct a PPT algorithm $\mathcal{B}$, such that $\mathcal{B}$ breaks weak-IND-sID-CPA security of FE scheme in Section 10.4.2 with non-negligible advantage

$$\mathsf{Adv}^{\text{weak-IND-sID-CPA}}_{\mathsf{FE},\mathcal{B}} \geq \frac{1}{4dN}\mathsf{Adv}^{\textbf{Thm A.3}}_{\mathcal{A},\bar{\mathcal{A}}} - \frac{1}{4}\nu_1.$$

where ν_1 is as in Assumption 4 and $\mathsf{Adv}^{\text{weak-IND-sID-CPA}}_{\mathsf{FE},\mathcal{B}}$ is defined in Section 10.4.2 The contradiction implies that our hypothesis is wrong, and thus Theorem A.3 is proved.

weak-IND-sID-CPA adversary $\mathcal{B}$ against FE scheme

Let $\widetilde{\mathcal{E}} = (\mathsf{KGen}, \mathsf{DEnc}, \mathsf{CollRes})$. We construct the adversary $\mathcal{B}$, which simulates the experiment $\mathsf{Exp}^{\widetilde{\mathcal{E}}}_{\mathcal{A}}$ by invoking the adversary $\mathcal{A}$, where $\mathcal{B}$ takes the role of Alice and $\mathcal{A}$ takes the role of Bob. Note that $\mathcal{B}$ makes only one delegation key query.

---

weak-IND-sID-CPA adversary $\mathcal{B}$ against FE scheme

Commit : Initialize $\mathcal{A}$'s status $\mathsf{view}_{\mathcal{A}}$. Invoke adversary $\mathcal{A}(\mathsf{view}_{\mathcal{A}})$, and $\mathcal{A}$ chooses a set $\mathbf{D} = \{\vec{\boldsymbol{x}}_1, \ldots, \vec{\boldsymbol{x}}_N\}$ of N d-dimensional points in $[\mathcal{Z}]^d$. $\mathcal{B}$ chooses $i^* \in [N]$ at random, and sends $\vec{\boldsymbol{x}}_{i^*}$ to the challenger $\mathcal{C}$ as the target identity.

Setup : The challenger $\mathcal{C}$ runs the setup algorithm $f\mathsf{Setup}$ and gives $\mathcal{A}$ the resulting system parameters $pk = (p, \mathbb{G}, \widetilde{\mathbb{G}}, e, \Omega)$, keeping the secret key $sk = (pk, \mathsf{params}, \mathsf{master\text{-}key}, \vec{\boldsymbol{\tau}})$ to itself.

Challenge : $\mathcal{C}$ chooses two plaintexts $\mathsf{Msg}_0, \mathsf{Msg}_1$ at random from the message space $\mathbb{Z}_p^*$, and a random bit $b \in \{0, 1\}$. $\mathcal{C}$ sets the challenge ciphertext to $\mathsf{CT} = f\mathsf{Enc}(\mathsf{Msg}_b, \vec{\boldsymbol{x}}_{i^*}, sk)$, and sends $(\mathsf{CT}, f_1(\mathsf{Msg}_0), f_1(\mathsf{Msg}_1))$ to $\mathcal{B}$.

Learning Phase : $\mathcal{B}$ (playing the role of Alice) interacts with $\mathcal{A}$ (playing the role of Bob) to simulate the experiment $\mathsf{Exp}^{\widetilde{\mathcal{E}}}_{\mathcal{A}}$. $\mathcal{B}$ proceeds as below.

KGen : Choose β, γ at random from $\mathbb{Z}_p^*$, and θ at random from $\widetilde{\mathbb{G}}$. Let (pk, sk) be the key pair generated by the challenger $\mathcal{C}$. Generate $\overline{pk}$ by removing Ω from pk, i.e. $\overline{pk} = (p, \mathbb{G}, \widetilde{\mathbb{G}}, e)$. Output $(\overline{pk}, sk)$ *Note: $\mathcal{B}$ knows pk, but not sk.*

DEnc :

1. Choose N random elements $W_1, \ldots, W_N$ from $\mathbb{Z}_p^*$ and N random elements $v_1, \ldots, v_N$ from $\widetilde{\mathbb{G}}$.
2. For each $i \in [N]$ except $i = i^*$, generate a tag $\vec{\boldsymbol{t}}_i = (\theta v_i, v_i^\beta, f_1(W_i)) \in \widetilde{\mathbb{G}}$. *Note: With* Ω*,* $\mathcal{B}$ *can evaluate function* $f_\rho(\cdot)$*.*
3. For each $i \in [N]$ except $i = i^*$, generate ciphertext CT_i:
   - Issue an encryption query $(W_i, \vec{\boldsymbol{x}}_i)$ to the challenger $\mathcal{C}$ and get reply CT'_i.
   - Apply the homomorphic property of the functional encryption scheme to attach v_i^γ to the ciphertext: $\mathsf{CT}_i \leftarrow f\mathsf{Mult}(\mathsf{CT}'_i, v_i^\gamma, pk)$.
4. Define the tag $\vec{\boldsymbol{t}}_{i^*}$ and ciphertext CT_{i^*} based on the challenge message $(\mathsf{CT}, f_1(\mathsf{Msg}_0), f_1(\mathsf{Msg}_1))$: Set $\vec{\boldsymbol{t}}_{i^*} = (\theta v_{i^*}, v_{i^*}^\beta, f_1(\mathsf{Msg}_0))$ and $\mathsf{CT}_{i^*} \leftarrow f\mathsf{Mult}(\mathsf{CT}, v_{i^*}^\gamma, pk)$.
5. Send $(\mathbf{D}, \mathbf{T} = \{\vec{\boldsymbol{t}}_i : i \in [N]\}, \mathbf{C} = \{\mathsf{CT}_i : i \in [N]\}, \overline{pk})$ to $\mathcal{A}$ (Bob).

$\mathcal{A}$ **in Learning Phase** : $\mathcal{A}$ issues queries $\mathbf{R}_1, \mathbf{R}_2, \ldots$. For each of such queries, $\mathcal{B}$ simulates Alice in $\mathsf{CollRes}$ as below.

