TRADE-OFFS AMONG DATA SECURITY, USABILITY AND COMPLEXITY IN MOBILE CLOUD COMPUTING

By

Kai Zhou

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Electrical Engineering — Doctor of Philosophy

2018

ABSTRACT

TRADE-OFFS AMONG DATA SECURITY, USABILITY AND COMPLEXITY IN MOBILE CLOUD COMPUTING

By

Kai Zhou

The proliferation of mobile cloud computing changes the way that data is utilized. Huge volume of data is collected at the network edge, transmitted through the network, gathered in the cloud, and utilized by various end-users. Data security, usability, and complexity are among the most important design issues in mobile computing environment. Trade-offs generally exist among these issues. While data encryption is a primary mechanism to ensure data security, it will inevitably introduce some computational overhead to the data owners. Also, data encryption will reduce data usability since typically it is hard to conduct computations over encrypted data.

In this thesis, we focus on designing secure, efficient and versatile protocols that can achieve trade-offs among data security, data usability, and computational complexity, featuring the mobile cloud computing environment. In particular, we explore the trade-offs from two aspects. First, we design secure computation outsourcing schemes for a wide variety of computational problems such as general scientific computation and cryptographic computation, trying to alleviate the computational overhead at the user side while preserving the security of the outsourced data. Our proposed scheme is cost-aware in that it provides different levels of security protection for the outsourced problem with different computational overhead. Second, we design a specific encryption scheme that enables certain computations to be conducted directly over encrypted data. In this way, data users can directly utilize encrypted data to meet the demands of various applications without compromising data

security. More specifically, our proposed encryption scheme encrypts two vectors in such a manner that the inner product of the vectors can be evaluated and compared to a pre-defined threshold. We show that an encryption scheme can be utilized as an essential building block to construct various privacy-preserving applications such as an online biometric authentication system. Our proposed schemes are highly efficient and suitable for resource-constrained devices in the mobile cloud computing environment to greatly expand the computational capacity and/or extend battery life.

Dedicated to my family, especially to my grandmother,
who brings me up and sets a good role model in my life.

ACKNOWLEDGMENTS

I would like to take the opportunity to express my deep appreciation to my advisor, Dr. Jian Ren, for his constant support, guidance, and encouragement throughout my Ph.D. years. He makes great effort to help me through many difficulties in academic research and personal development and growth. He himself sets a great example on these areas for me.

I want to thank Dr. Tongtong Li, Dr. Mi Zhang from Department of Electrical and Computer Engineering and Dr. Richard Embody from Department of Computer Science and Engineering for serving on my committee. I am really grateful to them for their kind support. I would express my special thanks to Dr. Tongtong Li, who provides great guidance in my research as well as in my life.

I am deeply indebted to my labmates, Dr. Jian Li, Dr. Di Tang, Afifi, and Ehab, for their valuable discussions on the research issues, as well as their helpful advice on the daily life on and off the campus. I am also grateful to all my friends who have made my life at Michigan State University an enjoyable experience.

I would like to thank my parents for their unyielding love and continuous support through all the ups and downs, without which nothing could ever be possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF PROTOCOLS

Chapter 1

Introduction

1.1 Backgrounds

The proliferation of cloud computing, ubiquitous mobile computing, Internet of Things (IoT) and big data spawns a new computational paradigm known as the mobile cloud computing. In such an environment, huge volume of data is collected from ubiquitous mobile devices, transmitted through the network, aggregated in cloud platforms, and utilized by end-users. There is a trend that users' private data is moving to the cloud, either for storage or computation. Numerous applications are merging from such a mobile cloud computing environment. Computation outsourcing, as a service provided by cloud computing, is becoming prevalent. In computation outsourcing, resource-constrained end-users can outsource their computational tasks to the resources abundant cloud for the computational tasks to be completed. In this way, end-users are able to utilize cloud resources in a pay-per-use manner, mitigating the need to invest in local infrastructures.

Along with the tremendous advantages that cloud computing furnishes, security and privacy of the user data become great concerns. This is because once the data is outsourced to the cloud, the end-users will totally lose control of the data. The cloud may try to extract some valuable information from users' outsourced data. Even if the cloud is trustworthy, it could still become a target for adversarial attacks. Thus, it is crucial for applications

to provide security and privacy guarantees for end-users in the mobile cloud computing environment.

Data security, data usability, and computational complexity are among the most important design issues in the mobile computing environment. Often there are trade-offs among these issues.

Trade-off between computational overhead and security: Data security is achieved by primarily utilizing encryption schemes that will introduce extra computational overhead. Often, the higher security guarantees we seek, the more computational power we need to afford. In the mobile computing environment, especially for resource-constrained devices, it is critical to achieve certain security requirements without involving too much computation. More desirably, users can choose how much computation they are willing to devote to meet different security standards, making encryption schemes cost-aware.

Trade-off between security and usability: Data are often gathered and stored in untrusted cloud that may be shared by many people and used at any time. For security and privacy, data is often encrypted, which, however, limits the usability of data. Conventional encryption schemes provide semantic security and allow only users who have the ability to decrypt to utilize the data. However, in many application scenarios, users do not always need to see the data in plaintext in order to fulfill certain functionalities. Examples include statistical aggregation, biometric identification, and information retrieval. It is desirable for certain services to be directly conducted on the encrypted data without disclosing the original data. Data usability may decrease as the level of security protection increases, which provides a trade-off between security and usability.

1.2 Overview

To achieve the trade-offs among data security, usability, and complexity, we mainly utilize two mechanisms: secure computation outsourcing and computing over encrypted data. In particular, we design secure computation outsourcing schemes for various computational problems to ensure data security while limiting local computational overhead, achieving the trade-off between data security and computational complexity. We design some special encryption techniques enabling the cloud to compute directly over the encrypted data. In this way, the end-users can utilize their encrypted data in a secure manner without sacrificing usability. An overview of our proposed schemes is presented as follows.

First, we propose secure outsourcing schemes for various computational problems. Such problems play a fundamental role in various fields such as engineering, finance and cryptography. The secure outsourcing of such problems is of much interest by itself or serves as an important building block in high-level applications. We mainly consider the following computational problems:

1. **Scientific Computation**: we target at general scientific computational problems which cover the scope of linear and non-linear problems such as the system of equations (linear or non-linear), linear programming and convex optimization. Due to the different natures of these problems, it is extremely challenging to design an outsourcing scheme that is suitable for various kinds of computational problems

2. **Cryptographic Computation**: we aim at outsourcing exponential operations in finite groups such as modular exponentiation and scalar multiplication on elliptic curves. Exponentiations are almost ubiquitous in public-key cryptosystems. However, due to large integers involved, exponentiations are considered prohibitively expensive for

resource-constrained devices such as mobile phones.

3. **Attribute Based Encryption**: different from the above fundamental computational problems, we aim at outsourcing an advanced cryptographic protocol named Attribute Based Encryption (ABE), which plays a key role in cryptographic access control. Based on secure outsourcing of ABE, we built up a secure fine-grained access control scheme for mobile users' data through the untrusted cloud.

Second, we propose an encryption scheme that enables any party to conduct specific computations directly over encrypted data. We mainly focus on the computation of vectors that can represent a large variety of data objects. Our proposed encryption scheme can encrypt two vectors such that the inner product of vectors can be evaluated and compared to a pre-defined threshold. During the computation, an adversary cannot derive any key information about the underlying vectors. We apply such an encryption scheme in designing a user-centric biometric authentication system.

1.3 Related Work

1.3.1 Secure Computation Outsourcing

The existing research in secure computation outsourcing has proposed secure outsourcing schemes for various types of computational problems, such as sequence comparison [1–3], linear algebra [4–9], and modular exponentiation [10–12]. While the problems vary, the techniques utilized by these schemes can be divided into two categories: encryption based schemes and disguising based schemes. Researchers from the cryptography community are trying to develop specific encryption schemes under which computation can be carried out on

encrypted data. For instance, in [13] the authors proposed a fully homomorphic encryption scheme under which an arbitrary boolean circuit can be evaluated directly over the encrypted data. Based on this homomorphic encryption and Yao's garbled circuit [14], the authors in [15] designed a secure outsourcing scheme for arbitrary functions where the input and output privacy are protected and the results can be verified in a non-interactive way. However, the main drawback of this type of schemes is that they all require expensive encryption operations thus making it impractical to be carried out in the cloud scenario. Researchers in the theoretic computer science community have developed some disguising techniques to transform various types of computational problems to disguised forms so that the private information of the original problems can be concealed. Based on this idea, the authors in [16] and [4] developed schemes to securely outsource some basic scientific operations such as matrix multiplication, matrix inversion, and convolution. More recently, secure and practical outsourcing schemes were proposed in [6] [17] for linear programming. In [18] [19], the authors focused on outsourcing of large-scale systems of linear equations. However, the disguising techniques discussed above are specially designed for a particular kind of scientific computation, mostly lies in the scope of linear algebra. Thus the application of the proposed schemes is quite limited.

1.3.2 Computing Over Encrypted Data

Computing over encrypted data enables a user to directly operate over encrypted data without first decrypting. In conventional cryptosystem, the decryption process will eventually recover the underlying plaintext m. As a result, all information of m is disclosed. Many applications, however, require only *partially disclosure* of the information of m. For example, a financial organization wants to filter out those customers whose transactions exceed

certain amount. For privacy concern, all the transactions of the customers are encrypted. In this case, instead of decrypting the transactions, a more desirable approach is to determine whether an transaction exceeds certain amount without disclosing the transaction. Such application scenarios motivate the research of functional encryption [20–22]. In a functional encryption scheme, a decryption key S_f is associated with a function f. Given the ciphertext C, the decryption process will evaluate the function $f(m)$, where m is the underlying plaintext. Note that in this process, the plaintext m cannot be recovered. Thus, by issuing different decryption keys S_{f_i}, functional encryption can actually implement *controlled disclosure* of the plaintext m.

A lot of research effort has been devoted to designing various functions f_i for functional encryption schemes. Representative works are Predicate Encryption (PE) [23, 24] and Inner Product Encryption (IPE) [25–28]. In PE, a message is modeled as a vector $\mathbf{x}$ and a decryption key is associated with a vector $\mathbf{y}$. The decryption result is meaningful (otherwise, a random number) if and only if the inner product of $\mathbf{x}$ and $\mathbf{y}$ is equal to 0. Based on this idea, predicates based on exact threshold, polynomial evaluation and set comparison have been realized. In contrast, IPE schemes will recover the value of the inner product of $\mathbf{x}$ and $\mathbf{y}$, without revealing neither $\mathbf{x}$ nor $\mathbf{y}$. In the context of controlled disclosure, IPE could disclose more information of the plaintext than that of PE. This is because, with PE, one can only determine whether the inner product of $\mathbf{x}$ and $\mathbf{y}$ is equal to a certain value or not while with IPE, one can get the value of the inner product. From the above discussion, it becomes clear that if the information is too less, it may not be able to fulfill the functionality of the applications, while more information may violate the user's privacy. Therefore, it is crucial that an encryption scheme should only reveal information necessary for the applications which is a very interesting and challenging research task.

1.4 Summary of Thesis Contributions

The contributions of this thesis can be summarized as follows:

- We proposed secure outsourcing schemes for various computational problems:
  - We proposed a cost-aware secure outsourcing scheme (CASO) that is generally suitable for a wide variety of computational problems, such as system of equations, linear programming and convex optimization.
  - We proposed a secure outsourcing scheme (ExpSOS) for exponential operations in finite groups such as modular exponentiations and scalar multiplications on elliptic curves. Also, we showed that ExpSOS can serve as an important component in building up secure and efficient advanced schemes such as biometric authentication and digital signature.
  - We proposed a secure outsourcing scheme for an advanced cryptographic protocol named Attribute Based Encryption (ABE), which plays a key role in cryptographic access control. Based on secure outsourcing of ABE, we built up a secure fine-grained access control scheme for mobile users' data through untrusted cloud.
- We also introduce a verification process which enables the end-users to verify the validity of the results returned from the cloud servers.
- We investigate the trade-off between the computational complexity and security such that end-users can choose the most suitable outsourcing scheme according to their own resource constraints and security demands.
- We investigate the trade-off between data security and usability by proposing novel methods to directly utilize encrypted data. In particular, we propose an encryption

scheme that enables an untrusted party is able to directly compute over user's encrypted data to extract information which is just adequate for certain applications. We apply such an encryption scheme to a user-centric biometric authentication system that can ensure the privacy of users' biometric templates.

1.5 Thesis Organization

The rest of this thesis is organized as follows. The first three chapters focus on designing secure outsourcing schemes for various computational problems. More specifically, in Chapter 2, we present CASO for secure outsourcing of general computational problems. In Chapter 3, we present ExpSOS for secure outsourcing of exponentiation operations. In Chapter 4, we propose a secure outsourcing scheme of Attribute Based Encryption. In Chapter 5, we present PassbBio, a privacy-preserving user-centric biometric authentication scheme. We conclude in Chapter 6.

Chapter 2

CASO: Cost-Aware Secure Outsourcing of General Computational Problems

2.1 Introduction

In this chapter, we aim at developing a secure outsourcing scheme that is suitable for general computational problems. The challenges come from various aspects. First, we target at general computational problems which cover the scope of linear and non-linear problems such as the system of equations (linear or non-linear), linear programming and convex optimization. Due to the different natures of these problems, it is extremely challenging to design an outsourcing scheme suitable for various kinds of computational problems. Second, in the cloud scenario, the end-users are resource-constrained which means that the operations can be implemented before and after the outsourcing are quite limited. Third, the end-users vary from handheld mobile devices to desktop workstations in terms of resource constraints and security requirements. Thus it is not easy to design a scheme that can meet the requirements of various end-users. Finally, our preliminary investigation shows that a more complex preprocessing of the problem will ensure a more secure outsourcing process. However, it also

creates more computational burden on the end-users. Thus there exists a trade-off between the computational complexity that the end-users can afford and the security they can get in return. All these concerns make it extremely hard to design a secure outsourcing scheme for general computational problems.

To deal with the aforementioned challenges, we propose a secure outsourcing scheme based on affine mappings. The basic idea is that before outsourcing, the independent variables of the computational problem is mapped to a new group of variables through an affine mapping. Correspondingly, the original problem is transformed to a new form that can be securely outsourced to the cloud. Then the cloud can generate valid results from the transformed problem and return the results of the transformed problem back to the end-user. By applying an inverse affine transformation on the results returned from the cloud, the end-user can derive the valid results to the original problem efficiently at the local environment.

It is worth mentioning that the principles of our scheme are different from the previous schemes such as in [18] and [19]. Given a computational problem, the previous schemes try to extract some key parameters that can represent the problem. For example, the key parameters of a linear programming problem are $\{c, \mathbf{A}, \mathbf{b}, \mathbf{D}\}$ as stated in [18]. The main idea is to disguise these key parameters to a different form thus representing a different computational problem. In this way, the information of the original problem is protected from the cloud. While it is relatively easy to extract and disguise the key parameters of a linear computational problem (e.g., linear programming and system of linear equations), it is hard for non-linear problems. This is also the main reason that previous schemes are mainly focused on linear computational problems. In comparison, our scheme starts from the variables since, in essence, a computational problem is about the computation of the variables. We try to map the group of the original variables to another group of variables

in such a way that the secret information is protected. In light of this, we choose the affine mapping according to our design goals. We should notice that the effect of an affine mapping is not only on the variables but also on the problem. Because when we map the variables $\mathbf{x}$ to a new group of variables $\mathbf{y}$ through $\mathbf{x} = \phi(\mathbf{y})$, the original problem changes as $F(\mathbf{x}) = F(\phi(\mathbf{y})) = (F \circ \phi)(y)$, which can naturally be applied to both linear and non-linear problems. We prove that the proposed outsourcing scheme can ensure the security of the private information of the original problem.

The contributions of this chapter can be summarized as follows:

- We propose a cost-aware secure outsourcing scheme (CASO) that is generally suitable for a wide variety of computational problems, such as system of equations, linear programming and convex optimization.

- We investigate the trade-off between the computational complexity and security such that end-users can choose the most suitable outsourcing scheme according to their own resource constraints and security demands.

- Our analysis and performance comparison demonstrate that CASO is much more efficient than the existing schemes with comparable security levels.

- We also introduce a verification process which enables the end-users to verify the validity of the results returned from the cloud servers.

2.2 Problem Statement

2.2.1 System and Threat Model

We consider a system consisting of two entities: the end-user and the cloud. Suppose that an end-user wants to solve a general computational problem denoted by $F(\mathbf{x})$, where $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ is a group of independent variables. Note that $F(\mathbf{x})$ describes a general computational problem not necessarily restricted to a function. For example, it can be a system of equations or an optimization problem. However, due to lack of resources, the end-user needs to outsource the problem to the cloud which is considered to have infinite computing resources. Before outsourcing, the end-user will transform the original problem at the local side in order to prevent information leakage. On receiving the transformed problem, the cloud server will carry out the computing process and return the solution to the end-user. Then at the local side, an inverse transformation is carried out on the solution returned from the cloud to recover the solution of the original problem. Based on the transformation and the information returned by the cloud, the end-user is able to verify the validity of the received solution.

2.2.2 Design Goals

Under the above system and threat model, our proposed outsourcing scheme should achieve the following goals:

1. **Soundness**: Given that the cloud is trustworthy, the transformation on the problem and the inverse transformation of the returned result should guarantee that the recovered solution is correct.

2. **Security**: When the problem is outsourced to the cloud, it should be computationally infeasible for the cloud server to infer the direct information of the original outsourced problem.

3. **Verifiability**: In case that the cloud cannot be fully trusted, the end user should have the ability to verify the validity of the solution returned by the cloud.

4. **Efficiency**: The outsourcing scheme should be efficient in computation and communication. For computation, the overhead caused by the problem transformation, the inverse transformation and the result verification should be limited to $\mathcal{O}(n^2)$. For communication, the overhead caused by the outsourcing process should be in the same level as that of outsourcing the original problem.

5. **Cost-Awareness**: The end-users can select different outsourcing strategies according to their own computational constraints and security demands in a cost-aware manner.

2.2.3 Application Scenarios

Our secure outsourcing scheme serves as an important building block in various high-level applications, since we focus on general computational problems serving as the underlying mathematical models in many practical problems. For example, consider a cloud-assisted image reconstruction system, where some image sensors will upload compressed image samples to the cloud. The cloud will store the image samples and help to reconstruct the image. We note that the core process in image reconstruction can be modeled as a linear programming problem [29]. To preserve the privacy of the images, the sensors can transform the image samples following the procedure in our secure outsourcing scheme. Then, the cloud will help to solve the transformed linear programming problem and returned the disguised

result to the data users. At last, the data users can easily reconstruct the image based on the returned result.

2.3 Secure Outsourcing Based on Affine Mapping

2.3.1 Basic Framework

As mentioned previously, we assume that the end user has a general computational problem $F(\mathbf{x})$ to be solved. Due to the lack of resources, the end user needs to outsource $F(\mathbf{x})$ to the cloud. We formally divide the outsourcing process into the following phases.

1. **Key Generation**: $\mathsf{KeyGen}(\lambda) \rightarrow \mathbf{S}$. In this phase, the end-user generates the secret key $\mathbf{S}$ based on the security parameter λ.

2. **Problem Transformation**: $\mathsf{ProbTran}(\mathbf{S}, F(\mathbf{x})) \rightarrow G(\mathbf{y})$. Based on this secret key $\mathbf{S}$, the end-user transforms $F(\mathbf{x})$ to a new form $G(\mathbf{y})$, where $\mathbf{y}$ is the new input.

3. **Cloud Computation**: $\mathsf{CloudCom}(G(\mathbf{y})) \rightarrow \{\mathbf{y}^*, \Phi\}$. On receiving the transformed problem $G(\mathbf{y})$, the cloud carries out the necessary computation and gives the solution $\mathbf{y}^*$ as well as a proof Φ of the validity of the returned solution.

4. **Result Recovery and Verification**: $\mathsf{RecVeri}(\mathbf{y}^*, \mathbf{S}, \Phi) \rightarrow \{\mathbf{x}^*, \Lambda\}$. By utilizing the secret key $\mathbf{S}$, the end-user recovers solution $\mathbf{x}^*$ to the original problem from $\mathbf{y}^*$. Based on the proof Φ, the end-user gives the decision $\Lambda = \{\mathsf{Ture}, \mathsf{False}\}$, indicating the validity of $\mathbf{x}^*$.

2.3.2 Security Characterization

In this subsection, we characterize the security of a secure outsourcing scheme.

First, we characterize the information of the problem to be outsourced. For a computational problem $F(\mathbf{x})$, the most sensitive information is the problem itself $F(\cdot)$ and the output $\mathbf{x}^*$. Depending on the types of the computational problem, some other information, such as the zeros and poles, could also be sensitive. In light of this, we define direct information and indirect information of a computational problem as follows.

- **Direct Information**: for a computational problem $F(\mathbf{x})$, the direct information is the problem itself $F(\cdot)$ and the output $\mathbf{x}^*$;
- **Indirect Information**: other information besides direct information is defined as indirect information.

Based on the above characterization of information, we define the security notions for an outsourcing scheme.

Definition 2.1 (Security) *An outsourcing scheme achieves security if for any given set of transformed problems* $\{G(\mathbf{y}_i)\}$ *and the solution* $\{\mathbf{y}_i^*\}$*, it is computationally infeasible for the cloud to recover the direct information.*

In the scenario of computation outsourcing, the cloud is able to observe the transformed problem, which corresponds to the ciphertext in a cryptosystem. In this sense, the cloud is able to conduct the ciphertext-only attack. In the above security definition, we define security for direct information. To measure what indirect information the cloud can learn, we define the notion of privacy as follows. First, we define an experiment to model the attack by the cloud.

Outsourcing Experiment $\mathsf{Exp}_{\mathcal{A},\mathsf{out}}(\lambda)$:

- The adversary $\mathcal{A}$ outputs two computational problems $F_1(\mathbf{x})$ and $F_2(\mathbf{x})$ of the same type.
- The challenger runs $\mathsf{KeyGen}(\lambda)$ to obtain the secret key $\mathbf{S}$.
- The challenger outputs a uniform bit $b \in 0, 1$. It runs $\mathsf{ProbTran}(\mathbf{S}, F_b(\mathbf{x}))$ to obtain the transformed problem $G_b(\mathbf{y})$.
- $\mathcal{A}$ outputs a bit b'.
- The output of the experiment is define as 1 if $b = b'$. Otherwise, the output is 0.

Definition 2.2 (Privacy) *An outsourcing scheme achieves privacy for a given security parameter λ if for any probabilistic polynomial time adversary $\mathcal{A}$, there exists a negligible function* negl *such that*

$$|\mathsf{Pr}(\mathsf{Exp}_{\mathcal{A},\mathsf{out}}(\lambda) = 1) - \frac{1}{2}| \leq \mathsf{negl}(\lambda).$$

It should be made clear that the security notions defined here are different from those for traditional cryptosystems in that the transformation does not depend on a cryptographic algorithm, even though we adopted the notions such as semantic security under ciphertext-only attack in cryptography to describe the privacy of indirect information. This is because the semantic security requires that no key information can be derived from the ciphertext, which resembles our privacy requirement that no indirect information can be learned from the transformed problems.

Security Requirements The basic security is the minimum security that an outsourcing scheme should provide. That is, given the transformed problem, the cloud is unable to recover the original problem and solution (direct information). In contrast, privacy characterizes a stronger notion of security. Under the definition of privacy, the transformed problem should achieve indistinguishability. In other words, based on the transformed problem, the cloud

should not be able to recover any meaningful information (indirect information).

Cost-awareness The achievable privacy of an outsourcing scheme should be determined by the needs of the end-user. That is, in some scenario, an end-user may desire to achieve a strong notion of security; while in many other cases, the end-user may only need security of the direct information. On the other hand, in the practical design of secure outsourcing schemes, a stronger notion of security is achieved at the cost of a higher computational complexity. A cost-aware secure outsourcing scheme should provide an end-user the flexibility to select the most efficient outsourcing scheme that satisfies the end-user's security requirements.

2.3.3 Problem Transformation

The basic idea of problem transformation is to map the independent variables of the problem to a new group of variables such that the original problem is transformed to a new form. To be specific, suppose the original problem is $F(x)$. We assume that $\psi : \mathbb{R}^n \to \mathbb{R}^n$ is a general one-to-one mapping function. Let $\mathbf{x} = \psi(\mathbf{y})$, then $F(\mathbf{x}) = F(\psi(\mathbf{y})) = (F \circ \psi)(\mathbf{y}) = G(\mathbf{y})$. In this way, the original input $\mathbf{x}$ can be transformed to input $\mathbf{y}$ with the relationship determined by the function ψ. Below, we give the equivalence definition of two computational problems.

Definition 2.3 (Equivalence) *Denote a set of computational problems as* $\Omega = \{\Gamma \mid \Gamma : \mathbb{R}^n \to \mathbb{R}^n\}$. *For any* $F \in \Omega$, *if there exists a one-to-one mapping* $\psi : \mathbb{R}^n \to \mathbb{R}^n$ *such that* $F(\mathbf{x}) = F(\psi(\mathbf{y})) = (F \circ \psi)(\mathbf{y}) = G(\mathbf{y})$, *then* F *is said to be equivalent to* G. *We denote it as* $F \sim G$. *The equivalent class of* F *is denoted as* $[F] = \{\Gamma \in \Omega \mid \Gamma \sim F\}$.

Theorem 2.1 *The equivalence relation defined in Definition 2.3 is well-defined.*

Proof We only need to prove that the relation defined in Definition 2.3 is reflexive, symmetric and transitive. First, it is obvious that for every $F \in \Omega$, if we select the one-to-one mapping ψ to be the identity mapping, then we have $F(\mathbf{x}) = F(\psi(\mathbf{y})) = F(\mathbf{y})$. Thus for every $F \in \Omega$, we have $F \sim F$ which demonstrates the property of reflexivity. Second, for $F, G \in \Omega$, if $F \sim G$, then there exists a one-to-one mapping ψ such that $F(\mathbf{x}) = F(\psi(\mathbf{y})) = (F \circ \psi)(\mathbf{y}) = G(\mathbf{y})$, which indicates the existence of an inverse mapping ψ^{-1} such that $G(\mathbf{y}) = (F \circ \psi)(\psi^{-1}(\mathbf{x})) = F(\mathbf{x})$. Thus we have $G \sim F$ and the property of symmetry holds. To prove the property of transitivity, assume that $F, G, H \in \Omega$ such that $F \sim G$ and $G \sim H$. This means that there are two one-to-one mappings ψ and ϕ such that $\mathbf{x} = \psi(\mathbf{y})$, $F(\mathbf{x}) = F(\psi(\mathbf{y})) = G(\mathbf{y})$ and $\mathbf{y} = \phi(\mathbf{z})$, $G(\mathbf{y}) = G(\phi(\mathbf{z})) = H(\mathbf{z})$. Therefore, we have $F(\mathbf{x}) = F(\psi(\mathbf{y})) = F((\psi \circ \phi)(\mathbf{z})) = H(\mathbf{z})$. Since ψ and ϕ are both one-to-one mappings, the mapping $\psi \circ \phi$ is also one-to-one. Thus from the definition we have $F \sim H$ and the equivalence relation is transitive.

The above equivalence definition gives an insight of CASO. Based on a one-to-one mapping ψ, the end-user first transforms the original problem $F(\mathbf{x})$ to an equivalent form $G(\mathbf{y})$ that can be securely outsourced to the cloud. Since the solutions to the two problem satisfy $\mathbf{x}^* = \psi(\mathbf{y}^*)$, the end-user can always recover $\mathbf{x}^*$ from $\mathbf{y}^*$ returned by the cloud. Thus the essence of our proposed scheme lies in finding a proper one-to-one mapping that satisfies the various design goals.

Definition 2.4 *An affine mapping $\psi : \mathbb{R}^n \to \mathbb{R}^n$ is defined as a mapping from $\mathbf{x} \in \mathbb{R}^n$ to $\mathbf{y} \in \mathbb{R}^n$ satisfying $\mathbf{x} = \mathbf{K}\mathbf{y} + \mathbf{r}$, where $\mathbf{K} \in \mathbb{R}^{n \times n}$ is non-singular and $\mathbf{r} \in \mathbb{R}^n$.*

It is clear that as long as $\mathbf{K}$ is non-singular, the affine mapping defined above is a one-to-one mapping. The soundness of our proposed scheme based on affine mapping is guaranteed by

the following theorem.

Theorem 2.2 (Soundness) *Under the affine mapping, the transformed problem is equivalent to the original problem. That is the end-user is guaranteed to be able to recover the valid solution of the original problem from the solution returned by the cloud.*

Proof The proof of soundness follows the definition of equivalence. The affine mapping $\mathbf{x} = \mathbf{K}\mathbf{y} + \mathbf{r}$ is one-to-one as long as $\mathbf{K}$ is non-singular. Thus by definition, $F \sim G$ under this affine mapping. Since the solutions to the two problems satisfy $\mathbf{x}^* = \mathbf{K}\mathbf{y}^* + \mathbf{r}$, given $\mathbf{y}^*$ returned by the cloud, the end-user is able to recover $\mathbf{x}^*$ at the local side.

Remark Our scheme is fundamentally different from the previous schemes, such as [6] and [18]. Given a computational problem, the previous schemes try to extract the key parameters that can represent the problem, and then try to disguise these key parameters to a different form thus representing a different computational problem so that the original problem is protected from the cloud. While it is relatively easy to extract and disguise the key parameters of a linear computational problem (e.g., linear programming and system of linear equations), it is hard for non-linear problems, which limits the previous schemes to only linear problems such as linear programming and systems of linear equations.

In comparison, our scheme starts from the variables since, in essence, a computational problem is about computation of the variables. We map the group of the original variables to another group of variables in such a way that the secret information is protected. When we map the variables $\mathbf{x}$ to a new group of variables $\mathbf{y}$ through $\mathbf{x} = \psi(\mathbf{y})$, the original problem becomes $F(\mathbf{x}) = F(\psi(\mathbf{y})) = (F \circ \psi)(\mathbf{y})$, which can naturally be applied to both linear and non-linear problems.

2.4 Cost-Aware Design for Linear Systems

In this section, we present our cost-aware secure outsourcing scheme for general computational problems. In the region of linear computation, we deploy system of linear equations as a case study to show the principles of our design. Then we show that the proposed CASO can be well extended to linear programming.

2.4.1 Outsourcing Scheme

In the problem transformation phase, the end-user first generates a *random one-time secret key* $\mathbf{S} = \{\mathbf{K}, \mathbf{r}\}$, where $\mathbf{K} \in \mathbb{R}^{n \times n}$ is a non-singular matrix and $\mathbf{r} \in \mathbb{R}^n$. Then $\mathbf{x} = \mathbf{K}\mathbf{y} + \mathbf{r}$ is a one-to-one mapping from $\mathbf{x}$ to $\mathbf{y}$. The key S will be discarded after each use. The randomness of the key selection ensures that it is very unlikely for any key to be reused.

Suppose the computational problem is a system of linear equations $\mathbf{A}\mathbf{x} = \mathbf{b}$, where $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$ and $\mathbf{A}$ is an $n \times n$ non-singular matrix. The function $\mathsf{ProbTran}(\mathbf{S}, F(\mathbf{x})) \rightarrow G(\mathbf{y})$ takes the secret key $\mathbf{S} = \{\mathbf{K}, \mathbf{r}\}$ and the linear system as input and generates the output as $\mathbf{A}\mathbf{K}\mathbf{y} = \mathbf{b} - \mathbf{A}\mathbf{r}$. Denote $\mathbf{A}' = \mathbf{A}\mathbf{K}$ and $\mathbf{b}' = \mathbf{b} - \mathbf{A}\mathbf{r}$ and the system is transformed to $G(\mathbf{y}) : \mathbf{A}'\mathbf{y} = \mathbf{b}'$ which can be outsourced to the cloud.

In the phase of cloud computation, the cloud solves $G(\mathbf{y})$ utilizing the typical methods and returns the solution $\mathbf{y}^*$ to the end-user. Then in the result recovery phase, the end-user recovers the solution to the original system of linear equations as $\mathbf{x}^* = \mathbf{K}\mathbf{y}^* + \mathbf{r}$. In the following sections, we will discuss the detailed design of our secure outsourcing scheme. Some of the key notations are listed in Table 2.1.