Step A1: $\mathcal{B}$ makes a corresponding *anonymous delegation key query* $(\mathbf{R}_i)$ to the challenger $\mathcal{C}$, and sends the reply message $\vec{\boldsymbol{\delta}}_i$ to $\mathcal{A}$ (Bob).

Step A2: Do nothing. *Note: (1) According to our scheme and formulation, the accept/reject decision is always hidden from* $\mathcal{A}$*. So there is no need to do verification here. (2)* $\mathcal{B}$ *does not know the function key* ρ_i *for the delegation key* $\vec{\boldsymbol{\delta}}_i$*, so is not able to perform all verifications in step A2 of* $\mathsf{CollRes}$.

$\mathcal{A}$ **in Challenge Phase** : $\mathcal{A}$ issues a query with range $\mathbf{R}$: If $\vec{\boldsymbol{x}}_{i^*} \notin \mathbf{R}$, $\mathcal{B}$ simulates Alice in $\mathsf{CollRes}$ as below.

Step A1: $\mathcal{B}$ chooses a random element $\rho \in \mathbb{Z}_p^*$ and makes a corresponding *delegation key query* $(\mathbf{R}, \rho)$ to the challenger $\mathcal{C}$, and sends the reply message $\vec{\boldsymbol{\delta}}$ to $\mathcal{A}$ (Bob).

Step A2: Receive response (ζ, X, Ψ_2) for count query $\mathbf{R}$ associated with challenge message $\vec{\boldsymbol{\delta}}$) from $\mathcal{A}$. Perform all verifications as in Step A2 of $\mathsf{CollRes}$. *Note: In this case* $\mathcal{B}$ *does know the function key* ρ *for the delegation key* $\vec{\boldsymbol{\delta}}$ *and secret values* β, γ*, so is able to perform all verifications in step A2 of* $\mathsf{CollRes}$.

Otherwise, if $\vec{\boldsymbol{x}}_{i^*} \in \mathbf{R}$ then $\mathcal{B}$ abort and outputs a random bit $b' \in \{0,1\}$ (Denote this event as $\mathbf{E}_1$).

Guess : $\mathcal{B}$ outputs a guess bit b' as below.

1. Invoke the extractor $\bar{\mathcal{A}}(\mathsf{view}_{\mathcal{A}})$ for $\mathcal{A}$ and get output $\{\mu_i : i \in [N]\}$.
2. If $\zeta = \texttt{accept}, \Psi_2 = \prod_{i\in[N]} \left(v_i^{\beta}\right)^{\mu_i}$ and $\mu_{i^*} \neq 0$, then output $b' = 0$ (Denote this event as $\mathbf{E}_2$).
3. Otherwise, output a random bit $b' \in \{0,1\}$ (Denote this event as $\mathbf{E}_3$).

Note that all three events $\mathbf{E}_1$, $\mathbf{E}_2$ and $\mathbf{E}_3$ are mutually exclusive, and only $\mathbf{E}_2$ is the success case, and both of $\mathbf{E}_1$ and $\mathbf{E}_3$ correspond to failure.

$$\begin{aligned}
\mathsf{Pr}[b = b'] &= \mathsf{Pr}\left[\mathbf{E}_1 \vee \mathbf{E}_3\right] \mathsf{Pr}\left[b = b' | \mathbf{E}_1 \vee \mathbf{E}_3\right] + \mathsf{Pr}\left[b = b', \mathbf{E}_2\right] \\
&= \left(1 - \mathsf{Pr}\left[\mathbf{E}_2\right]\right) \times \frac{1}{2} + \mathsf{Pr}\left[b = b', \mathbf{E}_2\right] \\
&= \frac{1}{2} + \mathsf{Pr}\left[b = b', \mathbf{E}_2\right] - \frac{1}{2}\mathsf{Pr}\left[\mathbf{E}_2\right] \qquad \text{(A.11)}
\end{aligned}$$

Therefore,

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{weak\text{-}IND\text{-}sID\text{-}CPA}}_{\mathsf{FE},\mathcal{B}} &= \left| \mathsf{Pr}\left[b = b', \mathbf{E}_2\right] - \frac{1}{2}\mathsf{Pr}\left[\mathbf{E}_2\right] \right| \qquad \text{(A.12)} \\
&\geq \Bigg| \left| \frac{1}{2}\mathsf{Pr}\left[b = b', \mathbf{E}_2 \mid b = 0\right] - \frac{1}{4}\mathsf{Pr}\left[\mathbf{E}_2 \mid b = 0\right] \right| - \qquad \text{(A.13)} \\
&\qquad \left| \frac{1}{2}\mathsf{Pr}\left[b = b', \mathbf{E}_2 \mid b = 1\right] - \frac{1}{4}\mathsf{Pr}\left[\mathbf{E}_2 \mid b = 1\right] \right| \Bigg| \qquad \text{(A.14)}
\end{aligned}$$

$\mathsf{Adv}^{\mathsf{weak\text{-}IND\text{-}sID\text{-}CPA}}_{\mathsf{FE},\mathcal{B}}$ conditional on $b = 0$.

In case of $b = 0$, the forged tag $\vec{\boldsymbol{t}}_{i^*}$ and ciphertext CT_{i^*} are valid and consistent, and identical to those generated by DEnc. The simulated experiment $\mathsf{Exp}^{\widetilde{\mathcal{B}}}_{\mathcal{A}}$ by $\mathcal{B}$ is *identical* to a real one, to the view of $\mathcal{A}$ (even if $\mathcal{A}$ is computationally unbounded).