Table 2.1: Summary of Key Notations

Symbol	Interpretation
$F(\mathbf{x})$	original problem on variables $\mathbf{x}$
$G(\mathbf{y})$	transformed problem on variables $\mathbf{y}$
n	number of independent variables
$\mathbf{K}$, $\mathbf{r}$	one-time transformation key
$\mathbf{A}$	coefficient matrix
$\mathbf{A}'$	transformed coefficient matrix
W	bandwidth of a band matrix
θ	upper bound of non-zeros in each row or column of a sparse matrix
N	number of terms in a non-linear system
L	number of polynomials in a non-linear system
$\mathcal{T}_e$	user-side computational time with outsourcing
$\mathcal{T}_s$	user-side computational time without outsourcing
$\mathcal{I}$	computational gain from outsourcing

2.4.2 Design Analysis

From the above outsourcing scheme, we can see that the computational overhead for the end-user incurs both in the problem transformation and the result recovery phase. To be more specific, in the problem transformation phase, the end-user needs to calculate $\mathbf{AK}$ and $\mathbf{Ar}$. To recover the original solution $\mathbf{x}^*$ from the received solution $\mathbf{y}^*$, the end-user has to calculate $\mathbf{Ky}^*$. Among those operations, the matrix multiplication $\mathbf{AK}$ is the most computationally

expensive one. Thus in our discussion, we will analyze the number of multiplications M required to compute $\mathbf{AK}$. In the following analysis, we denote $\mathbf{A} = \{a_{ij}|i, j = 1, 2, \cdots, n\}$ and $\mathbf{K} = \{k_{ij}|i, j = 1, 2, \cdots, n\}$.

To multiply two arbitrary $n \times n$ matrices, the typical complexity is $\mathcal{O}(n^3)$, which is generally believed to be too high and unacceptable for mobile client computation. However, in our design, we can actually control the complexity by selecting matrix $\mathbf{K}$ properly so that the computational complexity can be effectively reduced without compromising security. Since matrix multiplication is the most expensive part of the end-user's processing, our goal is to ensure that the complexity of multiplying $\mathbf{K}$ with an arbitrary matrix $\mathbf{A}$ is bounded by $\mathcal{O}(n^2)$, which is within the end-user's computational constraints.

In the following sections, we provide four schemes with different types of non-singular secret key $\mathbf{K}$ based on the above-described complexity constraints.

2.4.2.1 K is a Diagonal Matrix (Scheme-1)

A diagonal matrix $\mathbf{K}$ has the format $\mathbf{K} = \{k_{ij}|k_{ij} = 0, \forall i \neq j\}$. Since $\mathbf{K}$ must be non-singular, all the entries in the diagonal have to be non-zero numbers. When $\mathbf{K}$ is a diagonal matrix, we have $M = n^2$.

2.4.2.2 K is a Permutation Matrix (Scheme-2)

A permutation matrix $\mathbf{K}$ has exactly one non-zero entry in each row and each column in the matrix. When $\mathbf{K}$ is a permutation matrix, we have $M = n^2$.

2.4.2.3 K is a Band Matrix (Scheme-3)

Suppose the band matrix $\mathbf{K}$ has an upper half-bandwidth p and a lower half-bandwidth q such that $k_{ij} = 0$ for $i > j + p$ and $j > i + q$. The total bandwidth of $\mathbf{K}$ is denoted by $W = p + q + 1$. When $\mathbf{K}$ is a band matrix, for simplicity, we assume that $\mathbf{K}$ has an equal upper and lower half-bandwidth $p = q = \omega$, then $W = 2\omega + 1$, and the number of multiplications M can be calculated as $M = (2\omega + 1)n^2 - (\omega^2 + \omega)n$.

2.4.2.4 K is a sparse matrix (Scheme-4)

Suppose $\mathbf{K}$ is a sparse matrix. The density d is defined as the ratio of non-zero elements in the matrix. We assume that the number of non-zero elements in each row and each column of $\mathbf{K}$ is up-bounded by a constant θ. When $\mathbf{K}$ is a sparse matrix, it is usually stored in a special manner such as Dictionary of Keys (DOK) [30] in computation. Thus the complexity of matrix multiplication can be approximately measured by the number of non-zero elements, which is dn^3 in our discussion. Since we have assumed that $d \leq \frac{\theta}{n}$, the number of multiplication becomes $M = \theta n^2$.

In summary, through the above analysis, we demonstrate that for the four proposed schemes, the complexity of multiplying $\mathbf{K}$ with an arbitrary matrix $\mathbf{A}$ is $\mathcal{O}(n^2)$. Since matrix multiplication is the most expensive part of the end-user's processing, we can derive that the overall computational complexity for the end-user is $\mathcal{O}(n^2)$, which is within the end-user's computational constraints.

2.4.3 Security Analysis

In this section, we will analyze the security of our proposed CASO. We will focus on the security of the coefficient matrix $\mathbf{A}$ of the original function $F(\mathbf{x})$, the variable $\mathbf{x}$ in the function $F(\mathbf{x})$ and the form of the function $F(\mathbf{x})$.

Theorem 2.3 *CASO can ensure security of the direct information. In other words, for the four schemes in CASO, it is computationally infeasible for the cloud to recover the original coefficient matrix A and the output $\mathbf{x}^*$ for the system of linear equations.*

Proof For a system of linear equations $\mathbf{Ax} = \mathbf{b}$, the original problem is represented by the matrix $\mathbf{A}$ and the vector $\mathbf{b}$. The output is $\mathbf{x}^*$, which is the solution of the system. Under the affine mapping, the system of equations is transformed to $\mathbf{A'y} = \mathbf{b'}$, where $\mathbf{A'} = \mathbf{AK}$ and $\mathbf{b'} = \mathbf{b} - \mathbf{Ar}$. Therefore, it is computationally infeasible for the cloud to recover $\mathbf{A}$ and $\mathbf{b}$ from $\mathbf{A'}$ and $\mathbf{b'}$ since both $\mathbf{K}$ and $\mathbf{r}$ are only used once and kept secret at the local side. Additionally, since the original solution is recovered by $\mathbf{x}^* = \mathbf{Ky}^* + \mathbf{r}$, without knowing $\mathbf{K}$ and $\mathbf{r}$, the cloud cannot recover $\mathbf{x}^*$. In this way, the output of the system is concealed. Thus, all the four schemes are secure in outsourcing the system of linear equations.

Theorem 2.4 *CASO can achieve the privacy of output $\mathbf{x}^*$.*

Proof From the definition of privacy, an end-user plays the role of the challenger and generates the secret key $(\mathbf{K}, \mathbf{r})$. An adversary $\mathcal{A}$ submits two outputs $\mathbf{y}_0$ and $\mathbf{y}_1$ to the end-user, The end-user generates a random bit b and transforms $\mathbf{y}_b$ to $\mathbf{x}_b = \mathbf{Ky}_b + \mathbf{r}$ and sends $\mathbf{x}_b$ back to the adversary. The task of the adversary is to output another bit b'. If $|\mathsf{Prob}(b = b') - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$, the adversary $\mathcal{A}$ will loose the game and it is proved that CASO can achieve the privacy of output $\mathbf{x}^*$. Note that $\mathbf{r}$ is randomly generated. As a

result, regardless of the selection of $\mathbf{K}$, $\mathbf{x}_b = \mathbf{K}\mathbf{y}_b + \mathbf{r}$ is random. Thus, the advantage for the adversary to distinguish $\mathbf{x}_0$ and $\mathbf{x}_1$ is negligible. In other words, the adversary can only generate a bit b' such that $|\mathsf{Prob}(b = b') - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$.

It is worth to mention that all the four schemes in CASO can successfully conceal the zeros and poles of the function since zeros and poles are information of the variables $\mathbf{x}$.

Remark The complete privacy of the coefficient matrix $\mathbf{A}$ is unachievable under affine mapping. This is because the adversary can always distinguish $\mathbf{A}'_0 = \mathbf{A}_0\mathbf{K}$ from $\mathbf{A}'_1 = \mathbf{A}_1\mathbf{K}$. For example, the adversary can select $\mathbf{A}_0$ and $\mathbf{A}_1$ such that one of these two matrix is singular. Then the rank of the retuning matrix $\mathbf{A}'_b$ would be different.

To this end, we have shown that the four schemes in CASO is able to achieve security and the privacy of the output $\mathbf{x}^*$. However, the privacy information of the coefficient matrix $\mathbf{A}$ s not fully achievable. In the following analysis, we will show different protection of indirect information provided by the four schemes in CASO.

Theorem 2.5 *Suppose ψ is a rational mapping, meaning that ψ can be represented as a quotient of two polynomial functions, $G = F \circ \psi$, then we have the following results:*

1. *If F is a rational function, then G is rational.*

2. *If F is an irrational function, then G is irrational.*

Proof Since ψ is a rational mapping, we assume $\psi(x) = \frac{P(x)}{Q(x)}$, where $P(x)$ and $Q(x)$ are polynomials. When F is a rational function, suppose

$$F(x) = \frac{f_1(x)}{f_2(x)}, \tag{2.1}$$

where $f_1(x) = a_0 + a_1 x + \cdots + a_n x^n$, and $f_2(x) = b_0 + b_1 x + \cdots + b_m x^m$. Then

$$(F \circ \psi)(x) = \frac{f_1(\psi(x))}{f_2(\psi(x))}. \tag{2.2}$$

Without loss of generality, we assume that $m > n$. Then we have

$$(F \circ \psi)(x) = \frac{Q^m(x) \cdot f_1(\psi(x))}{Q^m(x) \cdot f_2(\psi(x))}. \tag{2.3}$$

It is clear that both $Q^m(x) \cdot f_1(\psi(x))$ and $Q^m(x) \cdot f_2(\psi(x))$ are polynomials. Therefore, $F \circ \psi$ is the quotient of two polynomials and the composition $G = F \circ \psi$ is a rational function.

When F is irrational, the composition $G = F \circ \psi$ cannot be rational. Otherwise, there exists an inverse rational function ψ^{-1} such that $F = G \circ \psi^{-1} = F \circ \psi \circ \psi^{-1}$ becomes rational. Hence, $G = F \circ \psi$ is irrational when F is irrational.

Since the proposed affine mapping is rational, we have the following corollary.

Corollary 2.1 *Under an affine mapping ψ, the rationality of the function G is the same as the original function F.*

Theorem 2.5 and Corollary 2.1 state that the rationality of the function F cannot be changed through composition with a rational mapping or an affine mapping ψ. That is, if the function F is rational, after the composition $G = F \circ \psi$, the transformed function G is still rational. If F is irrational, G is still irrational. As a result, the side information that is related to the specific form of the function F (e.g., $\sin(\cdot)$ or $\log(\cdot)$) may not be fully concealed by an affine mapping or even a rational mapping.

Now, we will analyze the indirect information that can be revealed by the coefficient matrix $\mathbf{A}$ of the four schemes. Under an affine mapping, the coefficient matrix $\mathbf{A}$ is transformed

to $\mathbf{A'} = \mathbf{AK}$. Thus the problem is to characterize the indirect information of $\mathbf{A}$ given $\mathbf{A'}$. Let a_{ij},a'_{ij} and k_{ij} be the entries of $\mathbf{A}$, $\mathbf{A'}$ and $\mathbf{K}$, respectively. By affine mapping, the entries a'_{ij} are actually linear combinations of a_{ij} and k_{ij}. In our settings, we elaborately select some of the entries in $\mathbf{K}$ to be zeros to reduce the computational complexity. Thus, in a high-level view, multiplying $\mathbf{A}$ with $\mathbf{K}$ results in the combined effect of scaling and permuting of the columns of $\mathbf{A}$. In light of this, to characterize the effect of scaling and permuting, we introduce the *ratio privacy* concerning the ratio information of the entries of $\mathbf{A}$ and the *position privacy* concerning the composition of each entry of $\mathbf{A'}$ from entries of $\mathbf{A}$. In the following, we will analyze to what extent the four schemes can achieve the privacy. For scheme-i ($i = 1, 2, 3, 4$), we denote the secret key utilized in the scheme as $\mathbf{K}_i$.

For scheme-1, the secret key $\mathbf{K}$ is a diagonal matrix denoted by $\mathbf{K}_1 = \{k_{ij} | k_{ij} = 0, \forall i \neq j\}$. The entry a'_{ij} in $\mathbf{A'}$ can be calculated as $a'_{ij} = k_{ii}a_{ij}$. By investigating $\mathbf{A'}$, it is obvious that each column in $\mathbf{A'}$ is related in a simple way to that in $\mathbf{A}$ such that the i^{th} column in $\mathbf{A'}$ is the multiplication of the i^{th} column in $\mathbf{A}$ with k_{ii}. In this way, only based on $\mathbf{A'}$, the cloud can easily know the ratio between any two entries within the same column in $\mathbf{A}$. Moreover, it is also clear how each entry in $\mathbf{A'}$ is composed.

For $\mathbf{K}_2$ to be a permutation matrix in scheme-2, the difference is that $\mathbf{A'}$ in scheme-2 can be regarded as the result of permuting the columns of $\mathbf{A'}$ obtained from scheme-1. Thus, although the cloud can get a knowledge of the ratio between two entries in the same column of $\mathbf{A}$, it is not sure which particular column those two entries belong to. As a result, while scheme-2 can achieve position privacy, it may leak ratio privacy.

In scheme-3, for $\mathbf{K}_3$ to be a band matrix with upper half-bandwidth and lower half-

bandwidth both equal ω, it can be calculated that

$$a'_{ij} = \sum_{r=j-\omega}^{j+\omega} a_{ir} k_{rj}. \tag{2.4}$$

Since each entry in $\mathbf{A'}$ is a linear combination of entries in $\mathbf{A}$ and $\mathbf{K}$, the ratio information of entries in $\mathbf{A}$ is concealed. However, the disadvantage is that the cloud can still learn how a particular entry in $\mathbf{A'}$ is composed. For example, suppose $\omega = 1$, the cloud can know for sure that $a'_{ij} = a_{i(j-1)}k_{(j-1)j} + a_{ij}k_{jj} + a_{i(j+1)}k_{(j+1)j}$. In this sense, while scheme-3 can achieve ratio privacy, it may leak position privacy.

At last, for $\mathbf{K}_4$ to be a sparse matrix in scheme-4, we assume that there are exactly θ non-zero entries in each row and column of $\mathbf{K}$. Similar to scheme-3, the ratio information of entries in $\mathbf{A}$ can be concealed. Moreover, since the non-zero entries are randomly positioned in the sparse matrix $\mathbf{K}$, the cloud is unable to know how each entry in $\mathbf{A'}$ is composed. Thus, scheme-4 can achieve both ratio privacy and position privacy.

In summary, we categorized the privacy of the coefficient matrix $\mathbf{A}$ into ratio privacy and position privacy. Such categorization stems from the essence of matrix multiplication. In a high-level view, multiplying $\mathbf{A}$ with a specially designed secret matrix $\mathbf{K}$ can be separated into two critical operations: the weighted sum of the entries of $\mathbf{A}$ and random permutation. The former operation preserves the ratio privacy while the latter operation preserves the position privacy. Moreover, the number of non-zero entries in $\mathbf{K}$ determines to what degree the ratio privacy is preserved. However, as long as the positions of the non-zeros entries are random, the position privacy of $\mathbf{A}$ are preserved. We summarize the computational complexity and security of CASO in Table 2.2.

Table 2.2: Complexity and security of each scheme : ✓ denotes security can be guaranteed or privacy can be preserved. × denotes privacy cannot be preserved.

	Complexity	Security	Privacy of $\mathbf{x}$	Ratio Privacy of $\mathbf{A}$	Position Privacy of $\mathbf{A}$
Scheme-1	n^2	✓	✓	×	×
Scheme-2	n^2	✓	✓	×	✓
Scheme-3	Wn^2	✓	✓	✓	×
Scheme-4	θn^2	✓	✓	✓	✓

2.4.4 Trade-off between Complexity and Security

From the above complexity and security analysis, we can see that there is a trade-off between the computational complexity and security. As the simple scheme, scheme-1 is able to protect the original coefficient matrix while exposing the ratio between any two entries in the same column. In comparison, scheme-2 is slightly more expensive (e.g. the positions of the non-zero entries have to be stored), but it is this cost for non-zero entries' random positions that makes it effective to conceal the ratio information. The complexity of scheme-3 and scheme-4 is linearly dependent on W and θ, respectively. They are more costly than scheme-1 and scheme-2. However, the transformed matrix $\mathbf{A}'$ can conceal $\mathbf{A}$ and $\mathbf{K}$ in a more complex way since it can conceal the structure of the coefficient matrix. In summary, from scheme-1 to scheme-4, the security levels that they can provide increase at a cost of computational power.

In the context of cloud computing, the end-users vary from mobile devices to powerful workstations thus having different computational constraints as well as different security demands. Thus CASO provides end-users with the flexibility to choose the outsourcing schemes that are most suitable for them. These four schemes give cost-aware outsourcing for end-users to address the various security demands and computational constraints.

2.4.5 Application to Linear Programming

In this section, we will demonstrate that our design and analysis for system of linear equations can be well applied to many computational problems, such as linear programming. We consider a linear programming problem denoted by

$$F(\mathbf{x}) := \begin{cases} \text{minimize} & \mathbf{c}^T\mathbf{x} \\ \text{subject to} & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{D}\mathbf{x} \geq 0, \end{cases} \tag{2.5}$$

where $\mathbf{b}, \mathbf{c} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\mathbf{D} \in \mathbb{R}^{s\times n}$ $(m, s \leq n)$.

Under the affine mapping $\mathbf{x} = \mathbf{K}\mathbf{y} + \mathbf{r}$, the problem is transformed to

$$G(\mathbf{y}) := \begin{cases} \text{minimize} & \mathbf{c}^T\mathbf{K}\mathbf{y} + \mathbf{c}^T\mathbf{r} \\ \text{subject to} & \mathbf{A}\mathbf{K}\mathbf{y} = \mathbf{b} - \mathbf{A}\mathbf{r} \\ & \mathbf{D}\mathbf{K}\mathbf{y} \geq -\mathbf{D}\mathbf{r}, \end{cases} \tag{2.6}$$

from which we can see that the original coefficient matrix can be concealed by the secret key $\mathbf{K}$ and $\mathbf{r}$. It is obvious that the computational bottleneck lies in the multiplication of $\mathbf{K}$ with $\mathbf{A}$ and $\mathbf{D}$. Thus the same complexity and security analysis for systems of linear equations applies for linear programming. That is the complexity of the previous four schemes is all bounded by $\mathcal{O}(n^2)$. In terms of security, the four schemes are all secure in protecting the original coefficient matrix while providing different levels of protection of the side information.

In the next section, we explore the differences for non-linear computation by investigating

system of non-linear equations and convex optimization problems.

2.5 Extension to Non-linear Systems

In this section, we aim at exploring the different design issues between linear and non-linear computation. We consider a system of non-linear equations denoted by $F(\mathbf{x}) = 0$, where $F(\mathbf{x}) = \{f_i(\mathbf{x})|f_i(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}, i = 1, 2, \cdots, n\}$. Typically, it is hard to obtain a symbolic solution for the system. Thus the normal method is to solve the system of equations numerically in an iterative way. The main idea is that given a solution $\mathbf{x}_k$ in the k^{th} iteration, we need to solve the linear system $\partial F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_k}(\mathbf{x}_{k+1} - \mathbf{x}_k) = -F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_k}$, where $\partial F(\mathbf{x})$ is the Jacob matrix of $F(\mathbf{x})$. Then we can obtain the solution $\mathbf{x}_{k+1}$ in the $(k+1)^{th}$ iteration. The iteration will terminate when $\|F(\mathbf{x}^*)\| < \varepsilon$, where ε is the error tolerance and $\mathbf{x}^*$ is the final solution. To minimize the communication overhead and the energy consumption of the end-users, our goal is to design off-line scheme so that the end-users are not required to interact with the cloud except the problem outsourcing and result retrieving process. In this way, the end-users only need to focus on the high-level view of the problem without knowing the details of problem-solving process. The detailed design and analysis of the outsourcing scheme are presented as follows.

2.5.1 Outsourcing Scheme

Compared with the outsourcing of the system of linear equations, the main difference lies in the problem transformation phase. First, to start the iteration at the cloud side, an initial guess of the solution should also be outsourced. We assume that at the local side, the end-user generates an initial solution $\mathbf{x}_0$. Then with the affine mapping, the outsourced initial

solution becomes $\mathbf{y}_0 = \mathbf{K}^{-1}(\mathbf{x}_0 - \mathbf{r})$. We should notice that there is an inversion operation on $\mathbf{K}$ which will impose more constraints on our selection of $\mathbf{K}$ in terms of computational complexity. Second, after substituting $\mathbf{x}$ with $\mathbf{y}$, the problem should be further transformed. We use a simple example to illustrate this point. Suppose we want to solve a system of non-linear equations

$$F(\mathbf{x}) := \begin{cases} \sin(3x_1) + 4x_2^2 + x_2x_3 = 0 \\ 2x_1 + e^{3x_2} + 2x_3^3 = 0 \\ \lg(5x_1) + \frac{1}{2x_2+1} + 3(x_3+1)^2 = 0. \end{cases} \tag{2.7}$$

We take the affine mapping $\mathbf{x} = \mathbf{K}\mathbf{y} + \mathbf{r}$, where $\mathbf{r} = 0$ and

$$\mathbf{K} = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{bmatrix}.$$

Then the system is transformed to

$$G(\mathbf{y}) := \begin{cases} \sin(9y_1) + 16y_2^2 + 8y_2y_3 = 0 \\ 6y_1 + e^{6y_2} + 128y_3^3 = 0 \\ \lg(15y_1) + \frac{1}{4y_2+1} + 48y_3^2 + 24y_3 = -3. \end{cases} \tag{2.8}$$

It is obvious that to protect the cloud from revealing information from the transformed system, it is sufficient to mix the coefficient of each term in the equations with the key entry. To be specific, we assume that there are π_i terms in equation $f_i(\mathbf{x})$ and each term is denoted

by $f_i^j(\mathbf{tx})$, where $\mathbf{t}$ is the coefficient. Then each equation in the system can be written as

$$f_i(\mathbf{x}) = \sum_{j=1}^{\pi_i} f_i^j(\mathbf{tx}).$$

Under the affine mapping $\mathbf{x} = \mathbf{Ky} + \mathbf{r}$, $f_i^j(\mathbf{tx})$ is transformed to

$$g_i(\mathbf{y}) = f_i(\mathbf{Ky} + \mathbf{r}) = \sum_{j=1}^{\pi_i} f_i^j(\mathbf{t}(\mathbf{Ky} + \mathbf{r})).$$

Thus the coefficient $\mathbf{t}$ is concealed by $\mathbf{K}$ and $\mathbf{r}$, which is similar to the case of the system of linear equations. However, as illustrated in the example, the multiplication cannot be simply carried out when $f_i^j(\cdot)$ is a polynomial. Thus a further transformation is needed to mix $\mathbf{t}$ with $\mathbf{K}$ and $\mathbf{r}$ for polynomials.

Without loss of generality, we assume that the polynomial is denoted by $t_i x_i^m$ and in the affine mapping, $\mathbf{K}$ is a band matrix with bandwidth $W = 3$ and $\mathbf{r} = 0$. Thus under the affine mapping, the polynomial is transformed to

$$t_i(k_{i-1}y_{i-1} + k_i y_i + k_{i+1}y_{i+1})^m.$$

To mix the coefficient t_i with the secret keys, one straightforward way is to expand the polynomial and then multiple it with t_i. However, the complexity is unacceptable for high order polynomials. Instead, we propose that it is sufficient to split the secret keys as $k_s = pq_s$,

where $s = i-1, i, i+1$ such that

$$\begin{aligned} & t_i(k_{i-1}y_{i-1} + k_i y_i + k_{i+1}y_{i+1})^m \\ = \; & t_i(pq_{i-1}y_{i-1} + pq_i y_i + pq_{i+1}y_{i+1})^m \\ = \; & t_i p^m (q_{i-1}y_{i-1} + q_i y_i + q_{i+1}y_{i+1})^m. \end{aligned}$$

In this way, the coefficient t_i in the original function and the secret keys k_i are concealed.

2.5.2 Complexity Analysis

From the analysis above, we can see that the complexity of the problem transformation mainly depends on two aspects. One is the specific form of the equations, that is the number of polynomials in the equations. The other one is how $\mathbf{x}$ and $\mathbf{y}$ are related, which is determined by the number of non-zero entries in $\mathbf{K}$.

For a given system of non-linear equations, suppose that there are N terms in total in the systems, among which L are polynomials with orders no greater than m. Assume that the number of non-zero entries in $\mathbf{K}$ is up-bounded by λ (i. e. each x is substituted by at most λ y's). Thus for each non-polynomial term, the transformation takes λ multiplications between the coefficient of the term and the key entries. And for a polynomial term $t_i x_i^m$, we assume that it is replaced by

$$\begin{aligned} t_i(k_1 y_1 + \cdots + k_\lambda y_\lambda)^m &= t_i(pq_1 y_1 + \cdots + pq_\lambda y_\lambda)^m \\ &= t_i p^m (q_1 y_1 + \cdots + q_\lambda y_\lambda)^m. \end{aligned}$$

Then the operations involved in the transformation include one multiplication, λ division

Table 2.3: Complexity for system of non-linear equations

Scheme	Complexity
Diagonal matrix	$N + (\log^2 m + 1)L$
Permutation matrix	$N + (\log^2 m + 1)L$
Band matrix	$WN + (\log^2 m + 1)L$
Sparse matrix	$\theta N + (\log^2 m + 1)L$

and raising p to the power of m. As stated previously, we utilize the number of multiplication as a measurement of complexity. We assume that in terms of computational complexity, one division is equal to one multiplication and with the method of exponentiation by squaring, the computation for m^{th} power takes $\log^2 m$ multiplications. Thus, for a system of non-linear equations with N terms among which L are polynomials, the complexity can be calculated as

$$\lambda N + (\log^2 m + 1)L.$$

It is obvious that the complexity depends on λ which is further determined by the selection of $\mathbf{K}$. We summarize the complexity of the four different types of matrices in Table 2.3. We can see from the table that the complexities of all schemes are constrained to $\mathcal{O}(N)$, where N is the number of terms in the system of non-linear equations. Notice that typically for a system of equations, the number of terms N is in the level of n^2, where n is the number of independent variables. Thus the complexity is still bounded by $\mathcal{O}(n^2)$, which fulfills our design goals.

2.5.3 Security Analysis

Similar to the security analysis for linear systems, all of the proposed four schemes are secure in protecting the coefficient matrix, the zeros, poles and optimums of the outsourced problem. As stated in Corollary 2.1, CASO cannot conceal the specific form of the functions.

For instance, in the example given in Section 2.5.1, the original system of equations is transformed to $G(\mathbf{y})$ such that the coefficients in each term of the function are changed. However, the specific forms of the function (e.g., $\sin(\cdot)$, $\lg(\cdot)$, etc.) remain unchanged.

For the four schemes, generally as the complexity increases, more side information can be concealed from the cloud. Different from the linear equations, a non-linear function $f_i(x)$ may contain some side information, such as maximum or minimum value which is important in some applications. For instance, the plot of the function or the extreme values may expose the distribution of the incidence of a disease among different age groups. For scheme-1 and scheme-2, the curve of the function is just a scaled version. Though scheme-2 provides better protection since it can conceal the independent variables. In scheme-3 and scheme-4, each independent variables in the original problem is substituted by several new variables. Thus the side information, such as the curve and the extreme values can be perfectly concealed.

2.5.4 Application to Convex Optimization

In this section, we show that the above schemes and analysis can also be applied to convex optimization. Convex optimization is widely employed in various practical problems. We consider a convex optimization problem denoted by

$$F(\mathbf{x}) := \begin{cases} \text{minimize} & f_0(\mathbf{x}) \\ \text{subject to} & f_i(\mathbf{x}) \leq 0, i = 1, \cdots, m \\ & h_j(\mathbf{x}) = 0, j = 1, \cdots, t, \end{cases} \tag{2.9}$$

where $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 0, ..., m$ and $h_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, ..., t$ are all convex functions. Under the affine mapping $\mathbf{x} = \mathbf{K}\mathbf{y} + \mathbf{r}$, the original problem $F(\mathbf{x})$ is transformed to

$$G(\mathbf{y}) := \begin{cases} \text{minimize} & f_0(\mathbf{K}\mathbf{y} + \mathbf{r}) \\ \text{subject to} & f_i(\mathbf{K}\mathbf{y} + \mathbf{r}) \leq 0, i = 1, \cdots, m \\ & h_j(\mathbf{K}\mathbf{y} + \mathbf{r}) = 0, j = 1, \cdots, t. \end{cases} \tag{2.10}$$

Since the key matrix $\mathbf{K}$ and $\mathbf{r}$ are randomly generated and kept secret at the local side, the coefficient matrix of the outsourced problem is perfectly protected. And because the functions $f_i(\cdot)$ and $h_j(\cdot)$ are all non-linear functions, the security and the complexity analysis of system of non-linear equations can be well applied in this case. Thus we conclude that our outsourcing scheme is also applicable to convex optimization problems.

2.6 Results Verification

The general idea of our proposed verification scheme is to transform the problem with two independent affine mappings and outsource the two transformed problems to the cloud. Then the end-user is able to verify whether the two results returned by the cloud match with each other. We note that such a verification scheme is different from those that require two rounds of communications. In our scheme, the end-user does not need to wait for the result of the first round outsourcing before sending out the second transformed problem. This is because the results for the two transformed problems are received simultaneously. To be specific, under the affine mappings $\mathbf{x} = \mathbf{K}_1\mathbf{y} + \mathbf{r}_1$ and $\mathbf{x} = \mathbf{K}_2\mathbf{z} + \mathbf{r}_2$, the original problem $F(\mathbf{x})$ is transformed to $G(\mathbf{y})$ and $H(\mathbf{z})$ which are outsourced to the cloud. Then the cloud solves

the two outsourced problems and returns the corresponding results $\mathbf{y}^*$ and $\mathbf{z}^*$. Since the condition $\mathbf{K}_1\mathbf{y}^* + \mathbf{r}_1 = \mathbf{K}_2\mathbf{z}^* + \mathbf{r}_2$ holds for these two results, the end-users can utilize it as a criterion to verify whether the returned results are valid.

2.6.1 System of Non-Linear Equations

The idea introduced above can be applied to system of equations directly. When $F(\mathbf{x})$ is a system of linear equations, it is sufficient to verify directly whether $\|\mathbf{A}\mathbf{x}^*\| < \varepsilon$, where $\|\cdot\|$ denotes the Euclidean norm of a vector and ε is a pre-defined error tolerance. The complexity of this verification process is $\mathcal{O}(n^2)$.

When $F(\mathbf{x})$ is a system of non-linear equations, since the end-user will have to evaluate the non-linear functions, the computational cost for direct verification generally exceeds $\mathcal{O}(n^2)$. However, based on our idea of outsourcing twice, the end-user only needs to check the condition $\mathbf{K}_1\mathbf{y}^* + \mathbf{r}_1 = \mathbf{K}_2\mathbf{z}^* + \mathbf{r}_2$. Since the verification process involves only linear operations, the computational complexity is bounded by $\mathcal{O}(n^2)$. As the system of equations is typically solved by the iterative method, the solution is not accurate. Thus we may need to change the equality condition to

$$\|(\mathbf{K}_1\mathbf{y}^* + \mathbf{r}_1) - (\mathbf{K}\mathbf{z}^* + \mathbf{r}_2)\| < \varepsilon.$$

In the following analysis, we uniformly utilize the equality condition $\mathbf{K}_1\mathbf{y}^* + \mathbf{r}_1 = \mathbf{K}_2\mathbf{z}^* + \mathbf{r}_2$ as the verification criteria. When the computational problems are solved inaccurately, the equality condition should be changed to its inequality variation.

2.6.2 Optimization Problems

When $F(\mathbf{x})$ is an optimization problem, we utilize convex optimization as an example to illustrate the verification process. And it can be easily applied to other optimization problems, such as linear programming. The output of a convex optimization problem can be divided into three cases: normal, infeasible and unbounded [31, Chapter 4.1]. For the convex optimization problem defined in equation (2.9), the *domain* $\mathcal{D}$ is the set for which the objective function and the constraint functions are defined. That is

$$\mathcal{D} = \bigcap_{i=1}^{m} \mathbf{dom} f_i \cap \bigcap_{i=i}^{t} \mathbf{dom} h_i.$$

The *feasible set* is $\mathcal{E} = \{\mathbf{x} \in \mathcal{D} \,|f_i(\mathbf{x}) \leq 0, i = 1, \cdots, m, h_i(\mathbf{x}) = 0, i = 1, \cdots, t\}$. In the normal case, there exists an optimal point $\mathbf{x}^* \in \mathcal{E}$ such that $f_0(\mathbf{x}^*) \leq f_0(\mathbf{x}), \forall \mathbf{x} \in \mathcal{E}$. In the infeasible case, $\mathcal{E} = \emptyset$. In the unbounded case, there exists points $\mathbf{x}_k \in \mathcal{E}$ such that $f_0(\mathbf{x}_k) \to -\infty$ as $k \to \infty$.