Recall that by the hypothesis, $\mathcal{A}$ is a Type II adversary but not a Type III adversary. That is, $\zeta = \texttt{accept}$ and $\Psi_2 = \prod_{i\in[N]} \left(v_i^{\beta}\right)^{\mu_i}$ with o.h.p, and there exists $\vec{\boldsymbol{x}}_i \in \mathbf{D} \cap \mathbf{R}^{\complement}$, such that $\mu_i \neq 0$ with non-negligible probability. We denote with $\mathbf{E}_4$ the event that $\zeta = \texttt{accept}$ and $\Psi_2 = \prod_{i\in[N]} \left(v_i^{\beta}\right)^{\mu_i}$, and there exists $\vec{\boldsymbol{x}}_i \in \mathbf{D} \cap \mathbf{R}^{\complement}$, such that $\mu_i \neq 0$.

$$\begin{aligned}\mathsf{Pr}[\mathbf{E}_4 \mid b=0] &= \mathsf{Pr}\left[\zeta = \texttt{accept} \wedge \Psi_2 = \prod_{i\in[N]} \left(v_i^{\beta}\right)^{\mu_i} \wedge \exists \vec{\boldsymbol{x}}_i \in \mathbf{D} \cap \mathbf{R}^{\complement}, \mu_i \neq 0 \mid b=0\right] \\ &= \mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm A.3}} \end{aligned} \tag{A.15}$$

Denote with $\mathbf{E}_5$ the event that $i^* \in S_{\#} = \{i \in [N] : \vec{\boldsymbol{x}}_i \in \mathbf{D} \cap \mathbf{R}^{\complement}, \mu_i \neq 0\}$. In the case of $b=0$, the event $\mathbf{E}_2$ is equivalent to conjunctions of three events: $\neg\mathbf{E}_1$, $\mathbf{E}_4$, and $\mathbf{E}_5$, i.e. $\mathbf{E}_2 \equiv \neg\mathbf{E}_1 \wedge \mathbf{E}_4 \wedge \mathbf{E}_5$. Since the conjunctions of $\mathbf{E}_4$ and $\mathbf{E}_5$ implies that $\vec{\boldsymbol{x}}_{i^*} \notin \mathbf{R}$ and $\xi \in [d]$ is independently and randomly chosen, we have

$$\mathsf{Pr}\left[\neg\mathbf{E}_1 \mid \mathbf{E}_4 \wedge \mathbf{E}_5 \wedge b=0\right] = \mathsf{Pr}\left[\vec{\boldsymbol{x}}_{i^*}[\xi] \notin \mathbf{A}_{\xi} \mid \mathbf{E}_4 \wedge \mathbf{E}_5 \wedge b=0\right] \geq \frac{1}{d}.$$

Therefore,

$$\begin{aligned}\mathsf{Pr}\left[\mathbf{E}_2 \mid b=0\right] &= \mathsf{Pr}\left[\neg\mathbf{E}_1 \wedge \mathbf{E}_4 \wedge \mathbf{E}_5 \mid b=0\right] \\ &= \mathsf{Pr}\left[\neg\mathbf{E}_1 \mid \mathbf{E}_4 \wedge \mathbf{E}_5 \wedge b=0\right] \mathsf{Pr}\left[\mathbf{E}_4 \wedge \mathbf{E}_5 \mid b=0\right] \\ &\geq \frac{1}{d}\mathsf{Pr}\left[\mathbf{E}_4 \mid b=0\right] \mathsf{Pr}\left[\mathbf{E}_5 \mid \mathbf{E}_4, b=0\right] \\ &= \frac{1}{d}\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm A.3}} \cdot \frac{1}{|S_{\#}|} \\ &\geq \frac{1}{dN}\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm A.3}} \qquad (\textit{ Event } \mathbf{E}_4 \textit{ implies that } S_{\#} \neq \emptyset)\end{aligned}$$

According to the construction of $\mathcal{B}$, if $b=0$ and event $\mathbf{E}_2$ occurs, the algorithm $\mathcal{B}$ will output $b'=0$. That is, $\mathsf{Pr}\left[b=b' | \mathbf{E}_2, b=0\right] = 1$.

Hence, conditional on $b = 0$, the advantage of $\mathcal{B}$ is

$$\mathsf{Adv}^{\mathsf{weak\text{-}IND\text{-}sID\text{-}CPA}}_{\mathsf{FE},\mathcal{B}}|_{b=0} = \left|\mathsf{Pr}\left[\mathbf{E}_2 \mid b=0\right]\left(\mathsf{Pr}\left[b=b' \mid \mathbf{E}_2, b=0\right] - \frac{1}{2}\right)\right| \geq \frac{1}{2dN}\mathsf{Adv}^{\textbf{Thm A.3}}_{\mathcal{A},\bar{\mathcal{A}}}. \tag{A.16}$$

$\mathsf{Adv}^{\mathsf{weak\text{-}IND\text{-}sID\text{-}CPA}}_{\mathsf{FE},\mathcal{B}}$ **conditional on** $b = 1$**.**

Next we show that $\mathsf{Pr}[\mathbf{E}_2 \mid b = 1]$ is negligible under Computational Diffie Hellman (**CDH**) assumption.

Claim A.6.1 *There exists a PPT algorithm which solves Computational Diffie Hellman problem with probability equal to* $\mathsf{Pr}[\mathbf{E}_2 \mid b = 1]$.

Proof of Claim A.6.1: The proof idea is: Given input (v, v^γ, u), we choose a random number R, and simulate the scheme $\widetilde{\mathcal{E}} = (\mathsf{KGen}, \mathsf{DEnc}, \mathsf{CollRes})$ by embedding (u, R) into the tag/ciphertext for the target index i^* and embedding (v, v^γ) into tag/ciphertext for the other index, . If $b = 1$ and event $\mathbf{E}_2$ occurs, we try to compute u^γ with the help of adversary $\mathcal{A}$.

Algorithm $\mathcal{D}$: Break Computational Diffie Hellman problem

1. Input is $(v, v^a, u) \in \widetilde{\mathbb{G}}^3$, where the unknown exponent a is uniformly randomly distributed over $\mathbb{Z}_p^*$. The goal is to output u^a.
2. Simulate the scheme $\widetilde{\mathcal{E}} = (\mathsf{KGen}, \mathsf{DEnc}, \mathsf{CollRes})$:

   KGen: The same as in Section 11.1, except that let γ be the unknown value a: $\gamma \leftarrow a$.

   DEnc: (a) Choose N random elements $W_1, \ldots, W_N$ from $\mathbb{Z}_p^*$. Choose $i^* \in [N]$ at random.