For the cloud to cheat, it must return results in the same case for the two outsourced problems $G(\mathbf{y})$ and $H(\mathbf{z})$ as mentioned above. Suppose that $\mathbf{y}^*$ and $\mathbf{z}^*$ are the two returned results and they belong to the same case. In the following, we will present the verification scheme for the three different cases separately.

2.6.2.1 Normal Case

The above proposed verification scheme works well for the normal case. That is if the equality $\mathbf{K}_1\mathbf{y}^* + \mathbf{r}_1 = \mathbf{K}_2\mathbf{z}^* + \mathbf{r}_2$ holds, the end-user can make sure that a valid result can be recovered. This is because whatever the correct result is (normal, infeasible or unbounded), the cloud is not able to come up with two results that satisfy the equality without actually

conducting the computation process. And this verification process for normal case forms the basis for the verification for other cases.

2.6.2.2 Infeasible Case

The above verification scheme would fail if the cloud simply returns an infeasible result for any outsourced convex optimization problem. To deal with this issue, we utilize phase I method as described in [31, Chapter 11] to check the feasibility of the problem. For a convex optimization problem $F(\mathbf{x})$, a corresponding phase I optimization problem can be constructed as:

$$F_I(\mathbf{x}) := \begin{cases} \text{minimize} & \rho \\ \text{subject to} & f_i(\mathbf{x}) \leq \rho, i = 1, \cdots, m \\ & h_j(\mathbf{x}) = 0, j = 1, \cdots, t \end{cases},$$

where ρ is a single variable. It is obvious that when ρ is large enough, $F_I(\mathbf{x})$ is always feasible.

Suppose $\mathbf{x}^*$ minimizes the objective function and ρ^* is the corresponding minimum value. The phase I problem is designed in such a way that when $\rho^* \leq 0$, the original problem $F(\mathbf{x})$ is feasible and $F(\mathbf{x})$ is infeasible otherwise. Thus the verification scheme for the infeasible case can be designed as follows. When the cloud indicates that the solutions to the two outsourced problems $G(\mathbf{y})$ and $H(\mathbf{z})$ are infeasible, it then generates the corresponding two phase I problems $G_I(\mathbf{y})$ and $H_I(\mathbf{z})$ and computes the optimal points $\mathbf{y}^*$ and $\mathbf{z}^*$ and the minimum values ρ_G^* and ρ_H^*, respectively. Then at the local side, the verification is the same as that in the normal case. That is only when $\rho_G^* > 0$ and $\rho_H^* > 0$ and the equality $\mathbf{K}_1\mathbf{y}^* + \mathbf{r}_1 = \mathbf{K}_2\mathbf{z}^* + \mathbf{r}_2$ holds can the end-user be guaranteed to receive valid solutions.

2.6.2.3 Unbounded Case

In the unbounded case, the cloud indicates that the objective function $f_0(\mathbf{x}) \to -\infty$ in its domain. We utilize duality to verify the soundness of the returned result. For a convex optimization problem, we can construct the corresponding *Lagrangian* L as

$$L(\mathbf{x}, \mathbf{u}, \mathbf{v}) = f_0(\mathbf{x}) + \sum_{i=1}^{m} u_i f_i(\mathbf{x}) + \sum_{j=1}^{t} v_j h_j(\mathbf{x}),$$

where $\mathbf{u} \in \mathbb{R}^m$ and $\mathbf{v} \in \mathbb{R}^t$ are the associated *Lagrange multiplier vectors*. Then based on this Lagrangian $L(\mathbf{x}, \mathbf{u}, \mathbf{v})$, a *Lagrange dual function* can be constructed as

$$\begin{aligned} \Phi(\mathbf{u}, \mathbf{v}) &= \inf_{\mathbf{x} \in \mathcal{D}} L(\mathbf{x}, \mathbf{u}, \mathbf{v}) \\ &= \inf_{\mathbf{x} \in \mathcal{D}} \left(f_0(\mathbf{x}) + \sum_{i=1}^{m} u_i f_i(\mathbf{x}) + \sum_{j=1}^{t} v_j h_j(\mathbf{x}) \right), \end{aligned}$$

where $\mathcal{D}$ is the domain of the optimization problem. From this definition, it is easy to prove that $\forall \mathbf{u} \succeq 0$, we have the following inequality:

$$\Phi(\mathbf{u}, \mathbf{v}) \leq L(\mathbf{x}^*, \mathbf{u}, \mathbf{v}) \leq f_0(\mathbf{x}^*),$$

where $f_0(\mathbf{x}^*)$ denotes the optimal value of the objective function. The above inequality gives a lower bound of the objective function that depends on the selection of $\mathbf{u}$ and $\mathbf{v}$. Thus, among all the selections of $\mathbf{u}$ and $\mathbf{v}$, finding the optimal lower bound is equivalent to solving

the following optimization problem:

$$\begin{cases} \text{maximize} & \Phi(\mathbf{u}, \mathbf{v}) \\ \text{subject to} & \mathbf{u} \succeq 0. \end{cases}$$

The objective function $\Phi(\mathbf{u}, \mathbf{v})$ is concave since it is the point-wise infimum of a series of affine function of $(\mathbf{u}, \mathbf{v})$. Thus the above optimization problem is also a convex optimization problem. If the original problem is unbounded below, the convex optimization problem described above should be infeasible since it gives a lower bound of the optimal value in the original problem. Thus the remaining task is to verify the feasibility of the above convex optimization problem, which has been illustrated in the infeasible case. Let the cloud solve the phase I problems of the two Lagrange dual problems and return the optimal solutions denoted by $(\rho_G^*, \mathbf{y}^*, \mathbf{u}_G^*, \mathbf{v}_G^*)$ and $(\rho_H^*, \mathbf{z}^*, \mathbf{u}_H^*, \mathbf{v}_H^*)$. At the local side, the end-user then checks whether $\rho_G^* > 0$ and $\rho_H^* > 0$ and whether the equality $\mathbf{K}_1\mathbf{y}^* + \mathbf{r}_1 = \mathbf{K}_2\mathbf{z}^* + \mathbf{r}_2$ holds.

2.7 Evaluation

In this section, we will evaluate the performance of the proposed CASO scheme. We first compare CASO with several existing outsourcing schemes. Then we present some numeric results to show the efficiency of CASO.

2.7.1 Performance Comparison

The existing schemes on outsourcing of numeric computation mainly focus on some specific problems. To the best of our knowledge, no effective outsourcing schemes have been proposed

for non-linear problems. In the following part, we compare the performance of our proposed CASO scheme with three existing schemes specially designed for three types of problems in terms of security, computational complexity and communication overhead. To measure the communication overhead, we introduce a *communication overhead index* $\mathcal{I}_c$ which is defined as the fraction of the communication cost of transmitting the original problem over that of the transformed problem. Thus a larger $\mathcal{I}_c$ indicates better communication efficiency.

2.7.1.1 Linear Programming

In this section, we compare CASO for linear programming problems with the schemes proposed in [6] and [17] in both security and complexity. We will show that while achieving the same security level, our scheme outperforms them in terms of complexity. In addition, our scheme also provides end-users with the flexibility to select different outsourcing options with different complexity according to their security demands.

The general linear programming problems can be expressed as

$$\begin{cases} \text{minimize} & \mathbf{c^T x} \\ \text{subject to} & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{Dx} \geq 0. \end{cases} \tag{2.11}$$

In [6], to transform the problem, a secret key $\mathbf{K} = \{\mathbf{Q}, \mathbf{M}, \mathbf{r}, \boldsymbol{\lambda}, \gamma\}$ is generated, where $\mathbf{Q}$ is a randomly generated $m \times m$ non-singular matrix, $\mathbf{M}$ is a randomly generated $n \times n$ non-singular matrix, and $\mathbf{r}$ is an $n \times 1$ vector. With this secret key, the original problem is

transformed to the following problem

$$\begin{cases} \text{minimize} & \mathbf{c'^T x} \\ \text{subject to} & \mathbf{A'x} = \mathbf{b'} \\ & \mathbf{D'x} \geq 0, \end{cases} \tag{2.12}$$

where $\mathbf{A'} = \mathbf{QAM}, \mathbf{D'} = (\mathbf{D} - \lambda\mathbf{QA})\mathbf{M}, \mathbf{b'} = \mathbf{Q}(\mathbf{b} + \mathbf{Ar})$ and $\mathbf{c'} = \gamma\mathbf{M^T c}$. Then the transformed problem is outsourced to the cloud which is similar as our approach.

In terms of computational complexity, the computational overhead of the outsourcing scheme in [6], as well as our scheme, lies primarily in matrix multiplication. As stated in their paper, the overall computational complexity of the scheme proposed in [6] is slightly less than $\mathcal{O}(n^3)$ depending the algorithm chosen to implement matrix multiplication. For instance, when the Strassen algorithm is adopted, the complexity becomes $\mathcal{O}(n^{2.81})$; while for the Coppersmith-Winograd algorithm the complexity is $\mathcal{O}(n^{2.376})$. However, by carefully selecting the secret key $\mathbf{K}$, our scheme can limit the complexity within $\mathcal{O}(n^2)$.

In terms of communication overhead, the original problems in both schemes are transformed by matrix multiplication such that the resulting matrices are still in the same scale. As a result, the communication cost of the original and transformed problems are at the same level. Thus we have $\mathcal{I}_c = 1$ in our scheme and the scheme in [6].

In terms of security, both schemes can conceal the private information by some disguising techniques, that is to disguise the original matrices by multiplying them with some random matrices. As a consequence, the security they can achieve in protecting the original coefficient matrix is at the same level. Since the types of the transformation matrices (e.g. $\mathbf{Q}$, $\mathbf{M}$) are not specified, each entry in the disguised coefficient matrix $\mathbf{A'}$ can be the linear combination

of multiple entries in $\mathbf{A}$ and the transformation matrices. Thus, the ratio information can be concealed. In this sense, the security of the scheme in [6] is comparable with our scheme-4 in terms of protecting side information.

The scheme proposed in [17] can be regarded as a variation of that in [6]. The main difference is that the authors in [17] specify the transformation matrices as sparse matrices in order to achieve a lower computational complexity of $\mathcal{O}(n^2)$. For example, the schemes in [17] disguises the coefficient matrix by matrix multiplication as $\mathbf{A}' = \mathbf{MAN}$, where $\mathbf{M}$ and $\mathbf{N}$ are both sparse matrices. In this way, the complexity is reduced to $\mathcal{O}(n^2)$. Actually, this scheme can be considered as a special case of our proposed CASO where $\mathbf{K}$ is selected as a sparse matrix.

2.7.1.2 System of Linear Equations

In [18], the authors investigated outsourcing of system of linear equations $\mathbf{Ax} = \mathbf{b}$ based on iterative method. First, the problem is transformed to $\mathbf{Ay} = \mathbf{b}'$, where $\mathbf{y} = \mathbf{x} + \mathbf{r}$, $\mathbf{b}' = \mathbf{b} + \mathbf{Ar}$ and $\mathbf{r}$ is a random vector. Then the end-user solves the transformed problem iteratively with the aid of cloud servers and an initial guess $\mathbf{y}_0$ from the following iteration equation:

$$\mathbf{y_{k+1}} = \mathbf{T} \cdot \mathbf{y_k} + \mathbf{c}', \tag{2.13}$$

where $\mathbf{A} = \mathbf{D} + \mathbf{R}$ such that $\mathbf{D}$ is non-singular, $\mathbf{T} = -\mathbf{D}^{-1} \cdot \mathbf{R}$ and $\mathbf{c}' = \mathbf{D}^{-1} \cdot \mathbf{b}'$. The end-user utilizes the cloud servers to compute the most expensive part $\mathbf{T} \cdot \mathbf{y_k}$ based on homomorphic encryption to conceal the private information $\mathbf{T}$. To be specific, the matrix $\mathbf{T}$ is pre-computed at the local side and the encrypted version $\mathsf{Enc}(\mathbf{T})$ is outsourced to the cloud. At each iteration, the end-user sends $\mathbf{y_k}$ to the cloud and based on the homomorphic

properties of the encryption, the cloud servers compute $\mathsf{Enc}(\mathbf{T} \cdot \mathbf{y_k})$ by

$$\begin{aligned} \mathsf{Enc}(\mathbf{T} \cdot \mathbf{y_k})[i] &= \mathsf{Enc}(\textstyle\sum_{j=1}^{n} \mathbf{T}[i,j] \cdot y_{k,j}) \\ &= \textstyle\prod_{j=1}^{n} \mathsf{Enc}(\mathbf{T}[i,j])^{y_{k,j}} \end{aligned}$$

for $i = 1, \cdots, n$ and send $\mathsf{Enc}(\mathbf{T} \cdot \mathbf{y_k})$ back to the end-user. On receiving $\mathsf{Enc}(\mathbf{T} \cdot \mathbf{y_k})$, the end-user decrypts it and get $\mathbf{y_{k+1}}$. This iteration terminates when it converges to the final result $\mathbf{y}$. At last the end-user can recover the desired solution $\mathbf{x}$ by $\mathbf{x} = \mathbf{y} - \mathbf{r}$.

As stated above, the computational overhead at the local side primarily lies in the decryption of $\mathbf{T} \cdot \mathbf{y_k}$ in each iteration. Suppose the algorithm terminates after l rounds of iteration, then the end-user has to perform $l \cdot n$ times of decryption. However, the decryption process of public-key cryptosystem is much more expensive than simple multiplication of real numbers since it mainly consists of modular exponentiation of large numbers. For instance, the decryption process [32] adopted in [18] has a complexity of $\mathcal{O}(n^3)$ and a modified version can achieve a complexity of $\mathcal{O}(n^{2+\epsilon})$. Thus, the outsourcing scheme in [18] introduces $\mathcal{O}(n^{3+\epsilon})$ computational overhead at the local side. In terms of communication overhead, the outsourcing process requires the end-user to send $\mathbf{y_k}$ and receive $\mathsf{Enc}(\mathbf{T} \cdot \mathbf{y_k})$ at each iteration. As a consequence, the communication overhead index $\mathcal{I}_c = \frac{1}{l}$ is dependent on the convergence speed. Furthermore, this iteration process requires the end-user to be "online" for the process to continue. In comparison, our scheme can limit the computational overhead to $\mathcal{O}(n^2)$ with $\mathcal{I}_c = 1$. Moreover, during the outsourcing process, the end-user is "offline", which means that after outsourcing the transformed problem, the end-user does not need to interact with the cloud servers until the result is sent back.

The system of linear equations considered in [18] includes the coefficient matrix $\mathbf{T}$ and the solution vector $\mathbf{x}$. In [18], the matrix $\mathbf{T}$ is encrypted utilizing the Paillier cryptosystem [32]

as $\mathsf{Enc}(\mathbf{T})$ and the vector $\mathbf{x}$ is transformed to $\mathbf{y} = \mathbf{x} + \mathbf{r}$, where $\mathbf{r}$ is a random vector. In comparison, CASO disguises the coefficient matrix $\mathbf{A}$ and the solution vector $\mathbf{x}$ as $\mathbf{A'} = \mathbf{AK}$ and $\mathbf{x} = \mathbf{Ky} + \mathbf{r}$, respectively. In the Paillier cryptosystem, each entry of the coefficient matrix $\mathbf{T}(i,j)$ is encrypted as $\mathsf{Enc}(\mathbf{T}(i,j)) = g^{\mathbf{T}(i,j)} r^n \bmod n^2$, where g, r, n are parameters in the cryptosystem. There are two scenarios: (i) If r's are the same for all entries in the coefficient matrix, then all the identical entries in $\mathbf{A}$ will be encrypted to identical entries in $\mathbf{A'}$. In other words, by inspecting identical entries in $\mathbf{A'}$, we can determine whether entries in $\mathbf{A}$ are identical or not. However, in CASO, since an entry in $\mathbf{A'}$ is the linear combination of entries in $\mathbf{A}$ and $\mathbf{K}$, the identical entries in $\mathbf{A'}$ would not indicate that the corresponding entries in $\mathbf{A}$ are identical. Thus, in this case, CASO will *provide better security protection.* In this case, the end-user needs to compute $n^2 + 1$ exponential operation. (ii) If a different r is used for each entry of the coefficient matrix, then the end-user has to randomly select n^2 r's, which is quite complex. Furthermore, the end-user need to compute 2 exponential operations for each entry ($g^{a_i,j}$ and r^n). Therefore, altogether, the end-user has to compute $2n^2$ exponential operations. In addition, due to security requirement, n has to be at least 1024 bits long. In this case, n^2 would be 2048 bits. As an example, the size of the outsourced coefficient matrix for 5000 variables would be around 6MB without data compression. While in scheme-1 and scheme-2 of our proposed CASO, the transformation is applied in the column basis. As a result, the order information of each column may be exposed. In this sense, the scheme in [18] may provide better protection than scheme-1 and scheme-2 regarding the coefficient matrix $\mathbf{A}$. However, in scheme-3, each entry in $\mathbf{A}$ is transformed to

$$a'_{ij} = \sum_{r=j-\omega}^{j+\omega} a_{ir} k_{rj}.$$

When $\omega > 0$, since each k_{rj} in $\mathbf{K}$ is randomly chosen, the order information in each column will also be concealed. Thus the scheme in [18] can provide comparable security protection regarding the coefficient matrix $\mathbf{A}$ as scheme-3. In scheme-4, the entries in $\mathbf{A'}$ are further permuted. As a result, there exist no explicit relation between the entry a_{ij} in $\mathbf{A}$ and the corresponding entry a'_{ij} in $\mathbf{A'}$. However, one can know for sure that the entry t'_{ij} in $\mathsf{Enc}(\mathbf{T})$ is encrypted from the entry t_{ij} in $\mathbf{T}$. Thus scheme-4 can provide better protection of $\mathbf{A}$.

In terms of the solution vector $\mathbf{x}$, in [18], the solution vector $\mathbf{x}$ is protected by adding a random vector $\mathbf{r}$ as $\mathbf{y} = \mathbf{x} + \mathbf{r}$, while in our scheme, we conceal $\mathbf{x}$ by the affine mapping $\mathbf{x} = \mathbf{K}\mathbf{y} + \mathbf{r}$. Thus, CASO scheme can provide better security protection in this aspect.

2.7.1.3 Convex Optimization

In [33], the authors proposed a verification scheme for convex optimization problems. However, they did not give any outsourcing scheme. Compare to [33], in addition to result verification, CASO also provides a secure outsourcing scheme. Even in result verification, CASO outperforms it in terms of computational complexity.

The result verification of convex optimization is divided into three categories: normal, infeasible and unbounded. The verification for normal case forms the basis for other two cases. For the normal case, the basic idea in [33] is to check the Karush-Kuhn-Tucker (KKT) optimality condition. The end-user has to evaluate the original functions as well as their differentials based on the optimal points returned by the cloud. This verification process is much more expensive since all the original functions are non-linear. In comparison, our verification scheme requires only linear operations (e.g. multiplication and addition) on the independent variables and the returned solution, therefore, it must be more efficient.

Table 2.4: Performance Comparison

	Applicability				Computational Complexity	Communication Overhead Index $\mathcal{I}_c$
	LE	LP	NLE	COPT		
Our Scheme	√	√	√	√	$\mathcal{O}(n^2)$	1
[6]		√			$\mathcal{O}(n^{2.376})$	1
[18]	√				$\mathcal{O}(n^{3+\epsilon})$	$\frac{1}{L}$
[33]				Only Verification	Not Applicable	Not Applicable

2.7.1.4 Summary

We summarize the performance comparison of CASO with some existing works in Table 2.4. We have shown that in the case of outsourcing linear programming (LP) and system of linear equations (LE), CASO outperforms the existing schemes in computational complexity. In terms of security, all the schemes are secure in protecting the original coefficient matrix. That is, given the disguised problem, input and output, it is computationally infeasible to recover the original problem, input and output. CASO can also be applied to system of non-linear equations (NLE) and convex optimization (COPT). This shows that CASO possesses better applicability. Furthermore, compared to the existing works, CASO also gives end-users the flexibility to choose the most suitable outsourcing strategy on a cost-aware basis. That is the end user can select the secret key $\mathbf{K}$ for the outsourcing scheme based on its various security demands and computational resources.

2.7.2 Numeric Results

In this section, we measure the performance of CASO utilizing MATLAB. The computation of both the end-user and the cloud server is simulated using the same computer with an Intel Core 2 Due CPU running at 2. 53 GHz with 4GB RAM. We take outsourcing of the system of linear and non-linear equations as examples. In the process of outsourcing, we

focus on the overhead of problem transformation, result recovery and the performance gain that they can achieve by outsourcing problems to the cloud. We denote the time for local computation in the outsourcing process $\mathcal{T}_e$, the time cost without outsourcing $\mathcal{T}_s$, and the performance gain $\mathcal{I} = \mathcal{T}_s/\mathcal{T}_e$.

We first show the simulation results for outsourcing of system of linear equations $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is an $n \times n$ matrix. In complexity analysis, we show that the complexities of scheme-1 and scheme-2 are in the same level while the complexity for scheme-3 and scheme-4 are comparable.

In scheme-3, when the bandwidth W equals 1, it is reduced to scheme-1. Thus in our evaluation, we take scheme-3 as an example and let $\mathbf{K}$ be a band matrix with bandwidth W varying from 1 to 31. To investigate the impact of problem size on our proposed scheme, we let n vary from 1000 to 5000. The numeric results are shown in Table 2.5. First, we can learn from the results that when the bandwidth of the banded matrix $\mathbf{K}$ becomes larger, the computational overhead at local side grows and the performance gain decreases. This fact coincides with our analysis of the trade off between complexity and security. Second, the performance gain increases with the growth of the problem dimension n. This is because our scheme requires the end-users to carry out simple operations such as addition and multiplication. And this feature becomes more obvious for the case of non-linear computation.

Then we show the performance of our proposed scheme for system of non-linear equations. We assume that the non-linear system is composed of polynomials on ten variables and let the number of independent terms N vary from 1000 to 5000. Also for the same reason, we deploy band matrix as the key matrix and let the bandwidth W vary from 1 to 3. The simulation result is shown in Table 2.6. For system of non-linear equations, the performance gain is larger than its linear counterpart. This is because CASO requires only linear operations (e.g.

Table 2.5: Performance Evaluation for System of Linear Equations

Dimension	**Bandwidth**	$\mathcal{T}_e$ **(sec)**	$\mathcal{T}_s$ **(sec)**	$\mathcal{I}$
$n = 1000$	$W = 1$	0.0265	0.2356	8.9
	$W = 7$	0.0265	0.2402	9.1
	$W = 15$	0.0546	0.2356	4.3
	$W = 31$	0.0858	0.2387	2.8
$n = 2000$	$W = 1$	0.0593	1.3962	23.6
	$W = 7$	0.0936	1.4071	15.0
	$W = 15$	0.1248	1.3853	11.1
	$W = 31$	0.1950	1.3494	6.9
$n = 3000$	$W = 1$	0.1170	3.9234	33.5
	$W = 7$	0.1856	3.9281	21.2
	$W = 15$	0.3058	3.8844	12.7
	$W = 31$	0.4867	3.8766	8.0
$n = 4000$	$W = 1$	0.2184	8.5832	39.3
	$W = 7$	0.3416	8.6924	25.4
	$W = 15$	0.7129	8.6565	12.1
	$W = 31$	1.0171	8.6768	8.5
$n = 5000$	$W = 1$	0.3260	15.8138	48.5
	$W = 7$	0.5288	15.9839	30.2
	$W = 15$	1.2683	15.8793	12.5
	$W = 31$	1.8174	15.9698	8.8

multiplication and addition) in the local environment. Similar to that of the system of linear equations, the results clearly show that there exists a trade-off between the computational complexity and security.

Table 2.6: Performance Evaluation for System of Non-linear Equations

Dimension	**Bandwidth**	$\mathcal{T}_e$ **(sec)**	$\mathcal{T}_s$**(sec)**	$\mathcal{I}$
$N = 1000$	$W = 1$	1.6800	26.2	15.6
	$W = 2$	2.4500	27.1	11.1
	$W = 3$	3.0500	26.2	8.6
$N = 2000$	$W = 1$	3.1500	118.2	37.5
	$W = 2$	5.1200	118.8	23.2
	$W = 3$	6.3900	117.1	18.3
$N = 3000$	$W = 1$	5.1300	330.8	64.5
	$W = 2$	7.7100	313.0	40.6
	$W = 3$	9.7500	320.6	32.9
$N = 4000$	$W = 1$	7.1600	712.9	99.6
	$W = 2$	12.3800	713.1	57.6
	$W = 3$	13.9000	711.4	51.2
$N = 5000$	$W = 1$	9.3700	1187.2	126.7
	$W = 2$	16.0000	1190.1	74.4
	$W = 3$	20.6700	1191.1	57.6

Chapter 3

ExpSOS: Secure and Verifiable Outsourcing of Exponentiation Operations for Mobile Cloud Computing

3.1 Introduction

The outsourcing of cryptographic computations [10, 11, 34–37] has been a popular research topic among the community. Especially, outsourcing of the modular exponentiation has been extensively studied due to its significance in cryptographic computations. In [10], the authors considered outsourcing of modular exponentiation to two servers assuming that they would not collude. The basic idea of the proposed scheme in [10] is to split the base and exponent of the modular exponentiation into two random looking pieces and then separately outsource them to two servers. Then the end-user can combine the results returned by the servers to recover the desired result. Under this scheme, the end-user can check the validity of the returned results with probability $\frac{1}{2}$. Following [10], the authors in [11] proposed a similar scheme and improved the performance by reducing one query to the servers and increasing

the verifiability to $\frac{2}{3}$. In order to eliminate the assumption that the two-server would not collude, the authors in [38] proposed a scheme to outsource modular exponentiation to one single server. However, at the local side, the end-user still needs to carry out one modular exponentiation u^{χ}, where χ is a security parameter. As a result, the computational gain is limited for the end-user. Moreover, all these three schemes rely on pre-computation of modular exponentiations of some random integers. This will cause extra overhead to the end-user's limited computational power or storage space depending on the method by which pre-computation is implemented. In our recent work [39], we proposed an efficient scheme to securely outsource scalar multiplication on elliptic curves. However, it did not provide algorithms for result verification.

The existing research in this area can be classified into three categories based on the model of the cloud S: the Honest-but-Curious Single-server (HCS) model [40], the Malicious Multiple-servers (MM) model [10, 11, 41] and Malicious Single-server (MS) model [38, 42]. In particular, in the *honest but curious* model, the cloud will honestly fulfill its advertised functionality. However, S could be curious. It may try to exploit any key information from the outsourced task, which may include the input, the output as well as the intermediate computational results. When the outsourced data is sensitive, this could cause severe security and privacy issues. In the *malicious* model, the cloud S may not carry out the desired computation truthfully. This can happen for various reasons. A simple scenario could be that the cloud simply returns some trivial results since the computational resource is a commodity for the cloud server. As a result, the end-user E is unable to receive a valid result from the cloud server S.

The models are hierarchical in the sense that a secure outsourcing scheme designed for the single-server model can be extended to multiple-server model and a scheme for mali-

cious cloud can be extended to honest but curious cloud. Specifically, these models can be organized into three layers: at the bottom layer is the HCM (Honest-but-Curious Multiple-servers) model, in the middle are the MM and HCS and on the top is MS. A secure outsourcing scheme designed for a model in an upper layer is also suitable for that in a lower layer. Thus, a secure outsourcing scheme for MS is most widely applicable and achieves the highest security standard. In this paper, we first propose a secure outsourcing scheme for the HCS model. Then a verification scheme is proposed for MS model.

Apparently, it is much more desirable and secure to outsource exponentiation operations to one single server instead of multiple servers with security based on the assumption that the servers would not collude. The secure outsourcing scheme should not impose expensive computational overhead at local side. Otherwise, the performance gain from outsourcing would diminish. The scheme should also provide high verifiability. Ideally, the end-user should be able to verify the validity of the returned result with probability 1.

In this paper, we extend the notion of modular exponentiation to general exponential operations in a finite group, including scalar multiplication on elliptic curves. In general, each exponential operation consists of a series of basic group operations. The number of such operations varies with the exponent. In this sense, modular exponentiation and scalar multiplication can both be regarded as exponentiation operations. Thus, we propose a Secure Outsourcing Scheme for general Exponential (ExpSOS) operations. The proposed ExpSOS is based on a secure disguising procedure that maps the integers in the group $\mathbb{R}_N$ to the larger group $\mathbb{R}_L$ so that the cloud will carry out the computation in $\mathbb{R}_L$ while still keeps N secure. From the result returned by the cloud, the end-user can recover the result back to $\mathbb{R}_N$ efficiently.

The main contributions of this paper can be summarized as follows:

- We formally define a secure outsourcing scheme and four outsourcing models. The proposed ExpSOS is shown to be effective under all four different models.
- We develop schemes to securely outsource exponentiation operations in a general finite group, including modular exponentiation and scalar multiplication on elliptic curves.
- We outsource exponential operations to one single untrusted server eliminating the non-collusion assumption between multiple servers.
- Our proposed ExpSOS is efficient in that it requires only a small number of modular multiplications at local side.
- We propose a verification scheme such that the end-user can verify the validity of the result with probability approximately 1.

3.2 Secure Computation Outsourcing Model

3.2.1 System Model and Threat Model

In the general settings of computation outsourcing, the system consists of two entities: an end-user E and the cloud S. The end-user E is resource-constrained. It has limited computational power and storage space. The cloud S is regarded as possessing abundant resources and is able to carry out computational intensive operations. The cloud can be further modelled as the *single-server* model and the *multiple-server* model. In the single-server model, the cloud is viewed as one unit. In contrast, in the multiple-server model, the cloud is divided into two or more distinct units. Each unit carries out the computational tasks independently. While communication between different units is allowed, key information is only limited to

the individual unit since otherwise, security of the whole system may be in jeopardy.

In addition, the cloud S can be either honest (however, could be curious) or malicious. Our scheme can be applied to all four threat models combined with these two assumptions.

Suppose the end-user E wishes to accomplish a computationally intensive task $F(\mathbf{x}) = \omega$, where $\mathbf{x}$ is the input and ω is the output of the task. However, due to the limited resources, E may not be able to finish the task using the locally available resources. The computational task F could be outsourced to S. Unfortunately, the cloud is only a shared server and cannot be fully trusted. Therefore, we have to make sure that it is infeasible for S to derive any key information of both $\mathbf{x}$ and ω from the outsourced task.

3.2.2 Definition of Secure Outsourcing Scheme

We follow the security model defined in [10], which is also employed in following works such as [11, 38, 41, 42]. In [10], the authors consider splitting a cryptographic algorithm Alg to a trusted component T and an untrusted component C that T can make queries to. It is also proposed that an adversary A consists of two parts: the adversarial environment $\mathcal{E}$ that submits adversarially chosen inputs to Alg and a malicious component C' that operates in place of C. Based on this model, *algorithm with outsource-IO* is defined in [10] as an algorithm $\mathsf{Alg}(x_{hs}, x_{hp}, x_{hu}, x_{ap}, x_{au}) \rightarrow (y_s, y_p, y_u)$ that takes in five inputs and produces three outputs, where x_{hs} is the honest secret input which is honestly generated and is only known to T; x_{hp} is the honest and protected input that is known to both T and $\mathcal{E}$, but not to C; x_{hu} is the honest and unprotected input that is known to T, $\mathcal{E}$ and C; x_{ap} is the adversarial protected input that is adversarially generated and is known to both T and $\mathcal{E}$, but not to C; x_{au} is the adversarial input that is known to T, $\mathcal{E}$ and C. y_s is the secret output that is only known to T; y_p is the protected output that is known to both T and $\mathcal{E}$,

but not to C; y_u is the unprotected output that is known to T, $\mathcal{E}$ and C.

Based on this outsource-IO, the notion of outsourcing security is further defined in [10]. The security definition requires that for any adversary $A = (\mathcal{E}, C')$, there exists a pair of polynomial-time simulators $(\mathcal{S}_1, \mathcal{S}_2)$ that can simulate the views of $\mathcal{E}$ and C', respectively. Moreover, the two pairs of views are computationally indistinguishable. In the following, we propose the formal security definition for our outsourcing model. In particular, our definition can be considered as a tailored version of that in [10]. This is because we do not explicitly consider an adversary environment $\mathcal{E}$ since there exist no adversarially generated inputs in our settings. Accordingly, we define an algorithm with outsource-IO as follows.

Definition 3.1 (Algorithm with outsource-IO) *An algorithm* Alg *obeys the outsource input/output specification if it takes two inputs and produces two outputs, i.e.,* $\mathsf{Alg}(x_p, x_u) \rightarrow (y_p, y_u)$. *Especially,*

- x_p *is the protected input known only to* T.