   (b) For each $i \in [N]$ except i^*:

i. choose $z_i \in \mathbb{Z}_p^*$ at random and compute $v_i = v^{z_i}$ and $v_i^{\gamma} = (v^{\gamma})^{z_i} = (v^a)^{z_i}$;

ii. generate a tag $\vec{\boldsymbol{t}}_i = (v_i, v_i^{\beta}, f_1(W_i)) \in \widetilde{\mathbb{G}}^3$;

iii. generate a ciphertext CT_i as in Section 11.1.

(c) For i^*:

i. generate a tag $\vec{\boldsymbol{t}}_{i^*} = (v_{i^*}, v_{i^*}^{\beta}, f_1(W_{i^*}))$ where $v_{i^*} = u$;

ii. generate a ciphertext $\mathsf{CT}_{i^*} = f\mathsf{Mult}(\mathsf{CT}',\ R,\ pk)$, where $\mathsf{CT}' \leftarrow f\mathsf{Enc}(W_{i^*},\ \vec{\boldsymbol{x}}_{i^*},\ sk)$ and R is a random element in $\widetilde{\mathbb{G}}$.

(d) Send all tags and ciphertexts and pk to Bob as in Section 11.1.

CollRes: The same as in Section 11.1, except that the simulator does not perform the verifications in step A2 of $\mathsf{CollRes}$.

Note: Since γ is unknown, some verifications can not be done.

3. Invoke the adversary $\mathcal{A}$ and simulate the experiment $\mathsf{Exp}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$ using the above simulated scheme $\widetilde{\mathcal{E}}$.

   Let $(X, \Psi_1, \Psi_2, \Psi_3, \Psi_4)$ denote the reply returned by adversary $\mathcal{A}$ on the challenging query range $\mathbf{R}$ and ρ be the corresponding random nonce. *Note: When the adversary $\mathcal{A}$ is in challenging phase, the verification cannot be done, since $\gamma = a$ is unknown.*

4. Let $\mathsf{view}_{\mathcal{A}}$ be the view of $\mathcal{A}$ after the experiment. Invoke the extractor $\bar{\mathcal{A}}$ w.r.t. $\mathcal{A}$, and obtain output: $\{\mu_i : i \in [N]\} \leftarrow \bar{\mathcal{A}}(\mathsf{view}_{\mathcal{A}})$.

5. Compute ϕ as below and output $\phi^{\mu_{i^*}^{-1}}$:

$$\phi \leftarrow \frac{\Psi_4}{\Psi_3^{\rho} \cdot \prod_{i \in [N]}^{i \neq i^*} (v_i^{\gamma})^{\mu_i}}$$

Denote the experiment $\mathsf{Exp}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$ simulated by $\mathcal{D}$ as $\mathsf{Exp}_{\mathcal{D}}$; denote the experiment $\mathsf{Exp}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$ simulated by $\mathcal{B}$ in the case of $b = 1$ as $\mathsf{Exp}_{\mathcal{B}}$. Both simulated experiments $\mathsf{Exp}_{\mathcal{D}}$ and $\mathsf{Exp}_{\mathcal{B}}$ are *identical*, to the view of adversary $\mathcal{A}$ (even if $\mathcal{A}$ is computationally unbounded):

- In both simulated experiments, for each $i \in [N]$ except i^*, the tag $\vec{\boldsymbol{t}}_i$ and ciphertext CT_i are consistent and *identical* as those generated by the algorithm DEnc in Section 11.1.

- In both simulated experiments, the ciphertext CT_{i^*} is *independent* on the tag $\vec{\boldsymbol{t}}_{i^*}$:
  - In $\mathsf{Exp}_{\mathcal{D}}$, $\vec{\boldsymbol{t}}_{i^*} = (v_{i^*},\ v_{i^*}^{\beta},\ f_1(W_{i^*}))$ and ciphertext $\mathsf{CT}_{i^*} = f\mathsf{Mult}(\mathsf{CT}',\ R,\ pk)$, where $\mathsf{CT}' \leftarrow f\mathsf{Enc}(W_{i^*},\ \vec{\boldsymbol{x}}_{i^*},\ sk)$ and R is a random element in $\widetilde{\mathbb{G}}$. That is, the ciphertext CT_{i^*} is randomized due to the independent randomness R in the execution of $f\mathsf{Mult}$.
  - In $\mathsf{Exp}_{\mathcal{B}}$, $\vec{\boldsymbol{t}}_{i^*} = (v_{i^*},\ v_{i^*}^{\beta},\ f_1(\mathsf{Msg}_0))$ and $\mathsf{CT}_{i^*} \leftarrow f\mathsf{Mult}(\mathsf{CT}, v_{i^*}^{\gamma}, pk)$, where $\mathsf{CT} \leftarrow f\mathsf{Enc}(\mathsf{Msg}_1,\ \vec{\boldsymbol{x}}_{i^*},\ sk)$ is the ciphertext of Msg_1, and Msg_0, Msg_1 are two independent random elements in $\mathbb{Z}_p^*$. That is, the ciphertext CT_{i^*} is randomized[2] due to the independent randomness Msg_1 in the execution of $f\mathsf{Enc}$.
- In both simulated experiments, for any range query $\mathbf{R}$, $\mathcal{A}$ receives the same (identically distributed) reply as in $\mathsf{CollRes}$ in Section 11.1.

We remark that the differences in the capabilities of verifications in the two simulated experiments, are invisible to $\mathcal{A}$, since all accept/reject decisions are completely hidden from $\mathcal{A}$.

Suppose $b = 1$ and event $\mathbf{E}_2$ occurs[3], that is, $\zeta = \mathtt{accept}$ and $\Psi_2 = \prod_{i\in[N]} \left(v_i^{\beta}\right)^{\mu_i}$ and $\mu_{i^*} \neq 0$. It is easy to show that

$$\phi = v_{i^*}^{\gamma\mu_{i^*}};\quad \phi^{\mu_{i^*}^{-1}} = v_{i^*}^{\gamma} = u^{\gamma} = u^{a}.$$

[2]One can verify that randomization in $f\mathsf{Mult}$ is equivalent to randomization in $f\mathsf{Enc}$, by checking the constructions of $f\mathsf{Mult}$ and $f\mathsf{Enc}$ and the underlying BBG HIBE scheme. Note that public key params of the underlying BBG HIBE scheme and W_i's (Random numbers as in Step 1 of DEnc in Section 11.1) are unknown to the adversary $\mathcal{A}$.

[3]Note that the algorithm $\mathcal{D}$ cannot tell whether $\mathbf{E}_2$ occurs or not, since $\mathcal{D}$ does not know γ thus cannot perform some verifications. $\mathcal{D}$ simply guesses that event $\mathbf{E}_2$ does occur, and this guess will be correct with probability $\mathsf{Pr}[\mathbf{E}_2 | b = 1]$

Hence, the above algorithm $\mathcal{D}$ solve the **CDH** problem with probability

$$\mathsf{Pr}[\mathbf{E}_2 \mid b=1]\ \mathsf{Pr}[\phi^{\mu_{i^*}^{-1}} = u^a \mid \mathbf{E}_2, b=1] = \mathsf{Pr}[\mathbf{E}_2 \mid b=1].$$

□

Therefore, under **CDH** assumption, $\mathsf{Pr}[\mathbf{E}_2 \mid b=1] \leq \nu_1$, where $\nu_1(\cdot)$ is some negligible function. As a result, conditional on $b=1$, the advantage of $\mathcal{B}$ in breaking the FE scheme is

$$\mathsf{Adv}_{\mathsf{FE},\mathcal{B}}^{\mathsf{weak\text{-}IND\text{-}sID\text{-}CPA}}|_{b=1} = \left|\mathsf{Pr}\left[\mathbf{E}_2 \mid b=1\right]\left(\mathsf{Pr}\left[b=b' \mid \mathbf{E}_2, b=1\right] - \frac{1}{2}\right)\right| \leq \frac{1}{2}\nu_1. \quad \text{(A.17)}$$

$$\begin{aligned}\mathsf{Adv}_{\mathsf{FE},\mathcal{B}}^{\mathsf{weak\text{-}IND\text{-}sID\text{-}CPA}} &\geq \left|\frac{1}{2}\mathsf{Adv}_{\mathsf{FE},\mathcal{B}}^{\mathsf{weak\text{-}IND\text{-}sID\text{-}CPA}}|_{b=0} - \frac{1}{2}\mathsf{Adv}_{\mathsf{FE},\mathcal{B}}^{\mathsf{weak\text{-}IND\text{-}sID\text{-}CPA}}|_{b=1}\right| \\ &\geq \frac{1}{4dN}\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm A.3}} - \frac{1}{4}\nu_1.\end{aligned}$$

□

A.7 A valid proof should be generated by processing each point within intersection $\mathbf{D} \cap \mathbf{R}$ *for exactly once*

Theorem A.4 *Suppose Assumption 4 and Assumption 5 hold, and* BBG *[BBG05] HIBE scheme is* $\mathsf{IND\text{-}sID\text{-}CPA}$ *secure. For any PPT algorithm* $\mathcal{A}$*, there exists a PPT adversary* $\bar{\mathcal{A}}$*, such that all of* $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Lem A.2}}$*,* $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm A.3}}$*, and* $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm A.4}}$ *are negligible, where the advantage* $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm A.4}}$ *of* $\mathcal{A}$ *against* $\bar{\mathcal{A}}$ *w.r.t. scheme* $\mathcal{E} =$

$(\mathsf{KGen}, \mathsf{DEnc}, \mathsf{ProVer})$ *is defined as*

$$\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm A.4}}(1^\lambda) \overset{\text{def}}{=} 1 - \mathsf{Pr}\left[\begin{array}{l} (\zeta, X, \Delta, \mathsf{view}_{\mathcal{A}}^{\mathcal{E}}, \mathbf{D}, \mathbf{R}) \leftarrow \mathsf{Exp}_{\mathcal{A}}^{\mathcal{E}}(1^\lambda); \\ (\{\mu_i : i \in [N]\}) \leftarrow \bar{\mathcal{A}}(\mathsf{view}_{\mathcal{A}}^{\mathcal{E}}) : \\ \qquad \zeta = \texttt{accept} \Rightarrow \\ \qquad\qquad \left(\Delta = \prod_{i\in[N]} \left(v_i^\beta\right)^{\mu_i} \wedge \forall i \in [N], \mu_i = 1\right) \end{array}\right],$$

where v_i^β is the second component of tag $\vec{\boldsymbol{t}}_i$ for data point $\vec{\boldsymbol{x}}_i \in \mathbf{D}$ (See Step 2 of DEnc *in Section 11.1).*

Proof of Theorem A.4:

Idea of proof. For any PPT algorithm $\mathcal{A}$, applying Theorem A.3, let $\bar{\mathcal{A}}$ be the PPT algorithm, such that $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Lem A.2}} \le \epsilon_5$ and $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm A.3}} \le \epsilon_6$ for some negligible functions $\epsilon_5(\cdot)$ and $\epsilon_6(\cdot)$. Using proof of contradiction, assume that $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm A.4}} \ge \epsilon_7$ for some non-negligible function $\epsilon_7(\cdot)$. We construct a PPT algorithm $\mathcal{B}$ based on $\mathcal{A}$ and $\bar{\mathcal{A}}$, such that $\mathcal{B}$ breaks Discrete Log Problem with non-negligible advantage $\epsilon_7 - (2d+1)(\epsilon_5 + \epsilon_6)$.