- x_u *is the unprotected input, which is known to* T *and* C.

- y_p *is the protected output known only to* T.

- y_u *is the unprotected output, which is known to* T *and* C.

Definition 3.2 (Outsource-security) *Let* Alg *be an algorithm with outsource-IO. A pair of algorithms* (T, C) *is said to be an outsource-secure implementation of* Alg *if:*

Correctness T^C *is a correct implementation of* Alg.

Security *For any probabilistic polynomial-time adversary* C', *there exists probabilistic expected polynomial-time simulator* $\mathcal{S}$ *such that the following pair of random variables is computationally indistinguishable.*

$VIEW_{real} \sim VIEW_{ideal}$ *(The adversary* C' *learns nothing.)*

- *The view that the adversary* C' *obtains by participating in the following* REAL *process:*

$$VIEW_{real} = \{(istate, x_p, x_u) \leftarrow I(1^k);$$
$$(tstate, cstate, y_p, y_u) \leftarrow T^{C'}(x_p, x_u) : cstate\}.$$

In this real process, the state variable (e.g., istate, tstate and cstate) keeps records of the view of the corresponding party. First, the protected input x_p *and unprotected input* x_u *are picked using an honest, stateful process* I *to which the adversary does not have access. Next, the algorithm* $T^{C'}$ *is run on the inputs* (x_p, x_u) *and produces output* (y_p, y_u) *as well as tstate and cstate for* T *and* C'*, respectively. The view of* C' *in the real process is cstate.*

- *The IDEAL process:*

$$VIEW_{ideal} = \{(istate, x_p, x_u) \leftarrow I(1^k);$$
$$(astate, y_p, y_u) \leftarrow \mathsf{Alg}(x_p, x_u);$$
$$(sstate, cstate) \leftarrow \mathcal{S}^{C'}(x_u) : cstate\}$$

In the ideal process, a simulator $\mathcal{S}$ *is utilized to simulate the view of* C'*. Note that* $\mathcal{S}$ *is given only the unprotected input and can make queries to the adversary* C'*.*

Definition 3.3 (α-efficient) *Suppose the running time for a task* F *to be processed locally by* E *is* t_0*. Under a Secure Outsourcing Scheme (SOS), the running time of local processing for* E *is* t_p*. Then the SOS is* α*-efficient if* $\frac{t_0}{t_p} \geq \alpha$*.*

Definition 3.4 (β-verifiable) *Given the returned output Ω and the proof Φ, denote the probability that E is able to verify the validity of the result ω as κ. Then an SOS is β-verifiable if $\kappa \geq \beta$.*

From the definition above, we can see that a larger α indicates a better performance of a secure outsourcing scheme, while a larger β means a better verifiability.

3.3 ExpSOS: Secure Outsourcing of Exponentiation Operations

3.3.1 General Framework

The general framework of an SOS consists of four different functions $(\mathcal{T}, \mathcal{C}, \mathcal{R}, \mathcal{V})$.

1. **Problem Transformation** $\mathcal{T} : F(\mathbf{x}) \rightarrow G(\mathbf{y})$. The end-user $\mathcal{U}$ locally transforms the problem $F(\mathbf{x})$ to a new form $G(\mathbf{y})$, where $\mathbf{y}$ is the new input and G is the new problem description. E then outsources $G(\mathbf{y})$ to the cloud server S.

2. **Cloud Computation** $\mathcal{C} : G(\mathbf{y}) \rightarrow (\Omega, \Phi)$. The cloud S solves the transformed problem $G(\mathbf{y})$ to obtain the corresponding result Ω. At the same time, S returns Φ that is a proof of the validity of the result.

3. **Result Recovery** $\mathcal{R} : \Omega \rightarrow \omega$. Based on the returned result Ω, the end-user $\mathcal{U}$ recovers the result ω of the original problem $F(\mathbf{x})$.

4. **Result Verification** $\mathcal{V} : (\Omega, \Phi, \omega) \rightarrow \Lambda =$ True or False. Based on ω, Ω and the proof Φ, the end-user $\mathcal{U}$ verifies the validity of the result.

3.3.2 Secure Disguising Procedure

In this paper, we assume that for the end-user, exponentiation operations are operated in the integer ring modular N, denoted as $\mathbb{R}_N$. We note that N is not necessarily a prime number. It can also be a product of large prime numbers. To outsource modular exponentiation to the shared cloud, we first need to conceal the modular N. To achieve this, we multiply N by a randomly selected large prime p and define $L = pN$. We define a secure disguising procedure to map $x \in \mathbb{R}_N$ to $y \in \mathbb{R}_L$ as follows:

1. Select a random $k, 1 \leq k \leq p-1$.

2. Compute $y = x + kN \pmod{L}$.

Without the knowledge of k, it is hard to determine which point x is mapped to. To recover x from y, the end-user only needs to compute $y = (x + kN) = x \pmod{N}$. Therefore, regardless of which k is selected to outsource x, we will always have $y = x \pmod{N}$.

Now, we explore the properties of the computation outsource and recovery functions. They are key to our proposed secure outsourcing scheme.

Theorem 3.1 *For any $x_1, x_2 \in \mathbb{R}_N$ and their corresponding disguised form $y_1 = x_1 + k_1 N$ and $y_2 = x_2 + k_2 N$, where k_1 and k_2 are randomly selected integers, $1 \leq k_1, k_2 \leq p-1$, we have*

$$\begin{aligned} x_1 + x_2 &= (y_1 + y_2) \pmod{N}. \\ x_1 x_2 &= (y_1 y_2) \pmod{N}. \end{aligned}$$

Proof We can verify that

$$\begin{aligned}(y_1 + y_2) \pmod N &= ((x_1 + k_1 N) \pmod L + (x_2 + k_2 N) \pmod L) \pmod N \\ &= (x_1 + k_1 N + x_2 + k_2 N) \pmod N \\ &= (x_1 + x_2) \pmod N.\end{aligned}$$

Similarly, it can be verified that $x_1 x_2 = y_1 y_2 \pmod N$.

Corollary 3.1 *Suppose* $\mathbf{x} = (x_1, x_2, \cdots, x_n) \in \mathbb{R}_N^n$ *and* $\mathbf{y} = (y_1, y_2, \cdots, y_n) = (x_1 + k_1 N, x_2 + k_2 N, \ldots, x_n + k_n N)$ *and let* $\varphi \colon \mathbb{R}^n \to \mathbb{R}$ *be an n-variable polynomial function with coefficients in* $\mathbb{R}$*, where* $\mathbb{R}$ *can be* $\mathbb{R}_N$ *or* $\mathbb{R}_L$*. Then we have*

$$\varphi(\mathbf{x}) = \varphi(\mathbf{y}) \pmod N.$$

Theorem 3.1 enables us to transform the addition and multiplication in a ring $\mathbb{R}_N$ into the corresponding operations in another large ring $\mathbb{R}_L$. Since polynomial evaluation consists of addition and multiplication, Corollary 3.1 states that we can transform polynomial evaluation in $\mathbb{R}_N$ into corresponding operations in $\mathbb{R}_L$.

3.3.3 Secure Outsourcing of Modular Exponentiation under HCS Model

3.3.3.1 Conceal the Base in Modular Exponentiation Outsourcing

Consider modular exponentiation $R = u^a \pmod N$. We assume that N is either a large prime or product of large prime numbers, which is the typical situation in cryptosystems.

Theorem 3.1 states that the result of multiplication in the ring $\mathbb{R}_N$ can be obtained from the multiplication in $\mathbb{R}_L$ through the transformation function and the inverse function. If we take $x_1 = x_2 = u$, we can get

$$((u + rN) \pmod{L})^2 = u^2 \pmod{N}.$$

If we repeat the multiplication in $\mathbb{R}_N$ for a times, we have the following corollary.

Corollary 3.2 *For $u, a, r \in \mathbb{R}_N$, we have*

$$((u + rN) \pmod{L})^a = u^a \pmod{N}.$$

Corollary 3.2 gives us a way to conceal the base when outsourcing modular exponentiation. That is, we can first transform the original base u to $U = (u + rN) \pmod{L}$, where $r \in \mathbb{R}_N$ is a random integer. Then the cloud can compute $U^a \pmod{L}$ based on which the original result can be recovered by computing $U^a \pmod{N} = u^a \pmod{N}$.

3.3.3.2 Conceal the Exponent in Modular Exponentiation Outsourcing

The remaining task is to conceal the exponent a. We have the following theorem.

Theorem 3.2 *For $N = p_1 p_2 \cdots p_m$, where $p_1, p_2, \cdots, p_m$ are distinct prime numbers, we have*

$$u^{a+k\varphi(N)} = u^a \pmod{N},$$

where k is a random integer and $\varphi(\cdot)$ is the Euler's totient function.

Proof We first prove that $u^{1+k\varphi(N)} = u \pmod{N}$. For each prime factor p_i of N, $i = 1, 2, \cdots, m$. There are two possible cases:

- Case 1: $\gcd(u, p_i) \neq 1$, that is u and p_i are not relatively prime. Since p_i is prime, we have $p_i \mid u$. Thus

$$u^{1+k\varphi(N)} - u = 0 \pmod{p_i},$$

which means that $p_i \mid (u^{1+k\varphi(N)} - u)$.

- Case 2: $\gcd(u, p_i) = 1$, that is u and p_i are relatively prime. Then, by the Euler's Theorem, we have $u^{\varphi(p_i)} = 1 \pmod{p_i}$. From the multiplicative property of the Euler's totient function, we have $\varphi(N) = \varphi(p_1)\varphi(p_2)\cdots\varphi(p_m)$. Let $\theta(p_i) = \varphi(N)/\varphi(p_i)$. Then,

$$\begin{aligned} & u^{1+k\varphi(N)} \pmod{p_i} \\ = \; & u \cdot u^{k\varphi(p_1)\varphi(p_2)\cdots\varphi(p_m)} \pmod{p_i} \\ = \; & u \cdot (u^{\varphi(p_i)})^{k\theta(p_i)} \pmod{p_i} \\ = \; & u \cdot (1)^{k\theta(p_i)} \pmod{p_i} \\ = \; & u \pmod{p_i}. \end{aligned}$$

That is $(u^{1+k\varphi(N)} - u) = 0 \pmod{p_i}$.

Therefore, in both cases, we have proved that $p_i \mid (u^{1+k\varphi(N)} - u)$. Since p_i is arbitrarily selected and $p_1, p_2, \cdots, p_m$ are distinct primes, we have

$$N \mid (u^{1+k\varphi(N)} - u).$$

Hence, $u^{1+k\varphi(N)} = u \pmod{N}$. Multiplying both sides of the equation by u^{a-1}, we can obtain

$$u^{a+k\varphi(N)} = u^a \pmod{N}.$$

In Theorem 3.2, we do not require u and N to be co-prime as required in the Euler's theorem. Instead, we assume that N is the product of distinct prime numbers. For instance, in RSA, the modulus $N = pq$ is the product of two distinct prime numbers.

Theorem 3.2 introduces a way to conceal the exponent a. That is, by transforming the original exponent a to $A = a + k\varphi(N)$, where k is a random integer, we can conceal a due to the randomness of k. Now, based on Theorem 3.1 and Theorem 3.2, we can construct our secure outsourcing scheme for modular exponentiation. In the secure outsourcing scheme, we utilize a function $\mathcal{C}(U, A, L)$ to denote the computation of a modular exponentiation for the cloud as $\mathcal{C}(U, A, L) = U^A \pmod{L}$. The result recovery function is $\mathcal{R}(R, N) = R \pmod{N}$.

3.3.3.3 ExpSOS Protocol

The secure outsourcing scheme for modular exponentiation under HCS model is given in Protocol 1.

Protocol 1 Secure Outsourcing of Modular Exponentiation Under HCS Model

Input: $N, u, a \in \mathbb{R}_N$.
Output: $R = u^a \pmod{N}$.

Key Generation KeyGen$(1^\lambda, N) \to (p, L)$:

1: E generates a large prime p and calculates $L \leftarrow pN$.
2: The public key is $K_p = \{L\}$, and the private key is $K_s = \{p, N\}$.

Problem Transformation $\mathcal{T}(a, u) \to (A, U)$:

1: E selects random integers $r, k \in \mathbb{R}_N$ as the temporary key.
2: E calculates $A \leftarrow a + k\varphi(N)$, $U \leftarrow (u + rN) \pmod{L}$.
3: E outsources (U, A, L) to the cloud.

Cloud Computation $\mathcal{C}(A, U, L) \to R_1$:

1: S computes $R_1 \leftarrow \mathcal{C}(U, A, L) = U^A \pmod{L}$.
2: S returns R_1 to E.

Result Recovery $\mathcal{R}(R_1, N) \to R$:

1: E recovers the result as $R \leftarrow \mathcal{R}(R_1, N) = R_1 \pmod{N}$.

The soundness of the outsourcing scheme is guaranteed by the following theorem:

Theorem 3.3 *The secure outsourcing scheme for modular exponentiation is sound. That is* $R = R_1 = u^a \pmod N$.

The proof of Theorem 5.1 is straightforward based on Theorem 3.1 and Theorem 3.2. Specifically, by transforming the original problem of modular exponentiation to a disguised form, our proposed ExpSOS under HCS model is sound.

3.3.4 Secure Outsourcing of Scalar Multiplication under HCS Model

In this section, we consider secure outsourcing of scalar multiplication sP on an elliptic curve $E(\mathbb{F}_p)$ described by the following short Weierstrass equation:

$$E: \ y^2 = x^3 + bx + c, \tag{3.1}$$

where the coefficients b, c and the coordinates of the points are all in a finite field $\mathbb{F}_p$. Furthermore, for cryptographic applications, we usually work with points in a set of m-torsion points $E(\mathbb{F}_p)[m]$ defined as $E(\mathbb{F}_p)[m] = \{P \in E(\mathbb{F}_p) : [m]P = \mathcal{O}\}$, where $\mathcal{O}$ is the point at infinity. Thus, we assume $P \in E(\mathbb{F}_p)[m]$ and $s \in \mathbb{Z}_m$.

Secure outsourcing of scalar multiplication relies on two basic operations, point addition and point doubling. They play a similar role as modular multiplication in the outsourcing of modular exponentiation. Specifically, the "double-and-add" algorithm to calculate scalar multiplication on elliptic curves consists of a series of point addition and point doubling. Thus intuitively, we can regard secure outsourcing of point addition and point doubling as two building blocks to implement scalar multiplication.

We utilize projective coordinate to represent a point $P = (x, y, z)$ corresponding to the point $Q = \left(\frac{x}{z}, \frac{y}{z}\right)$ in the affine coordinates. As a result, the computation of point addition

and point doubling consists of only modular addition and multiplication. Specifically, given two points $P = (x_1, y_1, z_1)$ and $Q = (x_2, y_2, z_2)$ such that $P \neq \pm Q$, the point addition $P + Q = (x_3, y_3, z_3)$ can be calculated as follows:

$$x_3 = \sigma\tau, y_3 = \rho(\sigma^2 x_1 z_2 - \tau) - \sigma^3 y_1 z_2, z_3 = \sigma^3 z_1 z_2,$$

where

$$\rho = y_2 z_1 - y_1 z_2, \sigma = x_2 z_1 - x_1 z_2,$$
$$\tau = \rho^2 z_1 z_2 - \sigma^3 - 2\sigma^2 x_1 z_2.$$

The point doubling $2P = (x_4, y_4, z_4)$ can be calculated as follows:

$$x_4 = 2\sigma\mu, y_4 = \rho(4\tau - \mu) - 8y_1^2\sigma^2, z_4 = 8\sigma^3,$$

where

$$\rho = bz_1^2 + 3x_1^2, \sigma = y_1 z_1, \tau = x_1 y_1 \sigma, \mu = \rho^2 - 8\tau.$$

In projective coordinates, one point addition and doubling take 14 multiplications and 12 multiplications, respectively.

Corollary 3.1 states that by mapping the variables of a polynomial from a finite field to variables in a ring, we can evaluate the polynomial in the ring and recover the result in the finite field. This gives us the insight of our proposed scheme since essentially, point addition and point doubling are both the process of evaluating polynomials on the coordinates of the points. Thus, we can construct the secure computation scheme for point addition and point doubling as in Protocol 2.

Theorem 3.4 *The proposed secure point addition and point doubling protocol is sound.*

Protocol 2 Secure Point Addition and Point Doubling

Input: $E = \{b, c, p\}$ and $P = (x_1, y_1, z_1)$, $Q = (x_2, y_2, z_2)$.
Output: point $R = P + Q = (x_3, y_3, z_3)$.

1: Select a large prime q and compute $N = pq$.
2: For a coordinate x_i, select a random integer k_i and compute $x_i' = (x_i + k_i p) \pmod N$.
3: Transform the points P, Q and the elliptic curve E to $P' = (x_1', y_1', z_1')$, $Q' = (x_2', y_2', z_2')$ and $E' = \{b', c', N\}$ respectively as described in Step 2.
4: Outsource P', Q' and E' to the cloud.
5: Cloud computes $R' = P' + Q'$ following the point doubling or point addition prodecure.
6: On receiving $R' = (x_3', y_3', z_3')$, recover R as $R = (x_3', y_3', z_3') = (x_3, y_3, z_3) \pmod p$.

The proof of Theorem 3.4 is straightforward from the Corollary 3.1.

The above theorem enables us to conceal the points as well as the parameters of the elliptic curve from the cloud. To outsource scalar multiplication sP, the remaining part is to conceal the multiplier s. We utilize the property of the order m of the torsion group that is $rmP = \mathcal{O}$, for an arbitrary point $P \in E[m](\mathbb{F}_p)$ and any integer r. As a result, we can conceal s by adding it to a multiple of m as $s' = s + rm$, where r is a random integer. Now, we can summarize the secure outsourcing scheme of scalar multiplication as in Protocol 3.

Protocol 3 Secure Outsourcing of Scalar Multiplication Under HCS Model

Input: $E = \{b, c, p\}, P = (x_1, y_1, z_1), s, m$
Output: point $R = sP$.

Key Generation $\mathsf{KeyGen}(1^\lambda, p) \to N$:

1: End-user selects a large prime q and compute $N \leftarrow pq$.

Problem Transformation $\mathcal{T}(P, E) \to (P', E')$:

1: End-user generates random integers $k_1, k_2, k_3, k_4, k_6, r$.
2: Computes $x_1' \leftarrow (x_1 + k_1 p) \pmod N$, $y_1' \leftarrow (y_1 + k_2 p) \pmod N$, $z_1' \leftarrow (z_1 + k_3 p) \pmod N$, $b' \leftarrow (b + k_4 p) \pmod N$, $c' \leftarrow (c + k_6 p) \pmod N$, $s' \leftarrow s + rm$.
3: End-user outsources $P' = (x_1', y_1', z_1')$, $E' = \{b', c', N\}$ and s'.

Cloud Computation $\mathcal{C}(s', P') \to R'$:

1: The cloud computes $R' \leftarrow s'P' = (x_3', y_3', z_3')$ utilizing the double-and-add algorithm.

Result Recovery $\mathcal{R}(R', p) \to R$:

1: The end-user recovers the result R as $R \leftarrow R' = (x_3', y_3', z_3') \pmod p$.

Theorem 3.5 *The secure outsourcing scheme for scalar multiplication is sound in* $\mathbb{F}_q$. *That is* $R = sP$.

Proof From Theorem 3.4, we know that the secure computation scheme for point addition and point doubling is sound. Since the double-and-add algorithm to compute scalar multiplication consists of a series of point addition and point doubling, we have $R = s'P = (s + rm)P = sP + rmP = sP + \mathcal{O} = sP$.

In the next section, we propose a verification scheme to ensure that ExpSOS is secure under the MS model.

3.4 Result Verification

In this section, we first analyze the necessary properties of a result verification scheme through some counter examples. We then propose a result verification scheme for outsourcing of modular exponentiation under MS model. We show that the verification scheme can also be applied to the outsourcing of scalar multiplication.

For the HCS model discussed in the previous section, we assume that the cloud will honestly conduct its advertised functionality. That is, to compute the function $\mathcal{C}(U, A, L)$ and return the correct result $U^A \pmod{L}$. However, in the MS model, the cloud may manipulate the result in order to save computational resources. Thus, to verify the soundness of the result returned by the cloud is a critical issue.

A natural way to verify the result, as utilized in many previous works [10, 11, 34], is to outsource the problem multiple times and verify whether the returned results satisfy certain criteria. However, this methodology may cause potential security problems if it is not carefully designed. This is because outsourcing multiple times essentially gives more

information about the original problem to the cloud, which may increase the probability for the cloud to recover the original problem. Moreover, the cloud may manipulate the results in order to satisfy the criteria, thus passing the verification. Therefore, we believe that an effective verification scheme should at least have the following two properties:

- **Security**: The verification process should not reveal any key information about the original problem to the cloud.
- **Anti-manipulation**: It is infeasible for the cloud to manipulate the result and pass the verification process.

We utilize two counter-examples in verifying modular exponentiation to illustrate the significance of the above properties and emphasize the key issues in designing a verification scheme.

Example 3.1 *Transform the exponent a to $A_1 = a + k_1\varphi(N)$ and $A_2 = a + k_2\varphi(N)$. The cloud returns results $R_1 = U^{A_1} \pmod{L}$ and $R_2 = U^{A_2} \pmod{L}$. The end-user checks whether the condition $R_1 = R_2 \pmod{N}$ holds.*

Unfortunately, the above example violates the security property. When the cloud possesses A_1 and A_2, it can calculate $A_1 - A_2 = (k_1 - k_2)\varphi(N)$, which is a multiple of the Euler's totient function $\varphi(N)$. In this case, the cloud can factorize $(k_1 - k_2)\varphi(N)$ based on which, the cloud may be able to check the primality of N. Since N is a product of large primes, the consequence is that the cloud can limit the valid value of N to a short list. This means that the cloud can derive some key information from the outsourced problem thus making outsourcing insecure. Similarly, some variances of this type of method (e.g., $A_1 = a + k_1\varphi(N)$ and $A_2 = ca + k_2\varphi(N)$, where c is a known constant) may also have security problems.

Example 3.2 *Transform the exponent a to $A_1 = a + k_1\varphi(N)$ and $A_2 = a + t + k_2\varphi(N)$, where t is a relatively small integer, which makes computing $u^t \pmod N$ within the end-user's computational ability. The cloud returns results $R_1 = U^{A_1} \pmod L$ and $R_2 = U^{A_2} \pmod L$. The end-user checks whether the condition $(R_1 \cdot u^t) = R_2 \pmod N$ holds.*

Due to the randomness of t, the cloud is not able to obtain a multiple of $\varphi(N)$. However, from the equality condition $(R_1 \cdot u^t) = R_2 \pmod N$, we have $U^{A_1} \cdot u^t = U^{A_2} \pmod N$, which is equivalent to

$$u^t = U^{A_2 - A_1} \pmod N.$$

In this case, the cloud can manipulate the two integers A_1' and A_2'. As long as $A_2' - A_1' = A_2 - A_1$, the results will pass the verification but the recovered result $R = U^{A_1'} \pmod N$ is incorrect. This means that the cloud can manipulate a false result while passing the verification process.

From the above two counter examples, we can see that security and anti-manipulation are two critical issues in result verification schemes. In the following Protocol 4, we propose a verification scheme for modular exponentiation.

Now, we utilize an example to illustrate our proposed ExpSOS under MS model.

Example 3.3 *Suppose the end-user $\mathcal{U}$ wants to calculate $u^a \pmod N$, where $N = 431$ is a prime, $u = 189$ and $a = 346$. E can outsource $u^a \pmod N$ as follow:*

1. Key Generation: E select a prime number $p = 397$ and calculate $L = pN = 171107$. Then E selects random integers $r = 146, k_1 = 332, k_2 = 68$ and $t_1 = 4, t_2 = 12$ with $t_1, t_2 < b = 16$.

2. Problem Transformation: E calculates $A_1 = a + k_1\varphi(N) = 143106$, $A_2 = t_1 a + t_2 +$

Protocol 4 ExpSOS under MS Model

Input: $N, u, a \in \mathbb{R}_N$.
Output: $R_0 = u^a \pmod N$, $\Lambda =$ True or False.

Key Generation KeyGen$(1^\lambda, N) \to (p, L)$:

1: E generates a large prime p and calculate $L \leftarrow pN$.
2: The public key is $K_p = \{L\}$, and the private key is $K_s = \{p, N\}$.

Problem Transformation $\mathcal{T}(a, u) \to (A_1, A_2, U)$:

1: E selects random integers r, k_1, k_2, t_1, t_2 as the ephemeral key with the constraint that $t_1, t_2 \leq b$.
2: E calculates $A_1 \leftarrow a + k_1\varphi(N)$, $A_2 \leftarrow t_1 a + t_2 + k_2\varphi(N)$ and $U \leftarrow (u + rN) \pmod L$.
3: E outsources (U, A_1, A_2, L) to the cloud.

Cloud Computation $\mathcal{C}(A_1, A_2, U, L)$:

1: S computes $R_1 \leftarrow U^{A_1} \pmod L$ and $R_2 \leftarrow U^{A_2} \pmod L$.
2: S returns R_1 and R_2 to E.

Result Verification $\mathcal{V}(R_1, R_2) \to \Lambda$:

1: E checks whether $(R_1)^{t_1} \cdot u^{t_2} = R_2 \pmod N$.
2: If the equality holds, set $\Lambda \leftarrow$ True. Otherwise, set $\Lambda \leftarrow$ False.

Result Recovery $\mathcal{R}(R_1, N) \to R_0$:

1: E recovers the result as $R_0 \leftarrow \mathcal{R}(R_1, N) = R_1 \pmod N$.

$k_2\varphi(N) = 30636$ and $U = (u + rN) = 63115 \pmod L$. E then queries $\mathcal{C}(U, A_1, L)$ and $\mathcal{C}(U, A_2, L)$ to the cloud S.

3. Cloud computation: S computes $R_1 = U^{A_1} \pmod L = 63115^{143106} \pmod{171107} = 81281$, $R_2 = U^{A_2} \pmod L = 63115^{30636} \pmod{171107} = 55473$ and returns R_1 and R_2 to E.

4. Result Verification: E calculates $R_1^{t_1} \cdot u^{t_2} \pmod N = (190^4 \cdot 189^{12}) \pmod{431} = 305$ and $R_2 \pmod N = 55473 \pmod{431} = 305$ that satisfy $R_1^{t_1} \cdot u^{t_2} = R_2 \pmod N$. Thus the returned results are correct.

5. Result Recovery: E recovers the result as $R = R_1 \pmod N = 81281 \pmod{431} = 190$ that is equal to $u^a = 190 \pmod N$.

In Protocol 4, the two outsourced exponential operations are related through an affine function. As a result, the cloud is unable to derive a multiple of $\varphi(N)$ only based on A_1 and A_2. Moreover, the cloud cannot manipulate the results to create a verifiable equality.

This verification scheme can also be applied to the outsourcing of scalar multiplications. The base point P can be transformed to P' as described in Protocol 3. The exponent s can be transformed to $s_1 = s + r_1 m$ and $s_2 = t_1 s + t_2 + r_2 m$, where r_1, r_2, t_1, t_2 are random integers and $t_1, t_2 \leq b$. Then the end-user can check the condition $Q_2 = t_1 Q_1 + t_2 P$, where $Q_1 = s_1 P'$ and $Q_2 = s_2 P'$.

3.5 Complexity and Security Analysis

In this section, we analyze the security and the computational complexity of ExpSOS. We utilize the secure outsourcing of modular exponentiation as a representative to perform the analysis. The analysis of outsourcing scalar multiplication can be conducted in a similar way. We show that ExpSOS is secure under both HCS and MS model. Specifically, under the HCS model, the ExpSOS is $\frac{1}{2}\log_2 a$-efficient. Under the MS model, the ExpSOS is $\frac{1}{2}\log_b a$-efficient and $(1 - \frac{1}{2b})$-verifiable, where a is the exponent and b is the security parameter.

3.5.1 Security Analysis

In ExpSOS, we conceal the base u through the expansion function $(u + rN) \pmod L$ and the exponent a is mapped to $a + k\varphi(N)$. In our analysis, we show that given the public information $\{L, U, A_1, A_2\}$, the cloud cannot derive any key information about the input $\{u, a, N\}$ and the output $R = u^a \pmod N$.

Theorem 3.6 *Under the MS model, ExpSOS is an outsource-secure implementation of mod-*

ular exponentiation, where the inputs (u, a, N) *are protected.*

Proof The correctness of ExpSOS is guaranteed by Theorem 5.1. In the following, we prove the security of ExpSOS.

In our setting, the protected input is $x_p = (u, a, N)$ and the unprotected input is set to be $x_u = L$. The protected output is $y_p = R = u^a \pmod N$ while the unprotected output is empty. In the ideal process, the simulator $\mathcal{S}$ acts in the following manner. Whenever receives the input, $\mathcal{S}$ ignores it. Then, it makes queries (U^*, A_1^*) and (U^*, A_2^*) to malicious C'. Then $\mathcal{S}$ saves both its state and C''s state. We note that in the real process, the inputs are computationally disguised in a random manner by the end-user before querying to C'. In the ideal process, the simulator $\mathcal{S}$ always creates random and independent queries to C'. In this sense, the inputs to C' are computationally indistinguishable. Thus, we have $VIEW_{real} \sim VIEW_{ideal}$.

We show that the proposed verification scheme has the security and effectiveness properties as described previously. The verifiability is based on the likelihood of finding two integers R_1 and R_2 so that $R_1^{t1} \cdot u^{t2} = R_2 \pmod N$ holds true.

Lamma 3.1 *Given* (A_1, A_2, U, L)*, an adversary can generate* R_1 *and* R_2 *with probability* $1/b$ *such that* $R_1^{t1} \cdot u^{t2} = R_2 \pmod N$

Proof Given A_1 and A_2, an adversary can always select $B_1 = A_1 + \theta_1$ and $B_2 = A_2 + \theta_2$.

Then,

$$\begin{aligned} & U^{B_2 - t_1 B_1} \pmod N \\ = \; & U^{A_2 + \theta_2 - t_1 A_1 - t_1 \theta_1} \pmod N \\ = \; & u^{t_1 a + t_2 + k_2 \varphi(N) + \theta_2 - t_1 a - t_1 k_1 \varphi(N) - t_1 \theta_1} \pmod N \\ = \; & u^{t_2} \cdot u^{\theta_2 - t_1 \theta_1} \pmod N. \end{aligned}$$

It is obvious that as long as $\theta_2 - t_1\theta_1 = 0$, the equality will hold. If the value of t_1 is correctly guessed, an adversary can select θ_1 and θ_2 such that $\theta_2 - t_1\theta_1 = 0$. Since the probability to correctly guess t_1 is $1/b$, an adversary can manipulate the result with probability $1/b$.

Theorem 3.7 *Under MS model, the ExpSOS is* $(1 - \frac{1}{2b})$*-verifiable.*

This theorem indicates that if the cloud wants to manipulate the result, it has to guess the random integers, the probability to succeed is only $1/b$. In fact, if we outsource $\mathcal{C}(U, A_1, L)$ and $\mathcal{C}(U, A_2, L)$ in a random order, we can further reduce the probability for the cloud to guess the correct randoms to $1/2b$. According to Definition 3.4, ExpSOS is at least $(1-1/2b)$-verifiable.

The upper bound b is a security parameter that measures the confidence of the end-user about the returned result. In practical computation outsourcing systems, the cloud would be severely punished if cloud manipulation is detected. Therefore, the benefit for the cloud to cheat would be hardly justifiable in this setting.

3.5.2 Complexity Analysis

We utilize outsourcing of modular exponentiation as a representative to analysis complexity. The analysis can be applied to scalar multiplication similarly. The essence of ExpSOS is to limit the number of modular multiplications for the end-user to compute modular exponentiation with the aid of the cloud. In our analysis, we utilize the number of modular multiplications, denoted as π, as a measurement. To calculate $u^a \pmod N$, the number of multiplications is $\pi = \frac{3}{2} l_a$, where l_a is the bit length of a [43]. Therefore, in calculating the modular exponentiation $u^a \pmod N$, $l_a \approx \log_2 a$ and $\pi \approx \frac{3}{2} \log_2 a$.

In ExpSOS, under the HCS model, to calculate U, A and L, the end-user needs 3 multiplications. We notice that when the end-user knows the factors of N, it is computationally easy to calculate $\varphi(N)$. For example, when N is a prime, $\varphi(N) = N - 1$. Moreover, the calculation of $\varphi(N)$ is a one-time process. The computational overhead for calculating $\varphi(N)$ is negligible especially when the end-user outsources modular exponentiation multiple times. Thus, under HCS model, we have $\pi_{HCS} = 3$. Hence, the computational gain from outsourcing is $\alpha_{HCS} = \pi / \pi_{HCS} = \frac{1}{2} \log_2 a$. From Definition 3.3, ExpSOS is $\frac{1}{2} \log_2 a$-efficient under the HCS model.