Denote with $\mathbf{E}_1$ the event that $\zeta = \texttt{accept} \bigwedge \Delta \neq \prod_{i\in[N]} \left(v_i^\beta\right)^{\mu_i}$, and with $\mathbf{E}_2$ the event that $\zeta = \texttt{accept} \bigwedge \Delta = \prod_{i\in[N]} \left(v_i^\beta\right)^{\mu_i} \wedge \exists j \in [N], \mu_j \neq 1$. We can split the probability $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm A.4}}$ into two parts,

$$\begin{aligned}
\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm A.4}} &= \mathsf{Pr}\left[\begin{array}{l} (\zeta, X, \vec{\boldsymbol{\Psi}}, \mathsf{view}_{\mathcal{A}}^{\mathcal{E}}, \mathbf{D}, \mathbf{R}) \leftarrow \mathsf{Exp}_{\mathcal{A}}^{\mathcal{E}}(1^\lambda); \\ (\{\mu_i : i \in [N]\}) \leftarrow \bar{\mathcal{A}}(\mathsf{view}_{\mathcal{A}}^{\mathcal{E}}) : \\ \qquad \zeta = \texttt{accept} \bigwedge \Delta \neq \prod_{i\in[N]} \left(v_i^\beta\right)^{\mu_i} \end{array}\right] \\
&\quad + \mathsf{Pr}\left[\begin{array}{l} (\zeta, X, \vec{\boldsymbol{\Psi}}, \mathsf{view}_{\mathcal{A}}^{\mathcal{E}}, \mathbf{D}, \mathbf{R}) \leftarrow \mathsf{Exp}_{\mathcal{A}}^{\mathcal{E}}(1^\lambda); \\ (\{\mu_i : i \in [N]\}) \leftarrow \bar{\mathcal{A}}(\mathsf{view}_{\mathcal{A}}^{\mathcal{E}}) : \\ \qquad \zeta = \texttt{accept} \bigwedge \Delta = \prod_{i\in[N]} \left(v_i^\beta\right)^{\mu_i} \\ \wedge \exists j \in [N], \mu_j \neq 1 \end{array}\right] \\
&= \mathsf{Pr}[\mathbf{E}_1] + \mathsf{Pr}[\mathbf{E}_2].
\end{aligned}$$

Part I: $\mathsf{Pr}[\mathbf{E}_1] \le (2d+1)\left(\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Lem A.2}} + \mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm A.3}}\right)$.

Suppose that: (1) The challenging query range is $\mathbf{R}$. (2) Alice partitions $\mathbf{R}^{\mathsf{C}}$ into $2d$ rectangular ranges $\mathbf{R}_1, \ldots, \mathbf{R}_{2d}$ and sets $\mathbf{R}_0 = \mathbf{R}$. (3) For $0 \le \ell \le 2d$, denote with $(\zeta_\ell, X_\ell, \Psi_2^{(\ell)})$ the reply returned by adversary $\mathcal{A}$ in the execution of $\mathsf{CollRes}$ on range $\mathbf{R}_\ell$. (4) Denote with (ζ, X, Δ) the output of Alice in the execution of ProVer. (5) Recall that Alice keeps the value $\Delta = \prod_{i \in [N]} v_i^{\beta}$.

According to the construction in Section 11.1 (i.e. Step 3 of ProVer), we have

$$\left(\bigwedge_{\ell \in [0,2d]} \zeta_\ell = \mathtt{accept}\right) \wedge \ \Delta = \prod_{\ell \in [0,2d]} \Psi_2^{(\ell)} \ \Leftrightarrow \ \zeta = \mathtt{accept} \qquad (\textit{Denoted as statement } A) \tag{A.18}$$

In additional to statement A, let us define statement A_ℓ and B_ℓ as below:

A_ℓ**:** $\zeta_\ell = \mathtt{accept} \quad \Rightarrow \quad \Psi_2^{(\ell)} = \prod_{i \in [N]} v_i^{\beta \mu_{\ell,i}}$, $0 \le \ell \le 2d$.

B_ℓ**:** $\zeta_\ell = \mathtt{accept} \wedge \Psi_2^{(\ell)} = \prod_{i \in [N]} v_i^{\beta \mu_{\ell,i}} \quad \Rightarrow \quad \forall \vec{\boldsymbol{x}}_i \in \mathbf{D} \cap \mathbf{R}_\ell^{\mathsf{C}}, \mu_{\ell,i} = 0$, $0 \le \ell \le 2d$.

Let us define integers μ_i, $i \in [N]$, based on integers $\mu_{\ell,i}$'s, $\ell \in [0, 2d], i \in [N]$, as below: For each $i \in [N]$, find the unique rectangular range $\mathbf{R}_\ell$, $\ell \in [0, 2d]$, such that data point $\vec{\boldsymbol{x}}_i \in \mathbf{D} \cap \mathbf{R}_\ell$, then set $\mu_i = \mu_{\ell,i}$.

The conjunctions of statements A, A_ℓ's ($0 \le \ell \le 2d$), and B_ℓ's ($0 \le \ell \le 2d$), directly imply the following statement

$$\zeta = \mathtt{accept} \qquad \Rightarrow \qquad \Delta = \prod_{\vec{\boldsymbol{x}}_i \in \mathbf{D} \cap \left(\bigcup_{0 \le \ell \le 2d} \mathbf{R}_\ell\right)} v_i^{\beta \mu_i} = \prod_{\vec{\boldsymbol{x}}_i \in \mathbf{D}} v_i^{\beta \mu_i}. \tag{A.19}$$

Applying Proposition 3 and Proposition 4 in Appendix A.2, we have

$$\begin{aligned}\mathsf{Pr}\left[\zeta = \mathtt{accept} \Rightarrow \Delta = \prod_{\vec{\boldsymbol{x}}_i \in \mathbf{D}} v_i^{\beta\mu_i}\right] &\geq \mathsf{Pr}\left[A \wedge A_0 \wedge \ldots \wedge A_{2d} \wedge B_0 \wedge \ldots \wedge B_{2d}\right] \\ &\geq 1 - \mathsf{Pr}[\neg A] - \sum_{\ell=0}^{2d} \mathsf{Pr}\left[\neg A_\ell\right] - \sum_{\ell=0}^{2d} \mathsf{Pr}\left[\neg B_\ell\right] \\ &\geq 1 - 0 - \sum_{\ell=0}^{2d} \mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Lem A.2}} - \sum_{\ell=0}^{2d} \mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm A.3}} \\ &= 1 - (2d+1)\left(\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Lem A.2}} + \mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm A.3}}\right).\end{aligned}$$

Therefore,

$$\mathsf{Pr}[\mathbf{E}_1] = 1 - \mathsf{Pr}\left[\zeta = \mathtt{accept} \Rightarrow \Delta = \prod_{\vec{\boldsymbol{x}}_i \in \mathbf{D}} v_i^{\beta\mu_i}\right] \leq (2d+1)\left(\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Lem A.2}} + \mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm A.3}}\right).$$

Part II: Break Discrete Log Problem.

Applying the result in Part I, we have $\mathsf{Pr}[\mathbf{E}_2] = \mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm A.4}} - \mathsf{Pr}[\mathbf{E}_1] \geq \mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm A.4}} - (2d+1)\left(\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Lem A.2}} + \mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm A.3}}\right)$. We construct the following algorithm to break the Discrete Log Problem.

DLP Adversary $\mathcal{B}$

1. The input is $(v, v^a) \in \widetilde{\mathbb{G}}^2$. The goal is to find $a \in \mathbb{Z}_p$.

2. Invoke scheme $\mathcal{E} = (\mathsf{KGen}, \mathsf{DEnc}, \mathsf{ProVer})$ with f_2 defined as above, with the following modification:

    - In DEnc, for each $i \in [N]$, choose $y_i, z_i \in \mathbb{Z}_p$ at random and set $v_i = (v^a)^{y_i} \cdot v^{z_i} \in \widetilde{\mathbb{G}}$.

    Note: $\mathcal{B}$ has full information of private key.

3. Simulate the experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathcal{E}}$, by invoking the adversary $\mathcal{A}$ (playing the role Bob) to interact with Alice in $\mathcal{E}$. Then invoke $\bar{\mathcal{A}}(\mathsf{view}_{\mathcal{A}}^{\mathcal{E}})$ to obtain $\{\mu_i : i \in [N]\}$.

4. With probability equal to $\mathsf{Pr}[\mathbf{E}_2]$, it holds that $\zeta = \texttt{accept} \bigwedge \Delta = \prod_{i\in[N]} \left(v_i^\beta\right)^{\mu_i} \bigwedge \exists j \in [N], \mu_j \neq 1$.
5. According to our scheme in Section 11.1 (Step 4 of DEnc), $\Delta = \prod_{i\in[N]} v_i^\beta$. So a univariable equation in the unknown variable a of order 1 in group $\mathbb{Z}_p$ can be formed by substituting $v_j = v^{ay_j+z_j}$. Solve this equation and get a root a^*. Output a^*.

The PPT algorithm $\mathcal{B}$ constructed as above breaks **DLP** with probability $\mathsf{Pr}[\mathbf{E}_2]$. Therefore, under Computational Diffie Hellman Assumption 4, **DLP** is infeasible and thus $\mathsf{Pr}[\mathbf{E}_2]$ has to be negligible.

Combining results in Part I and II, we have

$$\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm A.4}} \leq (2d+1)\left(\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Lem A.2}} + \mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm A.3}}\right) + \mathsf{Adv}_{\mathcal{B}}^{\textbf{DLP}}.$$

□

A.8 Proof of Main Theorem 11.1

Theorem 11.1 (Main Theorem) *Suppose Assumption 4 and Assumption 5 hold, and BBG [BBG05] HIBE scheme is IND-sID-CPA secure. Then the $\mathcal{RC}$ protocol $\mathcal{E} =$ $(\mathsf{KGen}, \mathsf{DEnc}, \mathsf{ProVer})$ constructed in Section 11.1 is $\mathcal{VRC}$ w.r.t. function $F(\cdot,\cdot)$ as defined in Section 9.1, under Definition 11. Namely, $\mathcal{E}$ is* correct *and* sound *w.r.t. function F.*

Proof of Theorem 11.1: The correctness is straightforward once we have Lemma 10.1. Here we save the details and focus on the soundness part.

Suppose $\mathcal{E}$ is not sound, i.e. there exists a PPT algorithm $\mathcal{A}$, with non-negligible advantage ϵ_6 against $\mathcal{E}$:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathcal{E}} = \mathsf{Pr}\left[\begin{array}{c}(\zeta, X, \vec{\mathbf{\Psi}}, \mathsf{view}_{\mathcal{A}}^{\mathcal{E}}, \mathbf{D}, \mathbf{R}) \leftarrow \mathsf{Exp}_{\mathcal{A}}^{\mathcal{E}}(1^\lambda);\\ \zeta = \texttt{accept} \bigwedge X \neq F(\mathbf{D},\mathbf{R}) \pmod p\end{array}\right] \geq \epsilon_6.$$

Applying Theorem A.4, let $\bar{\mathcal{A}}$ be the extractor for $\mathcal{A}$ such that all of $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Lem A.2}}$, $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm A.3}}$, and $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm A.4}}$ are negligible.

We intend to construct a PPT algorithm $\mathcal{B}$ based on $\mathcal{A}$ to break Assumption 4 (Computational Diffie-Hellman Problem), and argue that $\mathcal{B}$ succeeds with probability about ϵ_6, with the help of $\bar{\mathcal{A}}$, under Assumption 4, Assumption 5, and the assumption that BBG [BBG05] HIBE is $\mathsf{IND\text{-}sID\text{-}CPA}$ secure. The contradiction will imply that such adversary $\mathcal{A}$ does not exist and the constructed scheme $\mathcal{E}$ is sound.