Under the MS model, the calculation of L, U, A_1, A_2 will take 4 multiplications. In the verification scheme, the end-user has to calculate $R_1^{t1} \pmod N$ and $u^{t2} \pmod N$. Thus, $\pi_{MS} = 4 + \frac{3}{2} \log_2 t_1 + \frac{3}{2} \log_2 t_2 + 1$. Since t_1 and t_2 are upper-bounded by b, we have

$\log_2 t_1 + \log_2 t_2 \leq 2 \log_2 b$. Hence the computational gain from outsourcing is

$$\begin{aligned} \alpha &= \frac{\pi}{\pi_{MS}} \\ &= \frac{\frac{3}{2}\log_2 a}{5 + \frac{3}{2}\log_2 t_1 + \frac{3}{2}\log_2 t_2} \\ &\geq \frac{\frac{3}{2}\log_2 a}{5 + 3\log_2 b} \\ &\approx \frac{1}{2}\log_b a. \end{aligned}$$

Thus under the MS model, ExpSOS is at least $\frac{1}{2}\log_b a$-efficient.

3.5.3 Trade-Off between Computation and Security

The above security and complexity analysis reveal the trade-off between computational overhead and security. In the MS model, ExpSOS is at least $\frac{1}{2}\log_b a$-efficient and $(1 - 1/2b)$-verifiable. Both measurements relate to the same parameter b. On one hand, b is the upper bound of the computational overhead that the end-user can tolerate. On the other hand, b reveals the confidence of the end-user about the returned result which is also regarded as the security level of the result. When b increases, the end-user has to carry out more computation. However, the probability that the end-user can verify the validity of the result also increases.

Thus, the proposed ExpSOS is cost-aware in the sense that it enables the end-user to have the flexibility to choose the most suitable outsourcing scheme according to its computational constraint and security demand. This is important especially when the end-users vary in computational power and security demands. It also makes ExpSOS widely applicable.

3.6 Applications

The proposed ExpSOS is able to conceal the base, the exponent and the module of the modular exponentiation $u^a \pmod N$. It can also be used to conceal the base point P and multiplier s of the scalar multiplication sP. With this feature, the parameters (private or public) within the cryptosystem are totally concealed from the outside, especially the cloud. Thus, the cryptosystem is isolated from the outsourced system. In this sense, ExpSOS can be regarded as a black box that takes as input $\{u, a, N, b\}$ and creates the output $u^a \pmod N$ as $\mathsf{ExpSOS}(u, a, N, b) \rightarrow u^a \pmod N$, where b is security parameter selected by the end-user. The end-user will have a performance gain of at least $\frac{1}{2}\log_b a$ and can verify the validity of the result with probability $1 - \frac{1}{2b}$.

In this section, we will explore efficient outsourcing of exponential operations in some typical cryptographic protocols to the cloud.

3.6.1 Outsourcing Inner Product Encryption for Biometric Authentication

A practical Inner Product Encryption (IPE) scheme was recently introduced in [44]. In IPE, given the master secret key msk, the encryption function $\mathsf{Encrypt}(\mathsf{msk}, \mathbf{y})$ encrypts a vector $\mathbf{y}$ as

$$\mathsf{ct}_{\mathbf{y}} = (C_1, C_2) = (g_2^{\beta}, g_2^{\beta \cdot \mathbf{y} \cdot \mathbf{B}^{\star}}),$$

where g_2 is a generator of the underlying bilinear group $\mathbb{G}_2$, $\mathbf{B} \leftarrow \mathbb{GL}_n(\mathbb{Z}_q)$ is a randomly generated matrix and $\mathbf{B}^* = {}_{\det}(\mathbf{B}) \cdot (\mathbf{B}^{-1})^T$. Similarly, a key generation function $\mathsf{KeyGen}(\mathsf{msk}, \mathbf{x})$ will generate a key $\mathsf{sk}_{\mathbf{x}}$ associated with a vector $\mathbf{x}$. Given $\mathsf{sk}_{\mathbf{x}}$, $\mathsf{ct}_{\mathbf{y}}$ and the decryption key,

the decryption process will produce the inner product $\mathbf{z} = \mathbf{x} \cdot \mathbf{y}$. If $\mathbf{x}$ and $\mathbf{y}$ are binary vectors, the inner product $\mathbf{z}$ is actually the Hamming distance between $\mathbf{x}$ and $\mathbf{y}$. Thus, IPE provides a way to evaluate the distance between two vectors from the ciphertext domain. Due to this feature, IPE is a promising technique to protect biometric templates in biometric authentication systems [44].

In biometric authentication systems, each user is identified by some biometric template (e.g., iris) represented by a vector $\mathbf{x}$. During enrollment, an authority encrypts such template $\mathbf{x}$ as $\mathsf{sk}_{\mathbf{x}}$ and stores it in a database together with the user's identity as a record $\langle ID, \mathsf{sk}_{\mathbf{x}} \rangle$. When another user with template $\mathbf{y}$ wants to authenticate himself, the authority encrypts the template as $\mathsf{ct}_{\mathbf{y}}$ and sends it to the database server. The decryption obtains the inner product $\mathbf{z} = \mathbf{x} \cdot \mathbf{y}$, which is generally the hamming distance. If $\mathbf{z}$ is within a pre-defined threshold, the user is authenticated. In the following, we show that our ExpSOS scheme can be well applied to biometric authentication system to outsource the encryption process.

Normally, the KeyGen and Encrypt functions are executed in a resource-constrained device such as a scanner to collect and encrypt users' biometric data. In the following, we take the Encrypt function as an example to illustrate the outsourcing procedure, which can also be applied to the function KeyGen. Note that bilinear operation is typically implemented based on elliptic curves. Thus the underlying group $\mathbb{G}_1$ and $\mathbb{G}_2$ are sets of points on elliptic curves. The exponentiations are actually scalar multiplications of the base point g_2. As a result, we can utilize Protocol 3 as the basic scheme to outsource Encrypt. To simplify notations, we use a function $\mathcal{T}(P, E) \rightarrow P'$ to represent the process in Protocol 3 to transform a point $P = (x_1, y_1, z_1)$ to its disguised form $P' = (x'_1, y'_1, z'_1)$. Similarly, a function $\mathcal{T}^{-1}(P') \rightarrow P$ will recover P from P'. Let m be the order of group $\mathbb{G}_2$. The main computation involved in Encrypt are exponentiations of the same base point g_2. Let $\beta \cdot \mathbf{y} \cdot \mathbf{B}^* = (\beta_1, \beta_2, \cdots, \beta_n)$.

We can transform g_2 to its disguised form G_2 according to Protocol 3. For each β_i, we transform β_i to $\beta_i' = \beta_i + k_i m$, where k_i is a randomly generated integer. Then, G_2 and β_i' are outsourced to the cloud for computation. The result can be easily recovered by applying the function $\mathcal{T}^{-1}$. For result verification, we can jointly verify the exponentiations since they share the same base. To be specific, we can randomly select a set $I \subset \{1, 2, \cdots, n\}$ and calculate $\sum_{j \in I} \beta_j$. Then a new exponent for verification is computed as $\delta = t_1 \sum_{j \in I} \beta_j + t_2$. The result verification is done by locally calculating exponentiations with small exponents t_1 and t_2. The protocol to outsource Encrypt is presented in Protocol 5. It is obvious that in the original Encrypt function, the major computation burden lies in the computation of $(n + 1)$ exponentiations, where n is the length of the vector. However, with ExpSOS, the end-user only needs to transform each entry in the vector as well as the base to the disguised form. The transformation for each entry only takes one multiplication. In result verification, the end-user only needs to conduct two exponentiations with small exponents and limited number of multiplications. In this sense, ExpSOS can achieve significant performance gain for the end-user.

3.6.2 Outsourcing Identity Based Encryption

Identity Based Encryption (IBE) system is proposed to alleviate the process of public key certification in traditional public key cryptosystems. In IBE system, a user can utilize his identity such as his email address as the public key. Then a trusted authority will generate and distribute private key to the message receiver. The idea of IBE was initialized by Shamir in [45]. A practical IBE system was proposed in [46] based on bilinear pairing on elliptic curves.

In an implementation of IBE system [35, Chapter 5], the public parameters are an elliptic

Protocol 5 Outsourcing Encryption of IPE

Input: $E, B = (\beta, \beta_1, \beta_2, \cdots, \beta_n), g_2$

Output: $c = g_2^{\beta}, c_i = g_2^{\beta_i}, C = (c, c_1, \ldots, c_n), \Lambda =$ True or False.

Key Generation $\mathsf{KenGen}(1^\lambda, m) \to L$:

1: E generates a large prime p and calculate $L = pm$.
2: The public key is $K_p = \{L\}$, and the private key is $K_s = \{p, m\}$.

Problem Transformation $\mathcal{T}(g_2, B, E) \to G_2$:

1: E calls function $\mathcal{T}$ in Protocol 3 to transform point g_2 to G_2.
2: E selects random integers t_1, t_2, k and $(k_1, k_2, \cdots, k_n)$ as the ephemeral key with the constraint that $t_1, t_2 \leq b$.
3: E calculates $\beta' = \beta + km$ and $\beta_i' = \beta_i + k_i m$ for $i = 1, 2, \cdots, n$.
4: E randomly selects a set $I \subset \{1, 2, \cdots, n\}$ and calculate $\delta = t_1 \sum_{j \in I} \beta_j + t_2$.
5: E outsources $(G_2, \delta, B' = (\beta', \beta_1', \beta_2', \cdots, \beta_n'))$ to the cloud.

Cloud Computation $\mathcal{C}(G_2, \delta, B') \to C$:

1: S computes $C' = (c_r' = G_2^{\delta}, c' = G_2^{\beta'}, c_1' = G_2^{\beta_1'}, \ldots, c_n' = G_2^{\beta_n'})$.
2: S returns C to E.

Result Verification $\mathcal{V}(C) \to \Lambda$:

1: E checks $\mathcal{T}^{-1}(c_r') = (\prod_{j \in I} c_j')^{t_1} \cdot g_2^{t_2}$.
2: If the equality holds, set $\Lambda \leftarrow$ True. Otherwise, set $\Lambda \leftarrow$ False.

Result Recovery $\mathcal{R}(C') \to C$:

1: E recovers the result as $C = (c = \mathcal{T}^{-1}(c'), c_1 = \mathcal{T}^{-1}(c_i'), \ldots, c_n = \mathcal{T}^{-1}(c_n'))$.

curve $E(\mathbb{F}_p)[m]$ and a base point $P \in E(\mathbb{F}_p)[m]$. Also, the trusted authority will publish his own public key $P_T \in E(\mathbb{F}_p)[m]$. The parameters are known to the authenticated users in the system. We assume that a user Alice uses the hash of her own identity to generate the public key which is a point on the elliptic curve, that is $P_A \in E(\mathbb{F}_p)[m]$. For any other user Bob who desires to send a message M to Alice, he will conduct the following encryption process:

1. Bob selects a random integer $r \in Z_m$;

2. Bob computes $C_1 = rP$;

3. Bob computes $C_2 = M \oplus H(e(P_A, P_T))^r$;

4. Bob sets the ciphertext as $C = (C_1, C_2)$.

In the above encryption algorithm, $e(P_A, P_T)$ denotes the pairing between public points P_A and P_T and $H(\cdot)$ is a hash function. We note that both the input and output of the pairing $e(P_A, P_T)$ are public. Thus, the end-user Bob can obtain the pairing result denoted as $g = e(P_A, P_T)$. At the end, we can see that the computational burden for Bob lies in the scalar multiplication rP and the modular exponentiation $g^r \pmod p$. We summarize the outsourcing of IBE as in Protocol 6.

Protocol 6 Secure Outsourcing of Identity Based Encryption

Input: $E, P = (x, y, z)$, r, $g = e(P_A, P_T)$
Output: $C_1 = rP$, $C_2 = H(g)^r$

Key Generation $\mathsf{KeyGen}(1^\lambda, p) \to L$:

1: Bob selects a large prime q and calculates $L \leftarrow pq$.

Problem Transformation $\mathcal{T}(p, r, L, P) \to (E', r_1, r_2, P', H(g), L)$:

1: Bob generates temporary key $k_1, k_2, k_3, k_4, k_5, t_1, t_2$ with $t_1, t_2 < b$.
2: Bob calculates $r_1 \leftarrow (r + k_1 p) \pmod L$, $r_2 \leftarrow (t_1 r + t_2 + k_2 p) \pmod L$, $x' \leftarrow (x + k_3 p) \pmod L$, $y' \leftarrow (y + k_4 p) \pmod L$, $z' \leftarrow (z + k_5 p) \pmod L$. Bob sets $P' \leftarrow (x', y', z')$.
3: Bob outsources $(E', r_1, r_2, P', H(g), L)$ to the cloud, where E' is the transformed elliptic curve defined in Protocol 3 and $H(g)$ is the hash value of g.

Cloud Computation $\mathcal{C}(E', r_1, r_2, P', H(g), L) \to (Q_1, Q_2, R_1, R_2)$:

1: S calculates $Q_1 \leftarrow r_1 P'$, $Q_2 \leftarrow r_2 P'$, $R_1 \leftarrow H(g)^{r_1}$ and $R_2 \leftarrow H(g)^{r_2}$.
2: S returns the results (Q_1, Q_2, R_1, R_2) to Bob.

Result Verification $\mathcal{V}(t_1, t_2, H(g), p, Q_1, Q_2, R_1, R_2) \to \Lambda$:

1: Bob verifies the results by checking $R_1^{t_1} \cdot H(g)^{t_2} = R_2 \pmod p$ and $t_1 Q_1 + t_2 P = Q_2 \pmod p$, where the modular is applied coordinate-wise.

Result Recovery $\mathcal{R}(Q_2, M, R_2, p) \to (C_1, C_2)$:

1: Bob recovers the results $C_1 \leftarrow Q_2 \pmod p$ and $C_2 \leftarrow M \oplus R_2 \pmod p$.

From the above two applications, we can summarize some techniques in designing secure outsourcing scheme utilizing the outsourcing of exponential operation as a building block.

- It is more efficient and secure to share some common parameters in different subroutines of the outsourcing process. In this way, we can not only reduce the computational

overhead, but alos expose less information to the cloud.

- When outsourcing modular exponentiation with the same base, the computational overhead can be reduced by jointly verifying the result. This is demonstrated in the outsourcing of IPE.

- When making multiple queries to the cloud, the end-user can randomize the order of queries to increase verifiability.

3.7 Performance Evaluation

To the best of our knowledge, previous research on secure outsourcing of cryptographic computations mainly focuses on modular exponentiation. In this section, we first compare ExpSOS with three existing works on secure outsourcing of modular exponentiation. Then we give some numerical results to show the efficiency of ExpSOS.

3.7.1 Performance Comparison

Secure outsourcing of cryptographic computations, especially modular exponentiation, has been a popular research topic [10, 11, 38, 40–42, 47–51]. For instance, the authors in [51] proposed a secure outsourcing scheme for modular exponentiation with variable-exponent fixed base and fixed-exponent variable-base under single untrusted server model. However, the base is known to the server. In [10], the authors considered outsourcing variable-base variable-exponent modular exponentiation to two untrusted servers. Following this work, the authors in [11] improved the scheme in [10] in both efficiency and verifiability. Then, the authors in [38] made further improvement by reducing the two-server model to one

single untrusted server model. The existing research in this area can be classified into three categories based on the supporting models, i.e, HCS model ([40]), MM model ([10, 11, 41]) and MS model ([38, 42]). In the following, we compare our ExpSOS with the six typical schemes in these three categories.

3.7.1.1 HCS Model

The most representative example in this category is the scheme introduced in [40]. It considers outsourcing u^a to a single honest-but-curious server, where $u \in \mathbb{G}$ and $a \in \mathbb{Z}_p$. It does not provide the ability to verify the result. The outsourcing process includes three steps. First, the end-user utilizes a Rand function to generate two pairs (k_1, g^{k_1}) and (k_2, g^{k_2}), where k_1 and k_2 are selected randomly. The base u is disguised as $v = u \cdot g^{k_1}$ and the exponent a is divided into two parts (a_0, a_1) such that $a = a_1 \cdot T + a_0$, where $T = \lceil \sqrt{p} \rceil$. In the second step the end-user makes two queries to the server as $\mathsf{S}(v, T) = v^T \rightarrow h$ and $\mathsf{S}(g, -ak_1 - k_2 \pmod p) = g^{-ak_1 - k_2} \rightarrow h_2$. The end-user then utilizes Protocol 1 in [40] to calculate $h_1 = v^{a_0} \cdot h^{a_1}$. In the third step the result is recovered as $h_1 h_2 g^{k_2} = u^a$.

It is clear that the computational bottleneck lies in the computation of $h_1 = v^{a_0} \cdot h^{a_1}$, where the bit length of a_0 and a_1 can be half of that of a. Analysis shows that it takes l multiplications to determine h_1, where l is the maximum of $\log a_0$ and $\log a_1$. In addition, the scheme in [40] also requires the end-user to call Rand function twice and then conduct Euclidean division to obtain a_0 and a_1.

3.7.1.2 MM Model

The most representative examples for MM model include [10, 11, 41]. MM model considers outsourcing modular exponentiation to two untrusted servers S_1 and S_2 and it is assumed

that the two-server do not collude which corresponds to our MM model. In these schemes, a subroutine Rand is utilized to generate random modular exponentiation pairs. Specifically, on input a base $g \in \mathbb{Z}_p^*$, the subroutine Rand will generate random pairs in the form of $(\theta, g^\theta \bmod p)$, where θ is a random number in $\mathbb{Z}_p^*$. Then the end-user can make queries to Rand and each query will return a random pair to the end-user. Typically, the subroutine Rand is implemented via two different methods. One method is that a table of random pairs is pre-computed from a trusted server and stored at the end-user. Whenever the end-user needs to make a query to Rand, it just randomly draws a pair from the table. The critical problem with this method is that it will take a lot of storage space from the end-user. Specifically, a random pair will take 2ℓ space, where ℓ is the bit length of p. In addition, to make the generation of the pairs look random, the table size should be large. As a result, the storage overhead becomes unacceptable for the resource-constrained end-users. The other method is to utilize some pre-processing techniques such as the BPV generator [49] and the EBPV generator [50]. To generate one random pair, the EBPV generator takes $\mathcal{O}(\log^2 \ell_a)$ modular multiplications, where ℓ_a is the bit length of the exponent.

The scheme proposed in [10] can be briefly summarized as follows. First, the end-user runs Rand 6 times to obtain random pairs $(\alpha, g^\alpha), (\beta, g^\beta), (t_1, g^{t1}), (t_2, g^{t2}), (r_1, g^{r1}), (r_2, g^{r2})$. Then u^α can be written as

$$u^a = v^b f^{a-b} \left(\frac{v}{f}\right)^{a-b} \left(\frac{u}{v}\right)^{d} \left(\frac{u}{v}\right)^{a-d},$$

where $v = g^\alpha, b = \frac{\beta}{\alpha}$, f and d are random integers. The end-user then makes queries in random order to the cloud server S_1 to derive $Q_1^1 = \left(\frac{u}{v}\right)^d, Q_1^2 = f^{a-b}, Q_1^3 = (g^{r1})^{\frac{t_1}{r_1}}, Q_1^4 = (g^{r2})^{\frac{t_2}{r_2}}$. Similarly, the end-user makes queries to the second cloud server S_2 to derive

$Q_2^1 = \left(\frac{u}{v}\right)^{a-d}, Q_2^2 = \left(\frac{v}{f}\right)^{a-b}, Q_2^3 = (g^{r1})^{\frac{t_1}{r1}}, Q_2^4 = (g^{r2})^{\frac{t_2}{r2}}$. The result can be recovered as $u^a = g^\beta \cdot Q_1^1 \cdot Q_1^2 \cdot Q_2^1 \cdot Q_2^2$. The result verification is carried out by checking whether $Q_1^3 = Q_2^3 = g^{t1}$ and $Q_1^4 = Q_2^4 = g^{t2}$. We note that the end-user needs to make queries to each server S_1 and S_2 for four times, among which the first two are computation queries and the other two are test queries. Since the test queries and the computation queries are independent, the servers can potentially compute the test queries honestly but cheat in the computation queries. The authors address this problem by sending out the queries in random order. The verifiability of this scheme is $\frac{1}{2}$. In the outsourcing process, E has to run the subroutine Rand 6 times, make 9 modular multiplications (MMul) and 5 modular inversions (MInv), where Rand has a complexity of $\mathcal{O}(\log^2 n)$ MMul and n is the bit length of the exponent.

Based on [10], the authors in [11] made some improvement by reducing the computational overhead to 5 Rand, 7 MMul and 3 MInv and the queries to the two servers are reduced to 6 times in total. Moreover, the verifiability is improved to $\frac{2}{3}$.

Unlike [10, 11], the scheme in [41] utilized the *Chinese Remainder Theorem* to disguise the secret values. To be specific, the group $\mathbb{Z}_p$ is extended to $\mathbb{Z}_n$, where $n = pr_1r_2$ and r_1 and r_2 are large primes. Together with some randomly generated values, the secret values u and a are transformed to the corresponding elements in $\mathbb{Z}_n$. The scheme also employs the function Rand to generate random exponentiation pairs (t_i, g^{t_i}). Result verification is achieved by comparing the returned results from two independent servers. Compared with [10, 11], the advantage of the scheme is that it can achieve checkability 1 for every query result. Complexity analysis shows that to outsource one modular exponentiation, the end-user has to carry out 7 multiplications in addition to executing the Rand function 4

times.

Protocol 7 ExpSOS under MM Model

Input: $N, u, a \in \mathbb{R}_N$.
Output: $R_0 = u^a \pmod N$, $\Lambda = \{\mathsf{True}, \mathsf{False}\}$.
Key Generation$(1^\lambda, p) \to L$:

1: E generates a large prime number p and calculate $L \leftarrow pN$. The public key is $K_p = \{L\}$ and the private key is $K_s = \{p, N\}$.
2: E selects random integers $r, k \in \mathbb{Z}_N$ as the temporary key.

Problem Transformation $\mathcal{T}(a, u, L) \to (A, U)$:

1: E calculates $A \leftarrow a + k\varphi(N)$ and $U \leftarrow (u + rN) \pmod L$.
2: E then outsources $\{U, A, L\}$ to both cloud servers S_1 and S_2.

Cloud Computation $\mathcal{C}(A, U, L) \to (R_1, R_2)$:

1: S_1 computes $R_1 \leftarrow U^A \pmod L$ and S_2 computes $R_2 \leftarrow U^A \pmod L$.
2: The results R_1 and R_2 are returned to E.

Result Verification $\mathcal{V}(R_1, R_2, N) \to \Lambda$:

1: E checks whether $R_1 = R_2 \pmod N$. Set $\Lambda \leftarrow \mathsf{True}$ if the equality holds; otherwise set $\Lambda \leftarrow \mathsf{False}$.

Result Recovery $\mathcal{R}(R_1, N) \to R$:

1: E recovers the result as $R \leftarrow R_1 \pmod N$.

3.7.1.3 MS Model

The most representative examples for MS model include [38,42]. MS model employs only one single server. The scheme in [38] can be summarized as follows. First, the end-user runs Rand 7 times to obtain random pairs $(\alpha_1, g^{\alpha_1}), (\alpha_2, g^{\alpha_2}), (\alpha_3, g^{\alpha_3}), (\alpha_4, g^{\alpha_4}), (t_1, g^{t_1}), (t_2, g^{t_2}), (t_3, g^{t_3})$. Then it calculates $c = (a - b\chi) \pmod p$, $\omega = u/\mu_1, h = u/\mu_3$, and $\theta = (\alpha_1 b - \alpha_2)\chi + (\alpha_3 c - \alpha_4) \pmod p$, where χ, b are randomly selected and $\mu_i = g^{\alpha_i}$, for $i = 1, 2, 3, 4$. The end-user then queries to a single cloud server S to compute $Q^{(i)}$ for $i = 1, 2, 3, 4$, where $Q^{(1)} = \left(g^{t_1}\right)^{\frac{\theta}{t_1}}, Q^{(2)} = \left(g^{t_2}\right)^{\frac{t_3 - \theta}{t_2}}, Q^{(3)} = \omega^b, Q^{(4)} = h^c$. The result is recovered as $u^a = (\mu_2 \cdot Q^{(3)})^\chi \cdot Q^{(1)} \cdot \mu_4 \cdot Q^{(4)}$. The result verification is carried out by checking whether $Q^{(1)} \cdot Q^{(2)} = g^{t_3}$ is true. Similarly, the queries can be divided into test queries and computation queries. As a result, the cloud can compute honestly on the test queries and cheat on

the computation queries. Thus, due to the random order of the queries, the verifiability of this scheme is $\frac{1}{2}$. We note that in the result recovery process, the end-user has to compute an exponentiation $(\mu_2 \cdot \omega^b)^\chi$ which takes $\frac{3}{2}\log\chi$ multiplications. The whole scheme will take 7 Rand, $12 + \frac{3}{2}\log\chi$ MMul, 4 MInv and make 4 queries to the cloud server. In comparison, ExpSOS can avoid inversion and only needs $(5 + 3\log b)$ MMul, where b is a small integer.

The scheme in [42] also investigates secure outsourcing of modular exponentiation with variable base and variable exponent to a single untrusted server. The main drawback of [42] is its large number of queries $(l+k+2)$ made to the server, which introduces high computational and communication overhead. In fact, the scheme in [42] first utilizes a sub-protocol SubAlg to outsource exponentiations with a fixed base. To support the case of variable bases, the base is split into multiple sets of different sizes. The scheme in [42] also employs the Rand five times to generate exponentiation pairs (t_i, g^{t_i}). Similar to our work, in the result verification phase, [42] also needs to conduct an exponentiation with a small exponent to achieve a checkability of $\frac{1}{c(c-1)}$. In addition, the end-user has to conduct $l + k + 8\log c + 38$ multiplications and 1 inversion operation. As shown in an example given in [42], when $l = k = 29$ and $c = 4$, the number of multiplications reaches 100 and the end-user needs to make 60 queries to the server. Also, as pointed out in [2], one modular inversion is about 100 times slower than a modular multiplication. This further increases the complexity of the scheme in [42].

3.7.1.4 Performance Comparison of ExpSOS with the Existing Schemes

In comparison, ExpSOS under MM model can be modified as in Protocol 7. Since the cloud servers S_1 and S_2 do not collude, the only way to make the equality condition satisfied is that S_1 and S_2 both compute honestly. Thus the verifiability is 1. Moreover, in this process,

Table 3.1: Performance Comparison

Scheme	Model	Pre-Processing	Multiplication	Inversion	Comm. Cost	Verifiability
[40]	HCS	2 Rand	$2\ \mathcal{O}(\mathsf{Rand}) + l(1+o(1))$	0	6ℓ	Not Applicable
[10]	MM	6 Rand	$6\ \mathcal{O}(\mathsf{Rand}) + 9$	5	24ℓ	1/2
[11]		5 Rand	$5\ \mathcal{O}(\mathsf{Rand}) + 7$	3	18ℓ	2/3
[41]		4 Rand	$4\ \mathcal{O}(\mathsf{Rand}) + 7$	0	26ℓ	1
[38]	MS	7 Rand	$7\ \mathcal{O}(\mathsf{Rand}) + \frac{3}{2}\log\chi + 12$	4	12ℓ	1/2
[42]		5 Rand	$5\ \mathcal{O}(\mathsf{Rand}) + l + k + 8\log c + 38$	1	$3(l+k+2)\ell$	$1 - 1/c(c-1)$
ExpSOS	HCS	Not Required	3	0	6ℓ	Not Applicable
	MM	Not Required	3	0	12ℓ	1
	MS	Not Required	$5 + 3\log b$	0	10ℓ	$1 - 1/2b \approx 1$

we successfully avoid inversion that is considered much more expensive than multiplication in field operations. The total computational overhead is only 3 MMul.

To measure the communication overhead, we utilize the total length of bits to be transmitted between the end-user and cloud server in order to outsource one exponentiation. Let ℓ be the bit length of N. Without loss of generality, we assume that it takes ℓ bits to transmit one element in $\mathbb{Z}_N$. In our scheme, the elements in $\mathbb{Z}_N$ is transformed to elements in $\mathbb{Z}_L$, where $L = pN$. Since p is comparable to the divisors of N, the bit length of L is at most 2ℓ. Thus, we can assume that it takes 2ℓ bits to transmit one element in $\mathbb{Z}_L$ (worse case). Following this analysis, the communication cost for each scheme is given in Table 3.1. It shows that our scheme can achieve the highest communication efficiency.

In terms of security, we have shown that ExpSOS can successfully conceal the base, exponent and the modulus of the modular exponentiation. It is computationally infeasible for the cloud to derive any key information from the disguised problem. In comparison, all the above three schemes [10, 11, 38] can only conceal the exponent and base while the modulus is exposed to the cloud. Thus ExpSOS can provide much-improved security. Moreover, the three schemes in [10], [11] and [38] achieve verifiability of $\frac{1}{2}$, $\frac{2}{3}$ and $\frac{1}{2}$ respectively. In comparison, the verifiability of ExpSOS is $1 - \frac{1}{2b}$ that is close to 1. This means that the end-user is more confident about the results returned by the cloud. Furthermore, the security

of the schemes in [10] and [11] relies on the assumption that the two cloud servers will not collude. The scheme [38] and the proposed ExpSOS are applicable to one single untrusted server hence eliminating the non-collusion assumption.

The comparison of ExpSOS and the schemes in [10,11,38] is summarized in Table 3.1. We can see that the proposed ExpSOS outperforms other schemes in both computational complexity and security. ExpSOS also makes the least queries to the cloud that will introduce the least communication overhead. Moreover, ExpSOS is cost-aware in computational overhead and security such that the end-users can select the most suitable outsourcing scheme according to their own constraints and demands. Also, ExpSOS can be modified such that it is applicable to HCS, MM and MS model.

3.7.2 Numeric Results

In this section, we compare the performance of ExpSOS for modular exponentiation with the schemes in [11, 38] through simulation in mobile phones. The computation of both the end-user and the cloud server is simulated in the same phone Samsung GT-I9100 with Android 4.1.2 operating system. The CPU is Dual-core 1.2 GHz Cortex-A9 with 1 GB RAM. In the outsourcing process, we focus on the computational gain, denoted as τ, of the outsourcing. We measure the local processing time (t_0) to compute the modular exponentiation $u^a \pmod N$ without outsourcing and the local processing time (t_s) with outsourcing which includes the problem transformation, result recovery, and result verification. To measure the performance of ExpSOS under different levels of complexity, we let the size of the ring l_N vary from 128 bits to 1024 bits. Also, to show the cost-awareness of ExpSOS, we let the size of the security parameter l_b vary from 4 bits to 16 bits. The processing time is averaged over 1000 independent rounds. The numeric result is shown in Table 3.2 where each number

Table 3.2: Numeric Results

l_N (bits)	ExpSOS (l_b (bits))									[11]		[38]	
	4			8		12		16					
	t_0 (ms)	t_s (ms)	τ	t_s (ms)	τ	t_s (ms)	τ	t_s (ms)	τ	t_s (ms)	τ	t_s (ms)	τ
128	1358	87	15.6	216	6.3	321	4.2	397	3.4	171	7.9	827	1.6
256	2554	89	28.6	244	10.5	346	7.4	459	5.6	172	14.9	917	2.8
384	4095	127	32.3	249	16.5	358	11.4	463	8.8	180	22.8	1004	4.1
512	7837	134	58.6	281	27.9	399	19.6	496	15.8	242	32.4	1097	7.1
640	10991	146	75.0	288	38.2	423	26.0	627	17.5	268	40.9	1269	8.7
768	11427	148	77.2	295	38.7	433	26.4	642	17.8	273	42.0	1377	8.3
896	17445	158	110.2	317	54.9	451	38.7	680	25.6	346	50.4	1467	11.9
1024	20235	174	116.2	329	61.5	504	40.1	739	27.4	355	57	1569	12.9

stands for the average processing time for 100 rounds. We can see that when the size of the ring l_N increases, the performance gain τ also increases for the same security parameter b. This means that when the original problem is more complex, ExpSOS would have a better performance. The reason is that the complexity of modular exponentiation depends on the number of multiplications that is positively correlated to the logarithm of the size of the ring l_N. However, in ExpSOS the local processing takes almost the same number of multiplications for a fixed security parameter b. We can also see that there exists a trade-off between security and computational overhead. When b increases, the computational overhead increases accordingly. Since the verifiability is $1 - \frac{1}{2b}$, a bigger b means better security guarantees.

The numeric results also demonstrate the high efficiency of ExpSOS compared to the schemes in [11, 38], which coincides with our theoretical analysis. We note that to achieve the same checkability, the schemes in [11,38] should be compared with ExpSOS with $l_b = 4$. In the simulation, the implementation of the Rand function is rather simplified and it only requires the end-user to conduct an average of 5 multiplications. From the simulation, we can see that the performance of the scheme in [38] is dominated by the computation of $(\cdot)^\chi$, where χ has 64 bits as recommended in [38]. Our high efficiency comes from the fact that

the required number of multiplications is much less and we do not need the end-user to call the Rand function and compute modular inversion.

Chapter 4

Secure Fine-Grained Access Control of Mobile User Data through Untrusted Cloud

4.1 Introduction

With the fast development of cloud computing, there is a trend for mobile users to upload and share their data through the cloud platform for unrestricted access regardless of the time and location. A typical scenario would be the Personal Healthcare System [52], where patients upload their personal health records to a public cloud that are accessible to the authorized physicians. In these scenarios, data owners are concerned about the security and privacy of their own data. On one hand, the untrusted cloud should be prevented from knowing the content of data. On the other hand, data owners should control who can access the data.

Attribute Based Encryption (ABE) cryptosystem turns out to be a promising candidate to address the above issue. As a generalization of Identity Based Encryption (IBE) [46, 53], ABE can be categorized into two types: Key-Policy ABE (KP-ABE) and Ciphertext-Policy ABE (CP-ABE) [54–56] depending on where the policy is specified. A typical CP-ABE

system consists of message senders, message receivers and an Attribute Authority (AA) that is responsible for generating secret keys. Each user (sender or receiver) is tagged with a set of attributes (e.g., {Student, ECE, Enrolled}). In message encryption, the sender can specify an access policy (e.g., {Professor $\wedge$ (ECE $\vee$ CSE)}) associated with the ciphertext. As a result, only those receivers whose attributes (e.g., {Professor, ECE}) satisfy the policy can decrypt the message. In contrast, KP-ABE tags ciphertext with sets of attributes and associates policies with decryption keys. The fact that CP-ABE enables fine-grained access control of encrypted data makes it suitable for various cloud based applications.

However, one concern in ABE is its relatively high computational complexity in encryption and decryption. For instance, message encryption will take two modular exponentiations for each attribute in the policy [55]. And a message receiver will conduct two pairing operations in bilinear groups for each attribute in decryption. As a result, the number of modular exponentiations and pairing operations involved in the encryption and decryption process is linear to the number of attributes in the access policy. The high computational overhead of ABE seems to be prohibitive for resource constrained mobile devices. In some recent research [57, 58], the performance of ABE has been measured on mobile phones. It is shown that, at a security level of 128 bits[1], the average execution time for encryption with 10 attributes on android smartphones is around 10 seconds. Although it is feasible to implement ABE on mobile devices, it is highly inefficient in terms of execution time and energy consumption.

To address this problem, some researchers [60] propose an online/offline computation scheme where the message is pre-encrypted without specifying a policy during the offline phase. Then based on the pre-computed results, mobile users can encrypt messages offline

[1]The description of different security levels can be found in [59]

by a specified policy with relatively low computational overhead. Some other research works focus on secure outsourcing of ABE systems [61–63]. However, they either focus on a specific access structure [62] or rely on a rather strong security assumption of cloud [63].

In this paper, we propose to outsource the whole access control functionality to the cloud. This is achieved based on a key observation that the message encryption part and access control part are highly detached in ABE. To be specific, the ciphertext of ABE can be roughly divided into a payload section and a control section. The payload section is simply message encryption $\mathsf{Enc}(m, s)$ using a secret key s. The control section is a linear secret sharing scheme that shares the secret key s to certain parties (i.e., attributes). Roughly speaking, in the decryption process, a receiver with certain attribute set that satisfies the policy can recover the secret key s. Thus the receiver can further decrypt the message m.

In our proposed scheme, we let the cloud generate the control section of the ciphertext and a mobile user only needs to conduct message encryption. However, it is not a trivial task. First, to share the secret key s, the cloud has to know the secret beforehand. In this case, the cloud can easily decrypt the payload section. Second, when the cloud is fully in charge of the access control, how to enforce the cloud to deny invalid access is a critical problem. Because in some situations, the cloud may be compromised by malicious users or the cloud may actively collude with data users. We address these problems by exploring the linear property of secret sharing schemes. Basically, the mobile users encrypt message with a secret s_1 while the cloud distributes a different secret s_2 through a secret sharing scheme. The control section in the ciphertext and the decryption algorithm are then modified in such a way that only valid users can recover the secret s_1.

In summary, our proposed scheme enables mobile users to conduct only normal encryption and specify an access control policy on the ciphertext. The access control is then honestly

fulfilled by the cloud. Moreover, the cloud will conduct partial decryption based on which the data users can recover the message through normal decryption. In this way, attribute based encryption becomes transparent to mobile users and cloud can provide access control of encrypted data as a service.

4.2 System Model and Threat Model

4.2.1 System Model

We consider a file sharing system consisting of four parties: an Attribute Authority (AA), data owners, data users (who could be data users simultaneously) and the cloud. The AA is responsible for setting up the system, authenticating users' attributes and generating secret keys when users request keys with their attributes. Data owners desire to upload their data either for storage or sharing. They also want to specify access policies on the encrypted data in order to control who can access the data. Data users that are tagged with attributes want to decrypt the data they can access by requesting secret keys from the AA. The cloud will provide storage service as well as access control service. It will generate control section for data owners' uploaded data and help to partially decrypt the data when requested by data users. The system model is shown in Fig. 4.1.

Our proposed outsourced system can be modeled as the following functions.

1. $\mathsf{Setup}(\lambda) \rightarrow \{\mathsf{PK}, \mathsf{MSK}\}$: the AA generates public parameter PK and master secret key MSK using a security parameter λ. The public parameter PK is known to all the users as well as the cloud.

2. $\mathsf{KeyGen}(\mathsf{PK}, S) \rightarrow \mathsf{SK}$: when a data user with a set of attributes S requests his secret

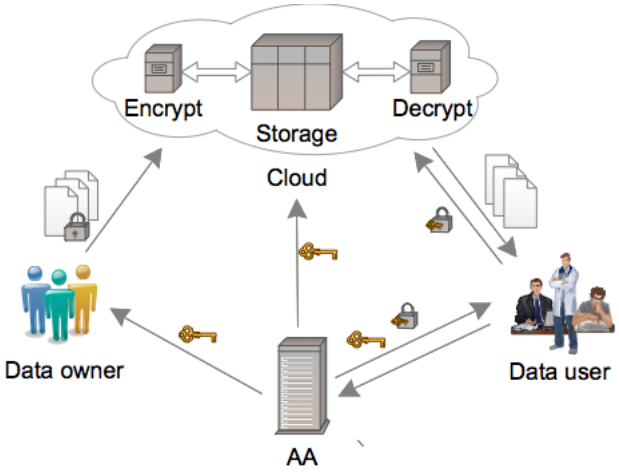

Figure 4.1: System Model of Outsourced ABE

key, the AA first authenticates that all the attributes in S are valid. Then AA generates and distributes the secret key SK associated with S to the data user through a secure channel. We note that the secret key SK is composed of SK_L and SK_S, where SK_L is kept secret by data user and SK_S is prepared for the cloud for partial decryption.

3. $\mathsf{Enc}_L(m, \mathsf{PK}) \rightarrow \{\mathsf{CT}_1, \mathsf{T}, \mathsf{aux}\}$: a data owner encrypts the message m to produce the ciphertext CT_1 and specifies an access policy T. Meanwhile, some auxiliary information is produced for further encryption. $\{\mathsf{CT}_1, \mathsf{T}, \mathsf{aux}\}$ is outsourced to the cloud.

4. $\mathsf{Enc}_S(\mathsf{PK}, \mathsf{T}, \mathsf{aux}) \rightarrow \mathsf{CT}_2$: the cloud enforce the access policy T to produce partial ciphertext CT_2 based on the auxiliary information aux. The ciphertext is formed as $\mathsf{CT} = \{\mathsf{CT}_1, \mathsf{CT}_2\}$ that is stored in the cloud.

5. $\mathsf{Dec}_S(\mathsf{CT}, \mathsf{SK}_S) \rightarrow \mathsf{CT}_S$: when a data user requests to access data CT, he sends the secret key SK_S to the cloud. The cloud decrypts the ciphertext CT to obtain partially decrypted ciphertext CT_S, which is then transmitted to the data user.

6. $\mathsf{Dec}_L(\mathsf{CT}_S, \mathsf{SK}_L) \rightarrow m$: the data user decrypts the ciphertext CT_S to obtain the message m using his secret key SK_L.

4.2.2 Threat Model

In the system, we assume that AA is fully trusted as is the case in typical ABE systems. We consider a *malicious* cloud model. On one hand, the cloud is curious in learning the message stored in cloud storage. On the other hand, the cloud can be compromised by malicious users. Invalid users without proper attribute sets may collude with the cloud in order to obtain the access right. The three functional modules (encryption, storage and decryption) are logically considered to be implemented in one single cloud environment. In this case, the cloud will have knowledge of the ciphertext CT as well as the secret key SK_S. We note that one highlight of our scheme is that we do not impose the strong assumption as in most related works that the cloud is only curious-but-honest. Later, we show that our scheme supports public auditing of all the operations conducted by the cloud. As a result, the cloud is enforced to honestly conduct the required operations.

4.3 A High Level View of CP-ABE

In this section, we introduce some necessary preliminaries of CP-ABE. We utilize the scheme in [55] as the underlying CP-ABE scheme for its highly expressive access structure. We emphasize on the realization of access control in ABE and model it as the process of encoding and decoding a secret.

4.3.1 Bilinear Pairing

A bilinear pairing is a mapping function $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are two multiplicative cyclic groups with prime order p. Let g be the generator of $\mathbb{G}_1$. The bilinear pairing e has the following properties:

1. The pairing e is bilinear. That is, $\forall g_1, g_2 \in \mathbb{G}_0, a_1, a_2 \in \mathbb{Z}_p, e(g_1^{a_1}, g_2^{a_2}) = e(g_1, g_2)^{a_1 a_2}$.

2. The pairing e is non-degenerate. That is $e(g, g) \neq 1$.

3. The paring e and the group operation in $\mathbb{G}_1$ are both efficiently computable.

4.3.2 Linear Secret Sharing Scheme

A secret sharing scheme is a method for a secret holder to distribute a secret to different parties and only under certain criteria can the parties reconstruct the secret. A secret sharing scheme is *linear* in that the mapping between a secret and the secret shares are realized by linear functions [64].

A typical construction of linear secret sharing scheme is Shamir's $(t-n)$-threshold secret sharing scheme [65]. In his construction, the secret and secret shares are all elements in a finite field $\mathbb{F}_q$. To share a secret s, the secret holder chooses a polynomial $f(x)$ of degree $(t-1)$ over $\mathbb{F}_q$. The coefficients $c_1, c_2, \cdots, c_{t-1}$ are randomly chosen from $\mathbb{F}_q$ and the constant term is set to be s. A secret share is a point $(a_i, f(a_i))$ on the polynomial, where $a_i \in \mathbb{F}_q$ is randomly generated. The basic idea is that for a polynomial of degree $(t-1)$, if t or more points on the polynomial are given, one can fully reconstruct the polynomial (coefficients) using polynomial interpolation. Suppose there are a total of n secret shares distributed to n parties. If there are t or more parties collude with each other, they can reconstruct the secret. Thus, this scheme is called the $(t-n)$-threshold secret sharing scheme.

4.3.3 Access Tree

In ABE schemes, a data owner needs to describe an access policy on encrypted data. An access tree is a kind of access structure that describing access policies. In the tree T, a

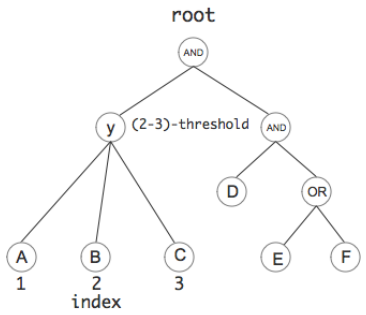

Figure 4.2: Access Tree

leaf node x stands for an attribute and we let A_x denote its attribute. A non-leaf node y represents a $(t_y - n_y)$-threshold gate, where $1 \leq t_y \leq n_y$ is its threshold value and n_y is the number of its children. Especially, when $t_y = 1$ node y represents an $\mathsf{OR}(\vee)$ gate and when $t_y = n_y$ it represents an $\mathsf{AND}(\wedge)$ gate. We denote the parent node of y as $p(y)$. The children of y are indexed from 1 to n_y and for a specific child x of y, a function $\rho(x)$ returns the index of x.

Let T_R denote a tree rooted at node R. For a set of attributes S and a leaf node x, we say set S satisfies T_x if and only if $A_x \in S$ and it is written as $\mathsf{T}_x(S) = 1$. Then $\mathsf{T}_R(S)$ can be calculated in a recursive way. We utilize Fig. 4.2 as an example to illustrate the access tree. In Fig. 4.2, there are 6 attributes $\{A, B, C, D, E, F\}$ in the access tree. The non-leaf node y represents a $(2-3)$-threshold gate. Thus the access tree describe a policy 2-out-of-$(A, B, C) \wedge D \wedge (E \vee F)$. An attribute set $S_1 = \{A, B, D, F\}$ will satisfy this policy while a set $S_2 = \{A, B, C, D\}$ will not.

4.3.4 Access Control: Encoding and Decoding a Secret

As stated previously, the ciphertext of ABE is divided into a payload section and a control section that are denoted as CT_e and CT_c , respectively. The data owner encrypts a message

m by some normal encryption algorithm to obtain

$$\mathsf{CT}_e = \mathsf{Enc}(m, s),$$

where s is the secret key. The control section can be modeled as the ciphertext of the access tree T encrypted with the secret key s, which is denoted as

$$\mathsf{CT}_c = \mathsf{Encode}(s, \mathsf{T}).$$

In decryption, given a secret key SK, a user can decode the secret key as

$$s = \mathsf{Decode}(\mathsf{CT}_c, \mathsf{SK}).$$

Then from s, the user can decrypt the message as

$$m = \mathsf{Dec}(\mathsf{CT}_e, s).$$

Now, we illustrate the encoding and decoding process in detail.

4.3.4.1 Encoding a Secret

The basic idea of encoding a secret according to an access tree T is to share the secret to each leaf node of the tree. Then, the secret share distributed to a leaf node is disguised by the attribute of the leaf node. Since each non-leaf node is a $(t-n)$-threshold gate, the secret is shared from the root of the node using $(t-n)$-threshold secret sharing schemes.

To be specific, given an access tree T, each node y (including the leaf node) is associated

with a randomly generated polynomial f_y of degree $d_y = t_y - 1$. These polynomials are generated in a recursive manner starting from the root node R. First, for root R, the constant term of f_R is set to be the secret key s, that is $f_R(0) = s$. Then another d_y points are chosen to fully define the polynomial f_R. For each child x, set $f_x(0) = f_{p(x)}(\rho(x))$ and another d_x points are randomly chosen to define polynomial f_x. In the end, for each leaf node l, we will obtain an associated secret share $f_l(0)$.

The next step is to disguise the secret share with the attribute for each leaf node $l \in L$, where L denotes the set of all leaf nodes. Especially, $\forall l \in L$, compute a pair of ciphertext $\mathsf{CT} = (C_{l1} = g^{f_l(0)}, C_{l2} = h(A_l)^{f_l(0)})$, where g is a public parameter and $h(\cdot)$ is a hash function that maps an arbitrary attribute to an element in $\mathbb{G}_1$. In summary, the secret encoding process can be modeled as a function

$$\begin{aligned} \mathsf{Encode}(s, \mathsf{T}) &= \mathsf{CT} \\ &= \{(C_{l1} = g^{f_l(0)}, C_{l2} = h(A_l)^{f_l(0)}), \forall l \in L\}. \end{aligned}$$

4.3.4.2 Decoding a Secret

The task of decoding is to reconstruct the secret key s encoded to the access tree T based on user's secret key SK. In ABE system, the secret key of each user is issued by the AA based on the user's attribute set. For a user with attribute set S, the secret key is

$$\mathsf{SK} = \{(D_{j1} = g^r \cdot h(j)^{r_j}, D_{j2} = g^{r_j}), \forall j \in S\},$$

where $r, r_j \in \mathbb{Z}_p$ are random numbers.

The decoding process is a recursive algorithm consisting of decoding each node in T.

Intuitively, the secret reconstruction starts from the leaf nodes and goes towards the root. Let

$$\mathsf{Decode}(\mathsf{CT}_c, \mathsf{SK}, S, x) \rightarrow c_x$$

be a function to decode a node x. When x is a leaf node, if $A_x \in S$, we can compute

$$\begin{aligned} c_x &= \frac{e(D_{A_x 1}, C_{x1})}{e(D_{A_x 2}, C_{x2})} \\ &= \frac{e(g^r \cdot h(A_x)^{r A_x}, g^{fx(0)})}{e(g^{r A_x}, h(A_x)^{fx(0)})} \\ &= \frac{e(g,g)^{r fx(0)} \cdot e(h(A_x), g)^{r A_x fx(0)}}{e(h(A_x), g)^{r A_x fx(0)}} \\ &= e(g,g)^{r fx(0)}. \end{aligned}$$

If $A_x \notin S$, set $c_x = \perp$.

When x is a non-leaf node, denote its children as a set S_{cx}. For each child $z \in S_{cx}$, the decoding result is c_z. If the number of children whose decoding result $c_z \neq \perp$ is smaller than the threshold value t_x, it means that the threshold gate associated with x is not satisfied. Then we set $c_x = \perp$. Otherwise, we can form a valid set S_{vx} of size t_x where each children node $z \in S_{vx}$ has a valid decoding result c_z. Let $I_{vx} = \{\rho(z) | z \in S_{vx}\}$ be the index set for the corresponding nodes in S_{vx}. For index $i \in I_{vx}$, denote the Lagrange coefficient be

$$L_i(u) = \prod_{j \in I_{vx}, j \neq i} \frac{u - i}{j - i}.$$

Then we can compute the decoding of node x as

$$\begin{aligned} c_x &= \prod_{z \in S_{vx}} c_z^{L_i(0)} \\ &= \prod_{z \in S_{vx}} e(g,g)^{r f_z(0) L_i(0)}. \end{aligned}$$

Since in encoding, the secret associated with node z is set as $f_z(0) = f_{p(z)}(\rho(z)) = f_x(i)$, c_x can be further written as

$$\begin{aligned} c_x &= \prod_{z \in S_{vx}} e(g,g)^{r f_x(i) L_i(0)} \\ &= e(g,g)^{r \sum_{i \in I_{vx}} f_x(i) L_i(0)}. \end{aligned}$$

From the Lagrange interpolation, we have

$$f_x(0) = \sum_{i \in I_{vs}} f_x(i) L_i(0).$$

Thus, we have

$$c_x = e(g,g)^{r f_x(0)}.$$

By repeating this decoding process, we can obtain

$$c_R = e(g,g)^{rs}.$$

We note that the secret decoding will not give the secret key s in clear form. Instead, the decoding result is a disguised form $e(g,g)^{rs}$ that is further used in message decryption. In

summary, the decoding process can be modeled as a function

$$\mathsf{Decode}(\mathsf{CT}_c, \mathsf{SK}, \mathsf{T}, S) = e(g,g)^{rs}.$$

4.4 Construction of Outsourced CP-ABE

In this section, we introduce the detailed construction of the outsourced CP-ABE system. We first present some intuitive ideas.

4.4.1 Exploring Linearity of Secret Sharing

In ABE system, the message m is encrypted as $\mathsf{CT}_e = \mathsf{Enc}(m, s)$ with a secret key s. The secret key s is then encoded as $\mathsf{CT}_c = \mathsf{Encode}(s, \mathsf{T})$. From the above analysis of encoding process, we can see that most of the computational burden lies in encoding. This motivates us to outsource the whole encoding portion to the cloud. However, if the cloud is in charge of encoding, it will inevitably learn the secret key s thus decryption the message. Moreover, the decoding process will produce the decoded secret as $e(g,g)^{rs}$. In an effective access control system, only the valid users can successfully decode the secret. Thus, it is crucial to prevent a compromised cloud and a malicious user to learn the decoding result $e(g,g)^{rs}$. These are the two critical issues in our design.

We address the above issues by exploring the linearity of secret sharing. Basically, the secret encoding process is a series of $(t-n)$-threshold secret sharing schemes according to the access tree T. In a secret sharing scheme, suppose the polynomial is

$$f(x) = s + c_1 x + c_2 x^2 + \cdots + c_d x^d.$$

Then a secret share is $(i, f(i))$. If we change all the secret shares to $(i, f(i)+\delta)$ with δ being a constant, then after the Lagrange interpolation, the recovered polynomial becomes

$$g(x) = (s+\delta) + c_1 x + c_2 x^2 + \cdots + c_d x^d.$$

That is, the shared secret changes from s to $s+\delta$. We define this linear relation between secret the secret shares as the linearity of secret sharing schemes.

The above analysis gives some insight of our proposed scheme. In ABE, the secret is encoded as

$$\mathsf{CT}_c = \{(C_{l1} = g^{f_l(0)}, C_{l2} = h(A_l)^{f_l(0)}), \forall l \in L\}.$$

Similarly, we can modify each pair of ciphertext (C_{l1}, C_{l2}) as $(C'_{l1} = g^{f_l(0)+\delta}, C'_{l2} = h(A_l)^{f_l(0)+\delta})$. Then the decoding process will reconstruct the secret as $e(g,g)^{r(s+\delta)}$. However, this modification requires data owners to calculate g^δ and $h(A_l)^\delta$ for each leaf node l in the access tree T. The computational overhead is still too expensive for mobile users. Instead, we propose to only modify the first part of the ciphertext pair. That is, set $(C'_{l1} = g^{f_l(0)+\delta}, C'_{l2} = C_{l2} = h(A_l)^{f_l(0)})$. We further re-design the decoding process such that the recovered secret would be $e(g,g)^{r(s+\delta)}$. The details will be provided in the construction of our outsourced ABE system.

In summary, the data owner will encryption the message as

$$\mathsf{CT}_e = \mathsf{Enc}(m, s+\delta).$$

The cloud will encode the secret key s (known to the cloud) according to an access tree T as

$$\mathsf{CT}_c = \mathsf{Encode}(s, \mathsf{T}).$$

On requesting, the cloud will decode CT_c to reconstruct the secret in the form of $e(g,g)^{r(s+\delta)}$. Based on the decoding result, the data user can decrypt the message m.

4.4.2 System Setup

The attribute authority executes $\mathsf{Setup}(\lambda) \rightarrow \{\mathsf{PK}, \mathsf{MSK}\}$.

1. AA generates a bilinear mapping $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, where $\mathbb{G}_1$ is a group of prime order p with generator g.

2. AA chooses two random exponents $\alpha, \beta \in \mathbb{Z}_p$ and calculates $h = g^{\beta}$ and $e(g,g)^{\alpha}$. The public key is published as

$$\mathsf{PK} = \{e, g, h = g^{\beta}, e(g,g)^{\alpha}\}.$$

3. AA sets the master key as

$$\mathsf{MSK} = \{\beta, \alpha\}.$$

4.4.3 Key Generation

When a user with attribute set S requests his secret key, the attribute authority executes $\mathsf{KeyGen}(\mathsf{PK}, S) \rightarrow \mathsf{SK}$ and delivers the secret key SK to that user.

1. AA chooses random integers $r, z \in \mathbb{Z}_p$ and computes

$$\begin{aligned} \mathsf{SK}_S &= \{D_1 = g^{(\alpha+r)/z}, \\ &\quad (D_{j1} = g^r \cdot h(j)^{rj}, D_{j2} = h(j)^{rj/\beta}, \\ &\quad D_{j3} = g^{rj}), \forall j \in S\}. \end{aligned}$$

2. AA computes $\mathsf{SK}_L = \{\frac{z}{\beta}\}$.

3. AA sets $\mathsf{SK} = \{\mathsf{SK}_L, \mathsf{SK}_S\}$. We note that SK_S is for the cloud to partially decrypt the ciphertext and SK_L is for the user to fully recover the plaintext.

4.4.4 Encryption

The encryption process consists of two phases: local side encryption $\mathsf{Enc}_L(m, \mathsf{PK}) \rightarrow \{\mathsf{CT}_1, \mathsf{T}, \mathsf{aux}\}$ and cloud side encryption $\mathsf{Enc}_S(\mathsf{PK}, \mathsf{T}, \mathsf{aux}) \rightarrow \mathsf{CT}_2$.

$\mathsf{Enc}_L(m, \mathsf{PK}) \rightarrow \{\mathsf{CT}_1, \mathsf{T}, \mathsf{aux}\}$

1. Data owner chooses random integers $s_1, s_2 \in \mathbb{Z}_p$ and computes

$$C_1 = m \cdot e(g, g)^{\alpha(s_1+s_2)}, C_2 = h^{s_1}, C_3 = g^{s_1}.$$

2. Data owner sets $\mathsf{aux} = \{s_2, C_2, C_3\}$ and $\mathsf{CT}_1 = C_1$.

3. Data owner specifies an access policy T and uploads $\{\mathsf{CT}_1, \mathsf{T}, \mathsf{aux}\}$ to the cloud.

$\mathsf{Enc}_S(\mathsf{PK}, \mathsf{T}, \mathsf{aux}) \rightarrow \mathsf{CT}_2$

1. Given the secret s_2 and access tree T, the cloud follows the encoding process introduced in Section 4.3.4.1. The secret s_2 is encoded as

$$\begin{aligned} \mathsf{Encode}(s_2, \mathsf{T}) &= \mathsf{CT}_c \\ &= \{(C_{l1} = g^{f_l(0)}, C_{l2} = h(A_l)^{f_l(0)}), \forall l \in L\}. \end{aligned}$$

2. The cloud computes $C_4 = C_2 \cdot h^{s_2} = h^{s_1+s_2}$ and $\forall l \in L$,

$$(C'_{l1} = C_{l1} \cdot C_3 = g^{f_l(0)+s_1}, C_{l2} = h(A_l)^{f_l(0)}).$$

3. The cloud sets

$$\mathsf{CT}_2 = \{C_4, (C'_{l1}, C_{l2}), \forall l \in L\}.$$

4. The whole ciphertext is $\mathsf{CT} = \{\mathsf{T}, \mathsf{CT}_1, \mathsf{CT}_2\}$, which is stored in the cloud storage.

4.4.5 Decryption

The decryption process consists of two phases: cloud side decryption $\mathsf{Dec}_S(\mathsf{T}, S, \mathsf{CT}, \mathsf{SK}_S) \rightarrow \mathsf{CT}_S$ and local side decryption $\mathsf{Dec}_L(\mathsf{CT}_S, \mathsf{SK}_L) \rightarrow m$.

$\mathsf{Dec}_S(\mathsf{T}, S, \mathsf{CT}, \mathsf{SK}_S) \rightarrow \mathsf{CT}_S$

1. When a data user wants to access encrypted data, he first sends part of his secret key SK_S to the cloud.

2. The cloud decodes the ciphertext as $\mathsf{Decode}(\mathsf{T}, \mathsf{CT}, \mathsf{SK}_S) \rightarrow m_1$. If the attribute set S associated with SK_S satisfies the access tree T, the decoding result is $m_1 = e(g,g)^{r(s_1+s_2)}$. Otherwise, $m_1 = \perp$ indicating that the data user does not have the right to access the encrypted data. The correctness of this decoding algorithm is given in Section 4.4.6.

3. The cloud computes

$$\begin{aligned} m_2 &= e(D_1, C_4) \\ &= e(g^{(\alpha+r)/z}, h^{s_1+s_2}) \\ &= e(g,g)^{(s_1+s_2)(\alpha+r)\beta/z}. \end{aligned}$$

4. The cloud sets $\mathsf{CT}_S = \{m_1, m_2\}$, which is then delivered to the data user.

$\mathsf{Dec}_L(\mathsf{CT}_S, \mathsf{SK}_L) \rightarrow m$

1. The data user decrypts the message as

$$\begin{aligned} \frac{C_1 \cdot m_2}{m_1^{\mathsf{SK}_L}} &= \frac{m \cdot e(g,g)^{\alpha(s_1+s_2)} e(g,g)^{r(s_1+s_2)}}{(e(g,g)^{(s_1+s_2)(a+r)\beta/z})^{z/\beta}} \\ &= \frac{m \cdot e(g,g)^{\alpha(s_1+s_2)} e(g,g)^{r(s_1+s_2)}}{e(g,g)^{(s_1+s_2)(a+r)}} \\ &= m. \end{aligned}$$

4.4.6 Proof of Correctness

In this section, we prove that given an attribute set S that satisfies the access tree T, the decoding process will decode the ciphertext as $m_1 = e(g,g)^{r(s_1+s_2)}$.

The decoding $\mathsf{Decode}(\cdot)$ is a recursive algorithm by decoding each node in T. Let $\mathsf{Decode}(\mathsf{T}, S, \mathsf{CT}, \mathsf{SK}_S, x) \to m_x$ be a function to decode a node x in T.

When x is a leaf node, the attribute associated with x is A_x. If $A_x \in S$, we can compute

$$\begin{aligned} m_x &= \frac{e(D_{A_x1}, C_{x1})}{e(D_{A_x3}, C_{x2}) \cdot e(D_{A_x2}, C_2)} \\ &= \frac{e(g^r \cdot h(A_x)^{rA_x}, g^{fx(0)+s_1})}{e(g^{rA_x}, h(A_x)^{fx(0)}) \cdot e(h(A_x)^{rA_x/\beta}, h^{s_1})} \\ &= \frac{e(g,g)^{r(fx(0)+s_1)} \cdot e(h(A_x), g)^{rA_x(fx(0)+s_1)}}{e(g, h(A_x))^{rA_x fx(0)} \cdot e(g, h(A_x))^{s_1 rA_x}} \\ &= e(g,g)^{r(fx(0)+s_1)}. \end{aligned}$$

If $A_x \notin S$, set $m_x = \perp$.

When x is a non-leaf node, denote its children as a set S_{cx}. For each child $z \in S_{cx}$, the decoding result is m_z. If the number of children whose decoding result $m_z \neq \perp$ is smaller than the threshold value t_x, it means the threshold gate associated with x is not satisfied. Then we set $m_x = \perp$. Otherwise, we can form a valid set S_{vx} of size t_x where each children node $z \in S_{vx}$ has a valid decoding result m_z. Let $I_{vx} = \{\rho(z) | z \in S_{vx}\}$ be the index set for the corresponding nodes in S_{vx}. For index $i \in I_{vx}$, denote the Lagrange coefficient as

$$L_i(z) = \prod_{j \in I_{vx}, j \neq i} \frac{u-i}{j-i}.$$

Then we can compute the decoding of node x as

$$\begin{aligned} m_x &= \prod_{z \in S_{vx}} m_z^{L_i(0)} \\ &= \prod_{z \in S_{vx}} (e(g,g)^{r(f_y(0)+s_1)})^{L_i(0)}. \end{aligned}$$

Since in encoding, the secret associated with node z is set as $f_z(0) = f_{p(z)}(\rho(z)) = f_x(i)$, m_x can be further written as

$$\begin{aligned} m_x &= \prod_{z \in S_{vx}} (e(g,g)^{r(f_x(i)+s_1)})^{L_i(0)} \\ &= e(g,g)^{r(\sum_{i \in I_{vx}} f_x(i) L_i(0) + s_1 \sum_{i \in I_{vx}} L_i(0))} \\ &= e(g,g)^{r(f_x(0)+s_1)}. \end{aligned}$$

The second equality comes from polynomial interpolation and the fact that the sum of Lagrange coefficients equals to 1. When the root of the tree is decoded, we can get

$$m_1 = m_R = e(g,g)^{r(f_R(0)+s_1)} = e(g,g)^{r(s_1+s_2)}.$$

Thus we have proved that

$$\mathsf{Decode}(\mathsf{T}, S, \mathsf{CT}, \mathsf{SK}_S) = m_1 = e(g,g)^{r(s_1+s_2)}$$

if and only if the attribute set S satisfies T. Otherwise, $m_1 = \perp$.

4.5 Complexity and Security Analysis

4.5.1 Complexity Analysis

The encryption and decryption process in ABE system contains mainly two computationally expensive operations: modular exponentiation (Exp) and bilinear pairing (Pair). Thus we utilize the number of Exp and Pair as measurements to evaluate the performance of our scheme.