Adversary $\mathcal{B}$ against Computational Diffie-Hellman Problem

1. The input is $(u, u^\beta, v^\beta) \in \widetilde{\mathbb{G}}$. The goal is to find v.
2. Choose a random number R_1 from $\widetilde{\mathbb{G}}$. Then $R_1 = v\theta$ for some unknown $\theta \in \widetilde{\mathbb{G}}$.
3. For $1 \leq j \leq m$, choose z_j at random from $\mathbb{Z}_p^*$ and set $u_j \leftarrow u^{z_j}$ and compute $u_j^\beta = \left(u^\beta\right)^{z_j}$. Let $W_m = (\{u_j, u_j^\beta : j \in [m]\})$.
4. Convert $(W_m, R_1, R_2 = v^\beta)$ to $S_{m+1} = \{(\theta v_i, v_i^\beta)\}_{i=0}^m$ in the same way as in construction of algorithm $\mathcal{A}_1$ in the proof of Lemma A.1 in Appendix A.5.1.
5. From S_{m+1}, simulate the scheme $\mathcal{E}$ just as adversary $\mathcal{B}$ in the proof of Lemma A.2 in Appendix A.5.2.
6. Invoke the adversary $\mathcal{A}$ and simulate the experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathcal{E}}$. Let $(X, \bar{\Psi}_1, \bar{\Psi}_2, \bar{\Psi}_3, \bar{\Psi}_4)$ be the reply returned by adversary $\mathcal{A}$ on challenging query range $\mathbf{R}$ in the execution of $\mathsf{CollRes}$.
7. Simulate the experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathcal{E}}$ honestly (just using the algorithm Eval instead of adversary $\mathcal{A}$) and get query result $Y = |\mathbf{D} \cap \mathbf{R}|$ and proof $(\Psi_1, \Psi_2, \Psi_3, \Psi_4)$.
8. Let Z be the inverse of $(X - Y)$ modulo p and compute $\theta' = \left(\frac{\bar{\Psi}_1}{\Psi_1}\right)^Z$.
   Note: (1) $Y = F(\mathbf{D}, \mathbf{R})$. *(2) If* $\mathcal{A}$ *succeeds, then* $X \neq F(\mathbf{D}, \mathbf{R}) \pmod p$. *Recall the definition of function* $F : \mathbb{D} \times \mathbb{R} \to \mathbb{Z}_p$ *in Section 9.1.*
9. Output $\frac{R_1}{\theta'}$.

Note that as in proof of Lemma A.2, the simulated scheme $\mathcal{E}$ is identical to a real one from the view of adversary $\mathcal{A}$.

For the constructed adversary $\mathcal{B}$, we make the following claim:

Claim A.8.1 *Suppose Assumption 4 and Assumption 5 hold, and* BBG *[BBG05] HIBE scheme is* IND-sID-CPA *secure. If $\mathcal{A}$ succeeds, it holds with o.h.p. (i.e. with probability $(1 - negl)$) that* $\left(\frac{\bar{\Psi}_1}{\theta^X}\right)^\beta = \bar{\Psi}_2 = \Psi_2 = \left(\frac{\Psi_1}{\theta^Y}\right)^\beta$.

Proof of Claim A.8.1: If $\mathcal{A}$ succeeds, then its output $(X, \bar{\Psi}_1, \bar{\Psi}_2, \bar{\Psi}_3, \bar{\Psi}_4)$ will pass all verifications in the scheme $\mathcal{E}$ (Step A2 of CollRes and Step 3 in ProVer in Section 11.1). So we have

$$\left(\frac{\bar{\Psi}_1}{\theta^X}\right)^\beta = \bar{\Psi}_2, \ \zeta = \texttt{accept}. \tag{A.20}$$

where $\zeta \in \{\texttt{accept}, \texttt{reject}\}$ denotes the corresponding decision (a part of output of ProVer) regarding $\mathcal{A}$'s reply on the challenging query.

Let $(\mu_1, \ldots, \mu_N)$ be the output of extractor $\bar{\mathcal{A}}$. Under Assumption 4, Assumption 5 and the assumption that BBG [BBG05] HIBE scheme is IND-sID-CPA secure, by applying Lemma A.2, Theorem A.3 and Theorem A.4, the following implications hold with o.h.p.,

$$\begin{aligned} \zeta = \texttt{accept} \ &\Rightarrow \ \left(\Delta = \prod_{i \in [N]} \left(v_i^\beta\right)^{\mu_i} \ \wedge \ \forall i \in [N], \mu_i = 1\right); \\ \zeta = \texttt{accept} \ &\Rightarrow \ \bar{\Psi}_2 = \prod_{\vec{\boldsymbol{x}}_i \in \mathbf{D} \cap \mathbf{R}} \left(v_i^\beta\right)^{\mu_i}. \end{aligned}$$

Hence, conditional on $\mathcal{A}$ succeeds, with o.h.p. we have

$$\bar{\Psi}_2 = \prod_{\vec{\boldsymbol{x}}_i \in \mathbf{D} \cap \mathbf{R}} \left(v_i^\beta\right)^{\mu_i} = \prod_{\vec{\boldsymbol{x}}_i \in \mathbf{D} \cap \mathbf{R}} v_i^\beta. \tag{A.21}$$

The output $(X, \Psi_1, \Psi_2, \Psi_3, \Psi_4)$ returned by an honest Bob also passes all verifications (Since the scheme $\mathcal{E}$ is *correct*).

$$\left(\frac{\Psi_1}{\theta^Y}\right)^\beta = \Psi_2, \text{ where } \Psi_2 = \prod_{\vec{\boldsymbol{x}}_i \in \mathbf{D} \cap \mathbf{R}} v_i^\beta \quad \text{is computed following the scheme.} \tag{A.22}$$

Combing equations (A.20)(A.21)(A.22), Claim A.8.1 can be implied directly:

$$\left(\frac{\bar{\Psi}_1}{\theta^X}\right)^\beta = \bar{\Psi}_2 = \Psi_2 = \left(\frac{\Psi_1}{\theta^Y}\right)^\beta .$$

□

From Claim A.8.1, it is straightforward that

$$\mathsf{Pr}\left[\frac{R_1}{\theta'} = v\right] = \mathsf{Pr}\left[\theta' = \theta\right] \geq \mathsf{Pr}\left[\mathcal{A} \text{ succeeds}\right](1 - negl) \geq \epsilon_6(1 - negl),$$

where $negl(\cdot)$ is some negligible function. Therefore, the constructed algorithm $\mathcal{B}$ breaks Assumption 4 with non-negligible probability $\epsilon_6(1 - negl)$. The contradiction implies that our hypothesis is wrong: such adversary $\mathcal{A}$ does not exist. Thus, the constructed scheme $\mathcal{E}$ is sound and Theorem 11.1 is proved. □