In ABE system [55], the complexity of encryption and decryption is determined by the access policy. Suppose there are n attributes involved in the access tree T. For each attribute, the computation of the control section CT_c takes 2 Exp. To compute the encryption section CT_e in the ciphertext, the user has to conduct 2 Exp. Thus the complexity of encryption can be measured as $(2 + 2n)$ Exp. To decode a leaf node, the user needs to conduct 2 Pair. The complexity of decoding the tree T is also determined by the threshold gates in the non-leaf nodes. To decode a threshold gate with threshold value t, the user needs to first select t valid children to decode and calculate t Exp. Denote the sum of the threshold values of all the non-leaf nodes as K. The whole decryption process takes $(2 + 2K)$ Pair and K Exp.

In comparison, at local encryption $\mathsf{Enc}_L(m, \mathsf{PK}) \rightarrow \{\mathsf{CT}_1, \mathsf{T}, \mathsf{aux}\}$, the data user needs to conduct 3 Exp. For cloud side encryption $\mathsf{Enc}_S(\mathsf{PK}, \mathsf{T}, \mathsf{aux}) \rightarrow \mathsf{CT}_2$, it takes $(1 + 2n)$ Exp for the cloud. In decryption, $\mathsf{Dec}_S(\mathsf{T}, S, \mathsf{CT}, \mathsf{SK}_S) \rightarrow \mathsf{CT}_S$ takes the cloud $(1 + 3K)$ Pair and K Exp. The message decryption $\mathsf{Dec}_L(\mathsf{CT}_S, \mathsf{SK}_L) \rightarrow m$ takes the user 1 Exp. We note that in the above analysis, the do not count in the complexity to share the secret according the access tree in the original ABE scheme. In our scheme, the user only needs to specify the access policy and the cloud will conduct the secret sharing.

In terms of communication, in the original ABE, the data owner has to compute the

Table 4.1: Complexity Comparison

	Encryption		Decryption		Communication Overhead
	User	Cloud	User	Cloud	
Original ABE [55]	$(2+2n)$ Exp	–	$(2+2K)$ Pair + K Exp	–	$2\mathsf{Len}_2 + (2+4n)\mathsf{Len}_1 + 2\mathsf{Len}_\mathsf{T}$
Outsourced ABE	3 Exp	$(1+2n)$ Exp	1 Exp	$(1+3K)$ Pair + K Exp	$2\mathsf{Len}_1 + 3\mathsf{Len}_2 + \mathsf{Len}_\mathsf{T}$

encryption section CT_e and the control section CT_c first and upload them to the cloud together with an access policy T. We denote the size of an element in group $\mathbb{G}_1$ and $\mathbb{G}_2$ as Len_1 and Len_2, respectively. The access policy T is simply a string and denote its size as Len_T. Thus the size of the total ciphertext is $1\mathsf{Len}_2 + (1+2n)\mathsf{Len}_1 + \mathsf{Len}_\mathsf{T}$. When a data user wants to access the encrypted data, he has to download the ciphertext from the cloud storage. Thus, the total communication overhead for encryption and decryption is $2\mathsf{Len}_2 + (2+4n)\mathsf{Len}_1 + 2\mathsf{Len}_\mathsf{T}$.

In comparison, in our proposed outsourcing scheme, a data owner only needs to calculate and upload the encryption section CT_e and specify an access tree T. Thus the communication overhead for encryption is $2\mathsf{Len}_1 + \mathsf{Len}_2 + \mathsf{Len}_T$. A data owner needs to download the partially decrypted message CT_S with size $2\mathsf{Len}_2$. Thus the total communication overhead is $2\mathsf{Len}_1 + 3\mathsf{Len}_2 + \mathsf{Len}_T$.

The computational and communication complexity of the original ABE scheme and our outsourced ABE scheme is summarized in Table 4.1. From the table, we can see that in general, our scheme introduce $\mathcal{O}(n)$ performance gain for mobile users in both encryption and decryption. In total, the user and cloud together will do more computation. However, the increase of computation on cloud side which should be insignificant for the cloud.

4.5.2 Security Analysis

In this section, we analyze the security of our proposed scheme mainly from three aspects. That is (i) confidentiality of the encrypted file; (ii) honest access control and (iii) collusion tolerance. In our analysis, we let a single cloud server provide the partial encryption and decryption service, which is quite a strong adversary model. As a result, the cloud can hold the partial encryption and decryption key at the same time. We also assume that the cloud can be compromised. For example, malicious users can collude with the cloud in order to gain illegal access to certain files.

4.5.2.1 Confidentiality

Confidentiality is the most basic security requirement for ABE systems. It requires that the cloud cannot learn the encrypted message. In our scheme, the message m is encrypted as $C_1 = m \cdot e(g, g)^{\alpha(s_1+s_2)}$, where s_2 is known to the cloud while s_1 is kept secret by the user. Thus the only way for the cloud to recover message m is to obtain the secret s_1. For each encryption, the user will generate a different secret s_1 independently. As a result, the cloud can only try to learn s_1 from the partial encryption key g^s1. However, given g^s1, finding the discrete logarithm s_1 is computationally infeasible. Thus, we conclude that our scheme is secure in protecting the confidentiality of the message.

4.5.2.2 Honest Access Control

Besides confidentiality, the users are also concerned whether the cloud can honestly complete access control. One key feature of our scheme is that all the computations conducted by the cloud is not "security-sensitive". That is the input and output of the algorithms implemented at the cloud side can be publicly known and will not degrade the security. This feature

enables the possibility of public audition. A trusted third party can always repeat the computations conducted by the cloud. As a result, it decreases the chance for the cloud to successfully cheat in computation.

On the other hand, access control is marked as a service provided by the cloud. Since the access structure is embedded in ciphertext, a user thus can decide whether he is qualified to decrypt even before decryption. If a qualified user cannot decrypt the ciphertext, he can report an error thus degrading the quality of service.

4.5.2.3 Collusion Tolerance

Honest access control enables users to access files as long as their attributes satisfy the policy. A secure access control scheme should also prevent users whose attributes do not satisfy the policy from accessing the files. In some situations, malicious users may collude with the cloud in order to gain illegal access. In this case, a malicious user will give his user secret key $SK_L = \frac{z}{\beta}$ to the cloud. We note that successfully decoding of access tree will produce $e(g,g)^{r(s_1+s_2)}$. Similar to the previous analysis, it is computationally infeasible for the cloud to derive $(s_1 + s_2)$ since s_2 is kept secret by the data owners. As a consequence, the only way to derive $e(g,g)^{r(s_1+s_2)}$ is through decoding of access tree with valid set of attributes.

4.6 Numeric Results

In this section, we present some experiment results to demonstrate the efficiency of our proposed scheme. The experiment is conducted in an Android phone Samsung GT-I9100 with Android 4.1.2 operating system (we note that it is a relatively old android phone). The CPU is Dual-core 1.2 GHz Cortex-A9 with 1 GB RAM.

We implement our outsourced ABE based on Java Pairing-Based Cryptography Library (JPBC) [66]. We compare the average CPU time for both encryption and decryption of our proposed scheme with that of the benchmark implemented in [66]. In the implementation, each attribute policy is a string where attribute names are connected by "t-of-n" gates. For example, a policy "A B 1-of-2 C D 2-of-3" means at least 2 of $\{A \vee B, C, D\}$ should be satisfied. To measure the average performance of decryption, we utilize a simple access policy in the testing. We let all the leaf nodes be connected by a single root node and set the root as a $(\frac{n}{2} - n)$-threshold gate. We let the message m be a single element in $\mathbb{G}_1$. In practice, this m can be the symmetric encryption key for a group of files. Any data user who can recover the key m can access those files.

In our experiments, we utilize three different security levels as shown in Table 4.2. And we let the number of attributes vary from 1 to 30.

Table 4.2: Security Levels of ABE

Security level bits	$L_1(80)$	$L_2(112)$	$L_3(128)$
Bit length of r	160	224	256
Bit length of q	512	1024	1536

The experiment results are shown in Fig. 4.3 and Fig. 4.4. In the figures, "enc" and "dec" stands for encryption and decryption, respectively. "sos" indicates secure outsourcing. L_1, L_2, L_3 represents three different security levels. "gain" means the performance gain of outsourcing in terms of execution time. We can see that the encryption and decryption time for the original ABE are both approximately linear to the number of attributes. In comparison, the time for encryption and decryption at the local side almost remains constant

regardless of the number of attributes. For instance, for the most secure level L_3, encrypting a message on mobile phone takes around 2 seconds and decryption takes around 0.2 seconds. In comparison, at level L_3, encryption in the original ABE will take more than 100 seconds for 15 attributes and decryption will take around 28 seconds. Generally, encryption is more time consuming than decryption under the access policy specified in the experiment.

From the theoretical analysis in Table 4.1, we can see that at the local side, both the data owner and the data user need to do a constant number of Exp. While in the original ABE, the computational overhead of encryption and decryption are linear to the number of attributes n or the sum of the threshold K. Thus the experiment results are consistent with the theoretical analysis.

4.7 Application Scenarios

In this section, we introduce some application scenarios where our proposed scheme can be utilized.

From a theoretic point of view, we introduce a systematic way to securely outsource ABE. We modeled ABE as two highly detached functionalities: encryption and encoding. We further point out that encoding is a series of cascaded linear secret sharing schemes. Thus, we propose to utilize the linearity of secret sharing schemes to modify the secret shares in such a way that users can control the secret to be shared. Currently, most ABE systems utilize linear secret sharing schemes to construct the expressive access structure. For instance, the access tree structure as discussed in this paper and the monotone span programs [67] utilized in [56]. We believe that our proposed scheme can be applied in such kind of ABE systems utilizing linear system sharing schemes.

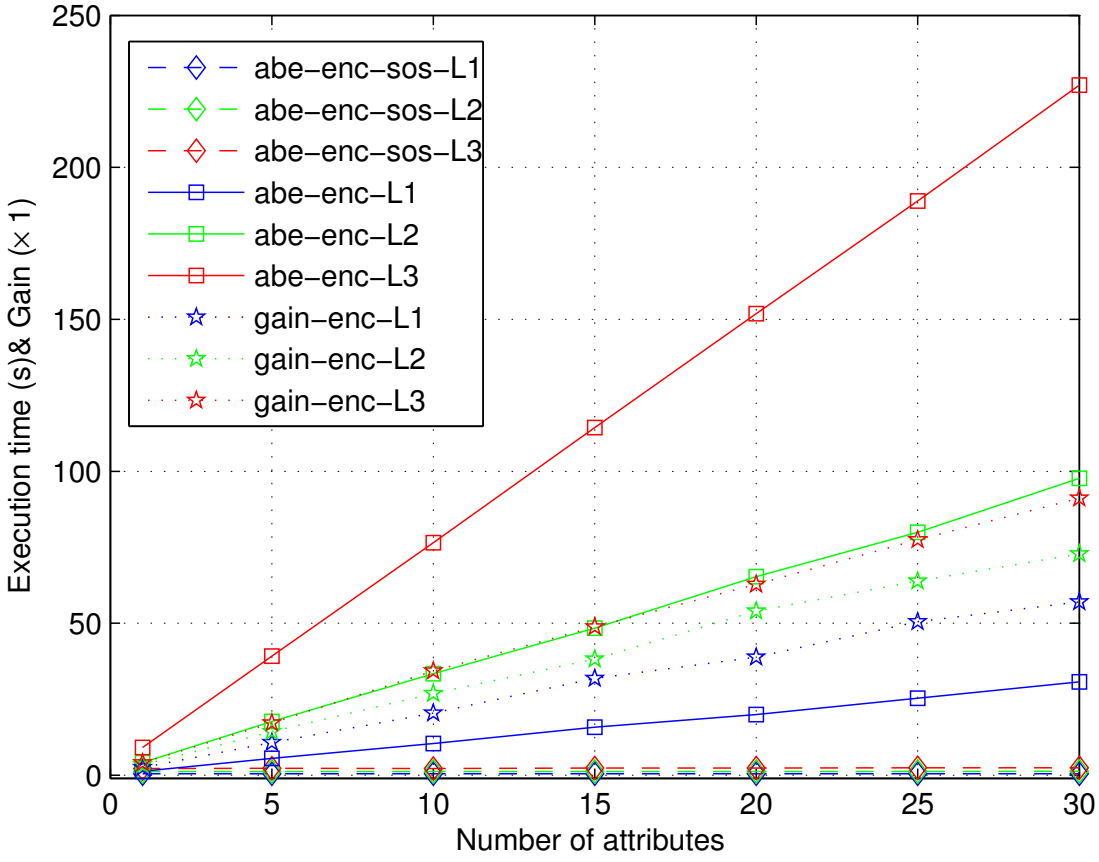

Figure 4.3: Performance Comparison of Outsourced Encryption

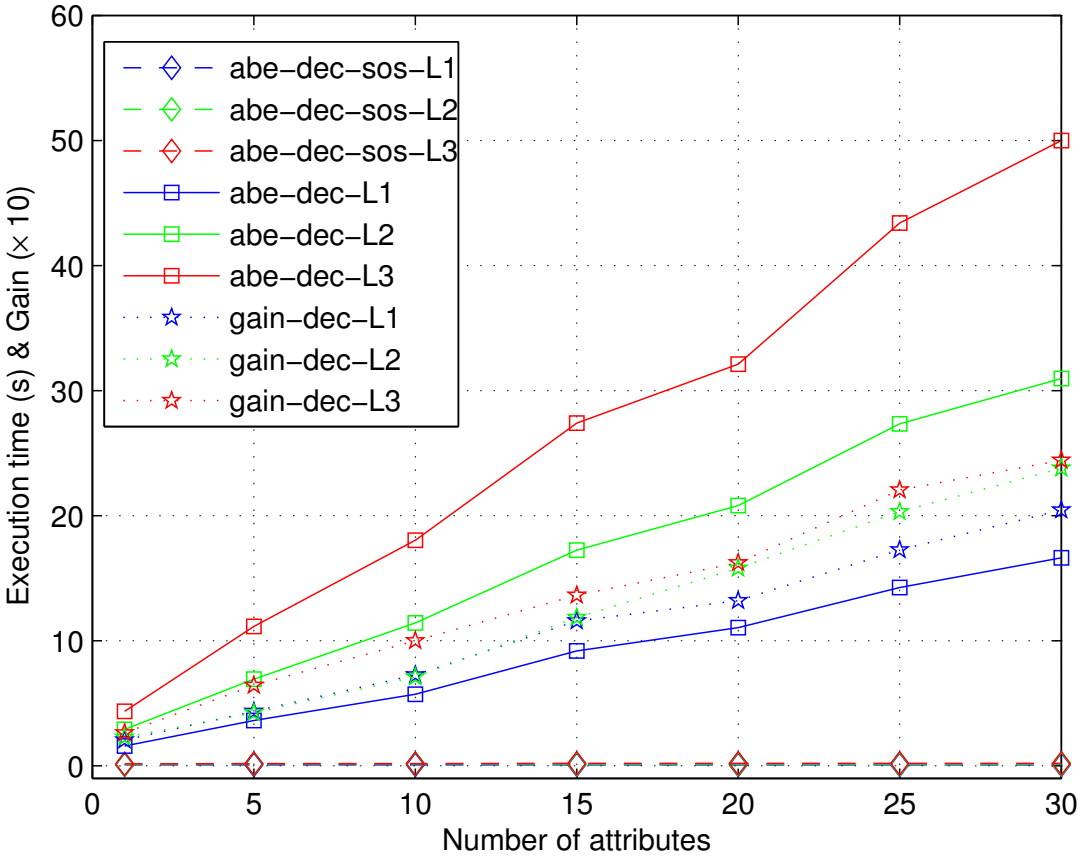

Figure 4.4: Performance Comparison of Outsourced Decryption

In practice, out scheme is suitable for the scenarios where mobile users need to store and share their private data in a fine-grained manner. More and more applications rely on mobile devices to collet sensitive data and upload the data to a sharing environment such as cloud. Then qualified users can utilized the data under the control of the data owner. For instance, in mobile healthcare system, wearable devices can sense and collect user's health related data that may be private. In this case, data owner can encrypt the data using efficient symmetric encryption schemes. A symmetric key may be related to a group of data files. The secret keys then can be encrypted using our proposed scheme. The whole encryption files are then uploaded to the cloud. The data owner can specify policies on the encrypted files controlling who can access the data. For example, he can specify that only his family or physicians in certain departments of certain hospitals can access the data. Compare to the traditional public key cryptosystems, the data owner can encrypt the data to a class of users he may not know in advance instead of a certain person he already knows. Moreover, data users do not have to request keys from the data owners. This enables the data owners to manage data off-line.

Chapter 5

PassBio: Privacy-Preserving User-Centric Biometric Authentication

5.1 Introduction

Biometric authentication has been incredibly useful in services such as access control to authenticate individuals based on their biometric traits. Unlike passwords or identity documents used in conventional authentication systems, biometric traits, such as fingerprint, iris and behavioral characteristics are physically linked to an individual that cannot be easily manipulated. Also due to such a strong connection, security and privacy of the biometric templates used in the authentication process is a critical issue [68–70].

Existing biometric authentication systems generally employ a two-phase mechanism [70]. In a registration phase, an end-user submits her biometric template to the service provider who will store the template along with the end-user's ID in a central database. In a query phase, the end-user requesting access to certain services will submit a fresh template to the service provider for authentication. Based on the end-user's ID, the service provider will retrieve the enrolled template for comparison. Only if the two templates are close enough

under certain distance metric, the end-user is successfully authenticated.

The above biometric authentication model can be regarded as *server-centric.* That is, the service provider will receive end-users' biometric templates in plaintext and is fully responsible for the security of the templates. Such an approach has several inherent deficiencies. First, the end-users have to fully trust the service provider to properly handle their templates; otherwise the security and privacy of the templates are at risk. For example, different service providers may crosscheck their databases to discover possible duplications, meaning that the same end-user may get enrolled in different services. As a consequence, the privacy of the end-user is violated. Second, unlike password, biometric templates are inherently noisy. As a result, the fresh template to be authenticated is not necessarily the same as the registered template. Such a property prevents the service provider from keeping the templates encrypted during the whole authentication process. At some point, the templates have to be recovered in plaintext for distance computation and comparison. This renders the adversaries with the opportunity to spy the registered or freshly submitted templates.

To address the above issues, we propose a *user-centric* model for biometric authentication. In terms of security, such a user-centric model has several unique features, compared to the server-centric model. First, biometric templates are encrypted at user side and then transmitted to the server. The service provider is only able to see encrypted versions of the registered templates and query templates. Second, the secret keys and the templates are generated and processed locally thus never leaving the local environment. Third, computations involved in authentication are all carried out on ciphertext, meaning that no templates are exposed in plaintext during the authentication. These features can effectively reduce the possibility for the server as well as outside adversaries to learn any key information of the biometric templates.

To meet the demands of the proposed user-centric model, the underlying encryption scheme should be efficient and expose as little information as possible. Since the key management and encryption are carried out at the user side, the encryption scheme should be computationally efficient. Some existing encryption schemes relying on heavy cryptographic operations such as Predicate Encryption (PE) [23, 24], Inner Product Encryption (IPE) [25–28] and Homomorphic Encryption (HE) [71, 72] may not be practical in such a scenario. Also, the encryption scheme should support certain kinds of computation on encrypted data. For example, given two encrypted vector, the server should be able to decide whether the two vectors are close enough (e.g., within a certain threshold) under some distance metric. The encryption scheme should expose as little template information as possible for security and privacy. Although some distance preserving transformation schemes [73] have been proposed for private nearest neighbor search on encrypted data, these schemes will inevitably expose the distance information between the registered and query template, which makes them vulnerable to security attacks [73].

In this paper, we propose a new primitive named Threshold Predicate Encryption (TPE). TPE encrypts two vectors $\mathbf{x}$ and $\mathbf{y}$ respectively as $C_{\mathbf{x}}$ and $C_{\mathbf{y}}$. Unlike traditional cryptosystems, the decryption of TPE will only reveal whether the inner product of $\mathbf{x}$ and $\mathbf{y}$ is within a threshold θ or not, instead of the plaintext. Therefore, no more information about the vectors and the inner product are exposed. TPE is fundamentally different from the previous schemes such as IPE [27] and PE [23]. IPE reveals the inner product of $\mathbf{x}$ and $\mathbf{y}$ thus the distance between the registered template and the query template, which makes the scheme vulnerable to security attacks [73]. PE can only reveal whether the inner product equals to a threshold or not. It is not flexible enough for biometric authentication since generally we want to know whether the distance between the two templates is within a threshold. In

comparison, our proposed TPE provides an excellent trade-off between information leakage and flexibility, which makes is uniquely suitable for biometric authentication.

TPE enables a compute-then-compare computational model over encrypted data. In this model, given ciphertexts, any party is able to **compute** the distance between the underlying plaintexts and then **compare** the distance with a threshold. The output is an indicator showing whether the distance is within the threshold or not. We show that such a computational model captures the essence of various applications such as privacy-preserving biometric identification and searching over encrypted data. TPE based schemes are able to fulfill the requirements of such applications while ensuring the security and privacy of the data.

The main contributions of this paper are summarized as follows:

- We propose a user-centric biometric authentication scheme enabling end-users to utilize their biometric templates for authentication while preserving template privacy.
- We propose a new primitive named TPE that can encrypt two vector $\mathbf{x}$ and $\mathbf{y}$ in such a manner that the decryption result only reveals whether the inner product of $\mathbf{x}$ and $\mathbf{y}$ is within a threshold or not.
- The proposed TPE enables a compute-then-compare computational model over encrypted data. We show that such a computational model can be applied to many privacy-preserving applications such as biometric identification and searching over encrypted data.

5.2 Related Work

The proposed TPE scheme can be regarded as an instance of functional encryption. That is, given the decryption key, the decryption process actually produces a function of the underlying plaintext, instead of the plaintext itself. From an application point of view, biometric authentication and identification is closely related to finding the nearest neighbor of a given point (i.e., nn or k-nn search). Thus, in this section, we review some related works concerning these two topics.

5.2.1 Functional Encryption and Controlled Disclosure

In conventional cryptosystem, the decryption process will eventually recover the underlying plaintext m. As a result, all information of m is disclosed. Many applications, however, require only *partially disclosure* of the information of m. For example, a financial organization wants to filter out those customers whose transactions exceed certain amount. For privacy concern, all the transactions of the customers are encrypted. In this case, instead of decrypting the transactions, a more desirable approach is to determine whether an transaction exceeds certain amount without disclosing the transaction. Such application scenarios motivate the research of functional encryption [20–22]. In a functional encryption scheme, a decryption key S_f is associated with a function f. Given the ciphertext C, the decryption process will evaluate the function $f(m)$, where m is the underlying plaintext. Note that in this process, the plaintext m cannot be recovered. Thus, by issuing different decryption keys S_{f_i}, functional encryption can actually implement *controlled disclosure* of the plaintext m.

Much research effort has been devoted to designing various functions f_i for functional encryption schemes. Representative works are Predicate Encryption (PE) [23, 24] and Inner

Product Encryption (IPE) [25–28]. In PE, a message is modeled as a vector $\mathbf{x}$ and a decryption key is associated with a vector $\mathbf{y}$. The decryption result is meaningful (otherwise, a random number) if and only if the inner product of $\mathbf{x}$ and $\mathbf{y}$ is equal to 0. Based on this basic implementation, different predicates are realized such as exact threshold, polynomial evaluation and set comparison. In contract, IPE schemes will recover the value of inner product of $\mathbf{x}$ and $\mathbf{y}$, without revealing neither $\mathbf{x}$ nor $\mathbf{y}$. In the context of controlled disclosure, IPE discloses more information of the plaintext than PE. This is because with PE, one can only decide whether the inner product of $\mathbf{x}$ and $\mathbf{y}$ is equal to a certain value or not while with IPE, one can know the value of the inner product. In comparison, with TPE, what we seek is to control the amount of information to be disclosed between those of PE and IPE. As a result, TPE can efficiently fulfill the task of biometric authentication while exposing less information about the templates.

5.2.2 Secure k-nn Search

The problem of secure k-nn search can be described as finding the k nearest neighbors (k-nn) of a given query point among a set of encrypted points. The schemes [73–77] for secure k-nn search mainly differ in the attack models they considered and the security levels they can provide. For instance, the scheme in [76] focused on search efficiency at the cost of partial privacy leakage. Both [73] and [77] considered a stronger known-plaintext attack model. The basic ideas of these two schemes are quite similar. Given two encrypted points in the data set and one encrypted query, the comparison process in the schemes is able to determine which point is closer to the query point. Repeating this comparison process will finally reveal which point in the data set the nearest neighbor to the query point.

Our proposed TPE scheme utilizes similar techniques as that in [77]. However, the

computational models as well as the security requirements are fundamentally different. In the biometric identification scheme in [77], given a query template, the server is able to identify the closest template in the database, which is returned to the end-user. After decryption of the returned template, the end-user is able to calculate the distance and determine whether the distance is within a threshold. We note that such a computational model cannot be easily applied to biometric authentication. This is because in biometric authentication, it is the server that compares the distance with a threshold while the server is not allowed to decrypt the templates thus calculating the distance. Moreover, secure k-nn based approaches will inherently expose more information than needed. From k-nn search, a sever can learn the relative distances between a query template and all the templates in the database. Such information is more than needed for biometric authentication and identification, where ideally, the server only needs to know whether the distance exceeds a pre-defined threshold.

5.3 Problem Statement

5.3.1 System model

We consider an online biometric authentication system consisting of two parties: an online service provider and a set of end-users. The service provider provides certain online services such as storage to its authenticated end-users. We assume that every end-user possesses a device such as a mobile phone that is able to collect the her biometric traits and transform the traits to biometric templates at the local side. Without loss of generality, we assume that each biometric template is represented by an n-dimensional vector $\mathbf{T} = (t_1, t_2, \ldots, t_n)$ of real numbers.

The biometric authentication process consists of two phases. In the registration phase, an end-user U_i will register with her biometric template $\mathbf{T}_i = (t_{i1}, t_{i2}, \ldots, t_{in})$ along with a unique identifier ID_i. We note that the template $\mathbf{T}_i$ is sent to the service provider in encrypted form denoted as $\mathsf{Enc}(\mathbf{T}_i)$ and ID_i can be any pseudorandom string that uniquely identifies U_i within the system. The tuple $\langle \mathsf{Enc}(\mathbf{T}_i), \mathsf{ID}_i \rangle$ for the end-user U_i is then stored at the server side by the service provider. In the query phase, when the end-user U_i desires to authenticate herself to the service provider, U_i will locally generate a fresh biometric template $\mathbf{T}'_i$ and send the tuple $\langle \mathsf{Enc}(\mathbf{T}'_i), \mathsf{ID}_i \rangle$ to the service provider, where $\mathsf{Enc}(\mathbf{T}'_i)$ is the encrypted form of $\mathbf{T}'_i$. On receiving the query, the service provider will retrieve the record $\langle \mathsf{Enc}(\mathbf{T}_i), \mathsf{ID}_i \rangle$ through searching ID_i in the server. Then distance between $\mathbf{T}_i$ and $\mathbf{T}'_i$ are computed based on $\mathsf{Enc}(\mathbf{T}_i)$ and $\mathsf{Enc}(\mathbf{T}'_i)$. If the distance is within a certain threshold θ, then the service provider will view the end-user U_i as a valid user. We also note that during the query phase, the service provider is only able to derive whether the distance between $\mathbf{T}_i$ and $\mathbf{T}'_i$ is within the threshold θ, instead of the exact distance between them.

5.3.2 Threat model

We assume the end-users are fully trusted in the registration phase. That is, they will honestly generate their own biometric templates and register at the service provider using the encrypted templates. In the query phase, we assume the encryption and decryption algorithms are publicly known. However, the secret keys are generated and kept secret at the local side throughout the whole authentication process. We do allow the adversaries to submit their own biometric templates through the local device. In this case, the local device acts as an oracle to encrypt templates and submit the encrypted templates to the service provider. The service provider can be honest-but-curious or malicious. In the former

case, the service provider will honestly follow the protocol but will try to obtain any useful information of end-users' biometric templates based only on the encrypted templates. In the latter case, the adversaries may collude with the service provider such as sharing with the service provider the invalid templates that are submitted through the local devices. In summary, depending on the different capabilities of the service provider and the adversaries, we propose two attack models as follows.

1. Passive Attack: the service provider is able to know the registered record $\langle \mathsf{Enc}(\mathbf{T}_i), \mathsf{ID}_i \rangle$ for end-user U_i and observe a series of m submitted queries $\mathsf{Enc}(\mathbf{T}_i^j)$, $j = 1, 2, \ldots, m$. However, the service provider does not know the underlying templates $\mathbf{T}_i^j$ in plaintext. Such an attack model is also known as the Ciphertext-Only-Attack in cryptography.

2. Active Attack: besides the registered record $\langle \mathsf{Enc}(\mathbf{T}_i), \mathsf{ID}_i \rangle$ for end-user U_i, the service provider is able to observe a series of m submitted queries $\mathsf{Enc}(\mathbf{T}_i^j)$ as well as the corresponding plaintext $\mathbf{T}_i^j$, $j = 1, 2, \ldots, m$. Such an attack model corresponds to the Chosen-Plaintext-Attack in cryptography. In practice, an adversary may submit her own templates through the local device. The service provider can then collude with the adversary to obtain the queries in plaintext as well as the encrypted queries.

Informally, the security requirement of biometric authentication is that the service provider is unable to learn any information about the templates than allowed through the authentication process. In particular, it should be possible for the service provider to determine whether the distance between two templates is within a threshold or not;but infeasible to derive any key information about the registered template as well as the query templates. We will formally define the security against both attacks in Section 5.6.

5.4 Proposed Threshold Predicate Encryption Scheme

A user-centric privacy-preserving biometric authentication scheme requires that an end-user is able to encrypt her registered biometric template as well as the freshly generated query templates. For the service provider, given two encrypted templates, it should be able to determine the distance between the two templates and compare the distance with a threshold. In this section, we introduce Threshold Predicate Encryption (TPE) that can fulfill the functionalities required by such a biometric authentication system.

5.4.1 Framework

Our proposed privacy-preserving biometric authentication scheme is based on the new primitive named Threshold Predicate Encryption (TPE). Generally speaking, TPE can be regarded as an instance of functional encryption [20, 21], where decryption will output a function of the plaintext instead of the plaintext itself. The framework of functional encryption can be briefly summarized as follows. A plaintext vector $\mathbf{x}$ is encrypted as $C_{\mathbf{x}}$ and a secret key associated with a vector $\mathbf{y}$ is generated as $S_{\mathbf{y}}$. Given $C_{\mathbf{x}}$ and $S_{\mathbf{y}}$, the decryption will give the value of $f(\mathbf{x}, \mathbf{y})$, where f is a pre-defined function. Two notable instances of functional encryption are Inner Product Encryption (IPE) [27] and Predicate Encryption (PE) [23]. The function f in IPE is the inner product. That is, the decryption of IPE will give the inner product of $\mathbf{x}$ and $\mathbf{y}$. In comparison, PE will produce a meaningful decryption result (e.g., a flag number 0) if and only if the inner product of $\mathbf{x}$ and $\mathbf{y}$ is 0. Otherwise, the decryption result is just some random number. An important predicate is that the inner product of $\mathbf{x}$ and $\mathbf{y}$ equals 0. Based on this, an extension of PE can implement exact threshold predicate encryption, meaning that the decryption result is meaningful only if the inner product of $\mathbf{x}$

and $\mathbf{y}$ is equal to a pre-defined threshold θ.

At the high-level view, functional encryption aims at revealing only limited information about the plaintext. As introduced above, IPE reveals the inner product of the plaintext and a vector. PE reveals whether the inner product is equal to 0 (or a threshold) or not. In application scenarios like biometric authentication, the amount of information revealed by IPE and PE are both inappropriate. As shown in our latter analysis, the inner product of $\mathbf{x}$ and $\mathbf{y}$ can be modeled as the distance between the registered template and the query template. As a result, IPE will give the exact distance between the two templates, which exposes too much information. With PE, one can decide whether the distance of the two templates is equal to a certain threshold, which is not sufficient for authentication purpose. What we need is an functional encryption scheme that can determine whether the distance between the two templates is within a threshold or not. Specifically, a TPE is composed of five algorithms:

- $\mathsf{TPE.Setup}() \rightarrow param$: the set up algorithm generates system parameters $param$.
- $\mathsf{TPE.KeyGen}(\lambda) \rightarrow sk$: on input of a security parameter λ, the key generation algorithm will generate a secret key sk.
- $\mathsf{TPE.Enc}(sk, \mathbf{x}) \rightarrow C_{\mathbf{x}}$: given a vector $\mathbf{x}$ and the secret key sk, the encryption algorithm will encrypt $\mathbf{x}$ as ciphertext $C_{\mathbf{x}}$.
- $\mathsf{TPE.TokenGen}(sk, \mathbf{y}) \rightarrow T_{\mathbf{y}}$: given a vector $\mathbf{y}$ and the secret key sk, the token generation algorithm will generate a token $T_{\mathbf{y}}$ for $\mathbf{y}$.
- $\mathsf{TPE.Dec}(C_{\mathbf{x}}, T_{\mathbf{y}}) \rightarrow \Lambda = \{0, 1\}$: given the ciphertext $C_{\mathbf{x}}$ and the token $T_{\mathbf{y}}$, the de-

cryption algorithm will output a result Λ satisfying

$$\Lambda = \begin{cases} 1, & \mathbf{x} \circ \mathbf{y} \leq \theta \\ 0, & \text{otherwise,} \end{cases}$$

where $\mathbf{x} \circ \mathbf{y}$ is the inner product of $\mathbf{x}$ and $\mathbf{y}$.

5.4.2 Design of TPE

While our proposed TPE scheme utilizes some similar techniques as the biometric identification scheme in [77], the settings of biometric authentication are fundamentally different. In particular, our proposed TPE is designed to address the following challenges.

Challenge 1 The system and threat model of outsourced biometric identification and biometric authentication are different. In biometric identification, the database owner possesses the encryption and decryption keys. The aim of the server is to identify the template closest to the query template. Then the database owner will retrieve the template, decrypt it and compare the distance to a threshold. However, in our scenario, the server does not possess the decryption key thus is unable to decrypt the encrypted template and calculate the distance. What we need is an encryption scheme that can directly determine whether the distance between the query template and the registered template is within the threshold based only on ciphertexts.

Challenge 2 The computation involved in biometric identification and authentication are different. In biometric identification, the sever needs to compute and compare the distances between a query template and all the templates in the database. However, in biometric

authentication, we need to compute the distance and compare it with a threshold.

Challenge 3 The decryption process in [77] will output a randomized distance between a query template and registered template. From this randomized distance, it is not easy to directly compare it with a threshold without first recovering the actual distance.

To address the above challenges, we first embed the threshold into the registered templates. To enhance security, we pad the templates with one-time randomness in a special manner and make random permutation to both the query template and registered template. After all these transformations, the decryption process can derive $\mathsf{dist}(\mathbf{x}, \mathbf{y}) - \theta$, where $\mathsf{dist}(\mathbf{x}, \mathbf{y})$ denotes the distance between a registered template $\mathbf{x}$ and a query template $\mathbf{y}$. However, if we output this value directly, it is inevitable that the exact value of $\mathsf{dist}(\mathbf{x}, \mathbf{y})$ will be exposed. Therefore, we introduce more one-time randomness into the encrypted templates. As a result, the decryption result becomes $\alpha\beta(\mathsf{dist}(\mathbf{x}, \mathbf{y}) - \theta)$, where α and β are positive one-time random numbers associated with $\mathbf{x}$ and $\mathbf{y}$, respectively. This design reveals only adequate information to determine whether the distance between is within the threshold and at the same time conceals the exact value of the distance.

5.4.3 Construction of TPE

Follow the aforementioned design of our threshold predicate encryption scheme, we give a detailed implementation in Protocol 8.

Now, we prove the correctness of the proposed TPE scheme. For a square matrix Y, the trace $\mathsf{Tr}(Y)$ is defined as the sum of the diagonal entries of Y. Given an invertible matrix M_1 of the same size, the transformation $M_1 Y M_1^{-1}$ is called similarity transformation of Y. We have the following lemma from linear algebra.

Lamma 5.1 *The trace of a square matrix remains unchanged under similarity transformation. That is,* $\mathsf{Tr}(Y) = \mathsf{Tr}(M_1 Y M_1^{-1})$.

Based on Lemma 5.1, we have the following theorem.

Theorem 5.1 *For the proposed TPE scheme in Protocol 1,* $\Lambda \leftarrow \mathsf{TPE.Dec}(C_x, T_y)$ *equals* 1 *if and only if* $\mathbf{x} \circ \mathbf{y} \leq \theta$*, where* $\mathbf{x} \circ \mathbf{y}$ *denotes the inner product of* $\mathbf{x}$ *and* $\mathbf{y}$.

Proof Following the procedure in Protocol 8, the vector $\mathbf{x}$ is transformed to $C_x = M_1 S_x X M_2$. The vector $\mathbf{y}$ is transformed to $T_y = M_2^{-1} Y S_y M_1^{-1}$. Then we have $C_x T_y = M_1 S_x X Y S_y M_1^{-1}$. From Lemma 5.1, we have $I = \mathsf{Tr}(C_x T_y) = \mathsf{Tr}(S_x X Y S_y)$. Since S_x and S_y are selected as lower triangular matrices, where all the diagonal entries are set to 1, the diagonal entries of $S_x X$ and $Y S_y$ are all the same as those of X and Y. Thus we have $I = \mathsf{Tr}(XY)$. Since X and Y are diagonal matrices, $I = \mathbf{x}'' \circ \mathbf{y}'' = \mathbf{x}' \circ \mathbf{y}' = \alpha\beta(\mathbf{x} \circ \mathbf{y} - \theta)$. Since α and β are positive, we have $\Lambda = 1$ (i.e., $I \leq 0$) if and only if $\mathbf{x} \circ \mathbf{y} \leq \theta$.

5.5 Biometric Authentication Under Different Distance Metrics

In this section, we will first introduce some necessary background on biometric authentication. Then, we show how to construct privacy-preserving biometric authentication systems utilizing our proposed TPE scheme under different distance metrics.

5.5.1 Backgrounds

The first critical step in biometric authentication is to efficiently transform biometric traits into templates that are easy for computation. Such a process is often called *feature extraction*.

The extracted features are often represented as *feature vectors*. Depending on the biometric traits, the process as well as the result of feature extraction could differ. For example, a fingerprint can be transformed to a FingerCode [78–80] that is a vector of integers with dimension 640. An Iris image is often represented as a binary string of 2048 bits. In the following, we briefly review the feature extraction process of fingerprints. The details can be found in [79, 80].

As illustrated in Fig. 5.1[1], given an image of a fingerprint, the first step is to identify a reference point. Then the region of interest around the reference point is divided into 5 bands and 16 sectors. Those sectors are further normalized and filtered by 8 different Gabor filters. At last, the features are extracted from each filtered image. The final result is a 640-dimensional vector (FingerCode) representing each fingerprint image, where each entry in the vector is an 8-bit integer. An import feature of the FingerCode is that it is translation invariant, meaning that translation of the fingerprint image would not result in much difference in the FingerCode. However, FingerCode is not rotation invariant. As a result, rotation of images will often cause different FingerCodes. To resolve this issue, a user is often associated with several (for example, 5) FingerCodes captured from rotated images in the database. In the following discussion, we assume that at the local side, there exists a sensor that can capture the end-user's biometric trait and transform it to a multi-dimensional vector.

In a user-centric biometric authentication system, an end-user will send her encrypted biometric template to the service provider in the registration phase. In the query phase, the end-user will encrypt a freshly generated template and send it to the service provider for authentication usage. Thus, a critical issue is to decide whether two templates are

[1]This figure is partially obtained from [80].

close enough. These problem is reduced to measuring the distance of two vectors in a metric space and compare the distance to a certain threshold. Such a compute-then-compare computational model on encrypted data is well suited for our proposed TPE scheme.

Furthermore, different biometric templates often rely on different similarity measurements. For example, in Iris recognition, the templates are represented by binary vectors and the similarity is generally measured by Hamming distance. For fingerprint, the Euclidean distance is normally utilized to measure the similarity. Our proposed TPE scheme is highly flexible in that it can be applied to measuring similarity based on different distance metrics. As a result, TPE can be utilized as the critical component to build different privacy-preserving biometric authentication systems. In the rest of this section, we will illustrate how to utilize TPE to construct a biometric authentication scheme based on Euclidean distance, Hamming distance and so on.

5.5.2 Euclidean Distance

Euclidean distance is often used to measure the similarity between vectors of non-binary entries. A FingerCode representing a fingerprint is an n-dimensional vector, where each entry is an l-bit integer. Typically, $n = 640$ and $l = 8$. We denote a registered FingerCode as $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and a query FingerCode as $\mathbf{y} = (y_1, y_2, \ldots, y_n)$. Let $\mathsf{d}_E(\mathbf{x}, \mathbf{y})$ be the Euclidean distance between $\mathbf{x}$ and $\mathbf{y}$. Then we have

$$\mathsf{d}_E^2(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} x_i^2 + \sum_{i=1}^{n} y_i^2 - 2\mathbf{x} \circ \mathbf{y},$$

where $\mathbf{x} \circ \mathbf{y}$ is the inner product of $\mathbf{x}$ and $\mathbf{y}$. Let θ be a pre-defined threshold. Our goal is to extend $\mathbf{x}$ and $\mathbf{y}$ to vectors $\mathbf{x}'$ and $\mathbf{y}'$ respectively such that the relation $\mathsf{d}_E^2(\mathbf{x}, \mathbf{y}) < \theta^2$ can be de-

termined through computing $\mathbf{x}' \circ \mathbf{y}'$. In light of this, we let $\mathbf{x}' = (2x_1, 2x_2, \ldots, 2x_n, -\sum_{i=1}^{n} x_i^2, 1, \theta^2)$ and $\mathbf{y}' = (y_1, y_2, \ldots, y_n, 1, -\sum_{i=1}^{n} y_i^2, 1)$. Then we have

$$\begin{aligned} \mathbf{x}' \circ \mathbf{y}' &= 2\mathbf{x} \circ \mathbf{y} + \theta^2 - \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} y_i^2 \\ &= \theta^2 - \mathsf{d}_E^2(\mathbf{x}, \mathbf{y}). \end{aligned}$$

To secure the biometric templates, we further add different randomnesses (i.e., α, β, r_x and r_y) to the extended vectors as shown in Protocol 9. The rest of the encryption procedures is then the same as those in TPE.Enc and TPE.TokenGen.

As presented in Protocol 9, during the registration phase, an end-user encrypts his template $\mathbf{x}$ as C_x and registers C_x along with her identity at the service provider. During the query phase, the end-user encrypts a freshly generated template $\mathbf{y}$ as T_y and sends T_y to the service provider. Then the service provider runs TPE.Dec with inputs C_x and T_y and outputs an authentication result. The correctness of this scheme is guaranteed by Theorem 5.1, with slight adaption to Euclidean distance. That is $\Lambda = \mathsf{Authenticated}$ if and only if $\mathsf{d}_E(\mathbf{x}, \mathbf{y}) \leq \theta$.

5.5.3 Distance in Hamming Space

From the construction of Euclidean distance, we know that the critical part in computing the distance through inner product lies in proper design of the extended vectors. Thus, in the following, we will focus on how to design the vectors in order to compute different distances.

Hamming distance is a popular metric to measure the similarity of binary template such as Iris. Now, we assume the registered template and query template are $\mathbf{x} = (x_1, x_2, \ldots, x_n)$

and $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ respectively, where x_i and y_i are 0 or 1. To calculate the Hamming distance $\mathsf{d}_H(\mathbf{x}, \mathbf{y})$ between $\mathbf{x}$ and $\mathbf{y}$, we first map the 0's in $\mathbf{x}$ and $\mathbf{y}$ to -1 and map 1's to 1. Then we have

$$2\mathsf{d}_H(\mathbf{x}, \mathbf{y}) = n - \mathbf{x} \circ \mathbf{y}.$$

The condition $\mathsf{d}_H(\mathbf{x}, \mathbf{y}) - \theta \leq 0$ is equivalent to $\mathbf{x} \circ \mathbf{y} + 2\theta - n \geq 0$. Thus, we need to design vectors $\mathbf{x}'$ and $\mathbf{y}'$ such that $\mathbf{x} \circ \mathbf{y} + 2\theta - n$ can be represented as $\mathbf{x}' \circ \mathbf{y}'$. In light of this, we let $\mathbf{x}' = (\beta x_1, \beta x_2, \ldots, \beta x_n, \beta(2\theta - n), r_x, 0)$ and $\mathbf{y}' = (\alpha y_1, \alpha y_2, \ldots, \alpha y_n, \alpha, 0, r_y)$. Then the rest of the authentication process is similarly as in Protocol 9.

In fact, the Hamming distance between two binary vectors is just one specific distance metric. There are many other different metrics such as Minkowski distance, Sokal & Michener similarity and Sokal & Sneath-II [81] introduced for different applications. Using our proposed TPE scheme, we are able to evaluate such metrics and compare them to a predefined threshold. The critical part is to properly design the vectors $\mathbf{x}'$ and $\mathbf{y}'$ given two binary vectors $\mathbf{x}$ and $\mathbf{y}$.

5.6 Security Analysis

In this section, we analyze the security of PassBio under both passive attack and active attack as defined in Section 5.3. PassBio is designed so that the service provider is unable to learn any critical information about the registered and query templates other than what is already revealed by the decryption process, given an encrypted registered template and a sequence of encrypted query templates.

Since PassBio is based on our proposed TPE, we will focus on the security analysis of TPE in the following discussion. An important difference between TPE and some traditional

symmetric encryption schemes is that it is the service provider (could be malicious) that carries out the decryption process. And the decryption process will reveal whether the inner product is within a threshold or not. Therefore, in the security analysis of TPE, it is necessary to analyze the security of both the encryption and decryption process, which will be discussed separately in the following sections.

5.6.1 Encryption Security

We first give a sketch of encryption security analysis. We will first utilize two experiments to model the ability of the adversary in passive attack and active attack, respectively. Then, we define the security of TPE under both passive and active attacks. At last, we prove the security of TPE under active attack since it implies the security under passive attack.

5.6.1.1 Security Against Passive Attack

In our scenario, the passive attack corresponds to the ciphertext-only-attack [82], where an adversary $\mathcal{A}$ observes a sequence of ciphertext. We define an experiment $\mathsf{Passive}^{\mathsf{mult}}_{\mathcal{A},\mathsf{TPE}}(\lambda)$ to simulate passive attacks, where the superscript mult denotes that the adversary $\mathcal{A}$ is able to submit multiply messages instead of one single message.

Based on $\mathsf{Passive}^{\mathsf{mult}}_{\mathcal{A},\mathsf{TPE}}(\lambda)$, we now define the security of TPE under passive attack.

Definition 5.1 *The proposed TPE scheme is secure against passive attack if for all polynomial-time adversary $\mathcal{A}$, there is a negligible function* negl *such that the probability*

$$|\mathsf{Pr}(\mathsf{Passive}^{\mathsf{mult}}_{\mathcal{A},\mathsf{TPE}}(\lambda) = 1) - \frac{1}{2}| \leq \mathsf{negl}(\lambda).$$

Remark In the above security definition, we only use the token generation function $\mathsf{TPE.TokenGen}$

as a representative. This is because the operations involved in TPE.Enc and TPE.TokenGen are almost the same. The security analysis for TPE.Token applies for TPE.Enc. However, in our security proof, we will show that both TPE.Enc and TPE.TokenGen meet the security requirement.

Based on Definition 1, we have the following theorem.

Theorem 5.2 *The proposed TPE scheme is secure against passive attack.*

We will omit the proof of Theorem 5.2. Instead, we will prove security against active attack since it implies the security under passive attack.

5.6.1.2 Security Against Active Attack

Under the active attack, the service provider is able to observe a sequence of pairs of query templates as well as their encrypted version. This can happen when, for example, some adversaries submit their templates and collude with the service provider. This attack scenario corresponds to the Chosen-Plaintext-Attack (CPA) in cryptography. Accordingly, an encryption scheme has CPA-security if it is secure against CPA. To prove that TPE has CPA-security, we model the active attack using and experiment $\mathsf{Active}_{\mathcal{A},\mathsf{TPE}}(\lambda)$. We define CPA-security for TPE as follows.

Definition 5.2 *The proposed TPE is secure against active attack if for all polynomial-time adversary $\mathcal{A}$, there is a negligible function* negl *such that the probability*

$$|\mathsf{Pr}(\mathsf{Active}_{\mathcal{A},\mathsf{TPE}}(\lambda) = 1) - \frac{1}{2}| \leq \mathsf{negl}(\lambda).$$

Remark Different from the passive attack experiment, the adversary will continually have

oracle access to the token generation function. This models the situation where the adversary is able to observe multiple pairs of messages and their ciphertexts.

Remark Unlike the passive attack experiment where the adversary submits multiple pairs of messages, we only discuss the situation where the adversary submits one pair of messages (m_0, m_1) to the challenger. This is because it is proved in [82] that any private-key encryption scheme that is CPA-secure is also CPA-secure for multiple encryptions. As a result, it is sufficient to prove that TPE is CPA-secure for one single encryption.

Theorem 5.3 *The proposed TPE is secure against active attack.*

Proof We need to prove that the adversary $\mathcal{A}$ cannot distinguish $\mathsf{TPE.TokenGen}(sk, m_0)$ and $\mathsf{TPE.TokenGen}(sk, m_1)$, even given the oracle access to $\mathsf{TPE.TokenGen}(sk, \cdot)$.

Consider the encryption of message m_0. Suppose $m_0 = (m_{0,1}, m_{0,2}, \ldots, m_{0,n})$ is an n-dimensional vector. Follow the procedure in $\mathsf{TPE.TokenGen}$, the vector m_0 is first extended to a vector $m_0' = (\alpha m_{0,1}, \alpha m_{0,2}, \ldots, \alpha m_{0,n}, \alpha, 0, r_0)$, where α and r_0 are random numbers. The vector m_0' is then permuted as m_0'', which is then extended to an $(n+3) \times (n+3)$ diagonal matrix Y_0. Then, the ciphertext for m_0 is $c_0 = M_2^{-1} Y_0 S_0 M_1^{-1}$, where S_0 is a random lower triangular matrix. We note that the product of Y_0 and S_0 will produce a lower triangular matrix denoted as G_0, with m_0' as the diagonal. Now we focus on the product $c_0 = M_2^{-1} G_0 M_1^{-1}$.

Denote the entries in M_2^{-1} and M_1^{-1} as a_{ij} and b_{ij}, respectively, where $i, j = 1, 2, \ldots, n+3$. For matrix G_0, denote its non-zero entries in the lower triangular part as s_{ij}, where $i > j$ and $i, j = 1, 2, \ldots, n+3$. Then, by law of matrix multiplication, each entry c_{ij} in c_0 can be

written in the form of

$$\begin{aligned} c_{ij} &= \sum [f_{ij}^1(a_{ij}, b_{ij})m_i + f_{ij}^2(a_{ij}, b_{ij})\alpha \\ &\quad + f_{ij}^3(a_{ij}, b_{ij})r_0 + f_{ij}^4(a_{ij}, b_{ij}, s_{ij})], \end{aligned} \tag{5.1}$$

where f_{ij}^t, $t = 1, 2, 3, 4$ are polynomials. Equation (5.1) is obtained by summing up each terms of m_i, α and r_0, respectively.

Now, observe Equation (5.1) in the context of the experiment $\mathsf{Active}_{\mathcal{A},\mathsf{TPE}}(\lambda)$. We know that a_{ij} and b_{ij} are fixed. a, r and s_{ij} are one-time random numbers. m_i are chosen and can be controlled by the adversary $\mathcal{A}$. In step 4) of experiment $\mathsf{Active}_{\mathcal{A},\mathsf{TPE}}(\lambda)$, the adversary $\mathcal{A}$ can select different m_i each time and observe the value of c_{ij} since $\mathcal{A}$ continuously has oracle access to $\mathsf{TPE.TokenGen}(sk_i, \cdot)$. However, since a, r and s_{ij} are one-time random numbers, the polynomials $f_{ij}^2(a_{ij}, b_{ij})\alpha$, $f_{ij}^3(a_{ij}, b_{ij})r$ and $f_{ij}^4(a_{ij}, b_{ij}, s_{ij})$ all looks random to $\mathcal{A}$. As a result, the summation c_{ij} looks random to $\mathcal{A}$. This means that, for any message m chosen by $\mathcal{A}$ and its corresponding ciphertext, $\mathcal{A}$ cannot distinguish which message is actually encrypted. Thus, the adversary $\mathcal{A}$ can only output b' by randomly guessing. Thus we have

$$|\mathsf{Pr}(\mathsf{Active}_{\mathcal{A},\mathsf{TPE}}(\lambda) = 1) - \frac{1}{2}| \leq \mathsf{negl}(\lambda).$$

5.6.2 Decryption Security

The decryption function $\mathsf{TPE.Dec}$ outputs an intermediate result denoted as $R = C_x T_y$ and a final result $I = \mathsf{Tr}(R)$. In the following security analysis, we discuss what information can be learned by the service provider from R and I.

As in Protocol 8, $R = M_1 S_x XY S_y M_1^{-1}$, where S_x and S_y are random matrices. Recall the proof for Theorem 5.3, where $c_0 = M_2^{-1} G_0 M_1^{-1}$. Since matrix G_0 and XY follow the same construction, it is obvious that the transformation $R = M_1 S_x XY S_y M_1^{-1}$ also has CPA-security. In other words, the transformation is semantically secure, meaning that the adversary is not able to derive any key information of X and Y from R.

Now, for the final result $I = \alpha\beta(\mathbf{x} \circ \mathbf{y} - \theta)$, we define a decryption oracle $\mathcal{O}$ as follows.

Theorem 5.4 *The oracle $\mathcal{O}$ does not have CPA-security.*

Proof We provide a proof sketch since the CPA-security proof process follows that for Theorem 5.3.

An adversary $\mathcal{A}$ is able to continuously have access to $\mathcal{O}$. $\mathcal{A}$ will submit $\mathbf{y}_i$ at her own choice and observe the output γ_i. Since α and β are positive, it is possible that there exists $\mathbf{y}_1$ and $\mathbf{y}_2$ such that $\gamma_1 > 0$ while $\gamma_2 < 0$. This means that, in an experiment defined for CPA-security, the adversary $\mathcal{A}$ is able to distinguish two ciphertext for two submitted messages. By definition, the oracle $\mathcal{O}$ does not have CPA-security.

Theorem 5.4 states that the final result I actually reveals some information about $\mathbf{x}$ and $\mathbf{y}$. This result is expected in our design since we want to determine if the inner product of $\mathbf{x}$ and $\mathbf{y}$ is within a threshold θ or not from the final result I. However, we note that in our proposed TPE, every vector $\mathbf{y}$ is associated with a one-time independent random number α and every vector $\mathbf{x}$ is associated with a one-time random number β. As a result, in the active attack, what an adversary can observe through decryption is a series of results $I_i = \alpha_i\beta(\mathbf{x} \circ \mathbf{y}_i - \theta)$. Since α_i are selected independently, the final results I_i only reveals whether $\alpha_i\beta(\mathbf{x} \circ \mathbf{y} - \theta)$ is positive or not. No more key information can be derive from I_i.

5.6.3 The Effect of Randomness on Security

Besides the randomly generated long-time keys (i.e., M_1, M_2 and π), we also introduce different one-time randomness in the encryption scheme. At the high-level view, the one-time randomness provides TPE with CPA-security similar to that of the one-time pad. From a cryptographic point of view, the one-time pad encryption scheme provides perfect security. However, it is not practical since the one-time secret key has the same length as the message itself. The most notable difference between TPE and the traditional encryption schemes is that TPE actually does not decrypt the message. Instead, TPE evaluate a function of the ciphertext in order to obtain the function value of the plaintext. As a result, TPE does not require the one-time randomness in the decryption process. In this sense, TPE can achieve the security comparable to the one-time pad while avoiding the impractical key management requirement.

It is important to understand the effect of different randomness on security. We briefly categorize the one-time randomness utilized by TPE into three types.

- Type I: *result-disguising randomness*. When extending the vectors in both TPE.Enc and TPE.TokenGen, we use random β and α respectively to multiply with each entry of $\mathbf{x}$ and $\mathbf{y}$. Since α and β will remain in the decryption result, we name it as result-disguising randomness.

- Type II: *vector-extension randomness*. In both TPE.Enc and TPE.TokenGen, we extend the vector and pad it with a random r.

- Type III: *matrix-multiplication randomness*. In both TPE.Enc and TPE.TokenGen, we multiply the extended matrices (X and Y) with random matrices (S_x and S_y).

These one-time randomnesses together ensure the CPA-security of the encryption process of TPE as analyzed in Section 5.6.1. The main function of decryption is to evaluate the trace of the matrix. We note that the trace function will cancel Type II and Type III randomness. However, Type I randomness will remain in the decryption result. This is important since it will only reveal partial information of the plaintext, which is just adequate for the purpose of biometric authentication. We will further demonstrate the effect of Type I randomness in Section 5.7.1.

5.7 Other Applications of TPE

Our proposed threshold predicate encryption scheme enables a compute-then-compare computational model over encryption data. That is, given two encrypted vector $\mathbf{x}$ and $\mathbf{y}$, an untrusted party is able to determine whether the inner product of $\mathbf{x}$ and $\mathbf{y}$ is greater than or within a threshold θ. No other key information about the value of $\mathbf{x}, \mathbf{y}$ or $\mathbf{x} \circ \mathbf{y}$ is exposed. Previously, we also showed that utilizing the inner product of $\mathbf{x}$ and $\mathbf{y}$, we are able to compute many distance and similarity metrics. Such properties of TPE are critical for many applications that require data security and privacy.

5.7.1 Improved Security for Outsourced Biometric Identification

Outsourcing of different computational problems to the cloud while preserving the security and privacy of the outsourced problem has becoming a new trend. Many previous works have considered secure outsourcing of different problems [8,77,83–86]. In [77], a secure outsourcing scheme is proposed for biometric identification. The system models of outsourced biometric identification and biometric authentication are fundamentally different. In outsourced

biometric identification, a data owner possesses a database of users' biometric templates. The goal of biometric identification is that given a query template, the data owner needs to identify a user to whom the query template belongs to.

To protect the security and privacy of biometric templates, [77] proposed an outsourcing scheme where the database owner will first encrypt the templates and then outsource the encrypted data to the cloud. Specifically, the data owner encrypts a biometric template $\mathbf{x}$ as C_x using a symmetric key sk. For a given query template $\mathbf{z}$, it is also encrypted as C_z using the same key sk. The scheme is designed in such a manner that given two encrypted templates C_x and C_y and a query template C_z, the cloud is able to determine which template ($\mathbf{x}$ or $\mathbf{y}$) is closer to $\mathbf{z}$, without learning any key information about $\mathbf{x}$, $\mathbf{z}$ and $\mathbf{y}$. By repeating this process, the cloud is able to identify the template $\mathbf{x}$ that is closest to $\mathbf{z}$. Then the encrypted version C_x is returned to the data owner, who can decrypt C_x to obtain $\mathbf{x}$ and calculate the actual distance between $\mathbf{x}$ and $\mathbf{z}$. Thus, the data owner can finally decide whether $\mathbf{x}$ and $\mathbf{y}$ are close enough such that they belong to the same person.

There are mainly two security and privacy issues regarding the above scheme. First, the registration phase is vulnerable to the registration attack [87], since an adversary (i.e., the cloud) is able to inject known templates into the database. During decryption, the cloud is able to derive the following equation (i.e., Equation (3) in [87]):

$$b_{ci} = \frac{(\mathsf{Tr}(Y_i^{'} B_c^{'}) - \mathsf{Tr}(X_i^{'} B_c^{'})) - (y_{i(n+1)} - x_{i(n+1)})}{y_{ii} - x_{ii}},$$

where b_{ci} is the i-th entry in a submitted query template $\mathbf{b}_c$. Since $\mathsf{Tr}(Y_i^{'} B_c^{'})$ and $\mathsf{Tr}(X_i^{'} B_c^{'})$ are computable and $\mathbf{x}$ and $\mathbf{y}$ are selected by the cloud, the cloud is able to recover b_{ci}. Repeating such attack will finally recover the whole query template $\mathbf{b}_c$ as demonstrated

in [87].

Second, from the decryption result, the cloud is able to learn more information than needed. In particular, the cloud is able to determine which one of any two encrypted template is closer to the query template. By repeating this process, the cloud can actually rank all the templates by their distances to the query template. This unnecessarily reveals more information than what is needed in biometric identification.

We now show that our proposed TPE scheme can address these two issues. The security vulnerability of the scheme in [77] was caused due to lacking of Type I randomness as defined in Section 5.6.3. The trace function $\mathsf{Tr}(\cdot)$ will cancel the Type III randomness, resulting in Equation (3) in [87].

Our proposed TPE scheme can be directly utilized in outsourced biometric identification. In the encryption part, each registered template $\mathbf{x}$ is encrypted with $\mathsf{TPE.Enc}$. A query template $\mathbf{z}$ is encrypted with $\mathsf{TPE.TokenGen}$. The decryption process will give $\alpha_z\beta_x(\mathsf{dist}^2(\mathbf{x},\mathbf{z})-\theta^2)$, where α_z and β_x are one-time random numbers associated with $\mathbf{z}$ and $\mathbf{x}$ respectively. As a result, Equation (3) in [87] is replaced by

$$b_{ci} = \frac{(\mathsf{Tr}(PB_c^{'}) - \mathsf{Tr}(QB_c^{'})) - (p_{n+1} - q_{n+1})}{\alpha_c\beta_x(p_n - q_n)}.$$

Note that α_c is a one-time random number associated with a query b_c and β_x is a one-time random number associated with $\mathbf{x}$. Thus, although the adversary is able to insert known templates into the database, it cannot derive b_{ci} due to the one-time randomness. In other words, the outsourced biometric identification scheme based on TPE is able to defend against registration attack.

For the second privacy issue, the decryption result $\alpha_z\beta_x(\mathsf{dist}^2(\mathbf{x},\mathbf{z})-\theta)$ will only reveal

whether the distance between the query $\mathbf{z}$ and the registered template $\mathbf{x}$ is within a threshold or not. Since β_x is a one-time randomness associated with each registered template $\mathbf{x}$, the relative distance information is concealed. As a result, the cloud is not able to rank all the registered templates according to the distance to the query template.

5.7.2 Searching Over Encrypted Data

With the development of cloud computing and storage, there is a clear motivation for searching over encrypted data [88–91]. For example, a medical institution may store its medical data in the cloud. To ensure security of the data, the institution chooses to encrypt all the data before outsourcing. Meanwhile, the institution wishes to maintain the searching ability over the encrypted data in order to retrieve the desired data files. The proposed TPE is a promising solution for searching over encrypted data. In the following, we discuss how to utilize TPE to implement different searching functionalities.

5.7.2.1 Set Intersection

We assume that a file F_i is indexed by a set of keywords S_i. The files and their associated keyword sets are encrypted and outsourced to the cloud. A search query consists of a set of keywords S_j. Given the search query, the cloud will return the file F_i if the overlap of keyword sets S_i and S_j exceeds a certain threshold θ. That is $|S_i \cap S_j| > \theta$.

The above set intersection search function can be implemented through TPE as follows. Suppose the universe of keywords is the set S with size n. Fix the order of the keywords within S. Then, an index S_i can be formulated as an n-dimensional binary vector $\mathbf{x}_i$, where $x_i^t = 1$ means that the t-th keyword in S appears in S_i. The vector $\mathbf{x}_i$ for file F_i is encrypted using $\mathsf{TPE.Enc}$. Each file is then encrypted using standard symmetric encryption schemes

such as AES. The encrypted files and index are outsourced to the cloud. For a search query S_j, a vector $\mathbf{x}_j$ can be formulated in a similar manner. Then a search token can be generated using TPE.TokenGen. With this formulation, it is obvious that $|S_i \cap S_j| = \mathbf{x}_i \circ \mathbf{x}_j$, where $\mathbf{x}_i \circ \mathbf{x}_j$ denotes the inner product of $\mathbf{x}_i$ and $\mathbf{x}_j$. With TPE, the cloud is able to identify the files whose associated indices satisfy $\mathbf{x}_i \circ \mathbf{x}_j > \theta$ while not learning any useful information about the indices.

5.7.2.2 Weighted Sum Evaluation

For many numeric data, it is significant to evaluate the weighted sum of the data record with different weights. For example, the grades of each subject for a student form a vector G_i. An evaluator wants to evaluate the performance of the students via some criteria. Each criterion can be formulated as the weighed sum of the grades. The different weights reflects different emphasis on the subjects.

We assume that an administrator possess the grades for all the students. For privacy issues, all the grades are encrypted using TPE.Enc and stored in an external server. An evaluator desires to identify those students whose performance meets certain standard. In this scenario, the evaluator can submit a vector of weights W_j to the administrator, who will then generate a search token for the evaluator through TPE.TokenGen. The evaluator can submit the token generated for W_j to the sever and search over the encrypted grades. The server is then able to identify the students whose grades satisfy $G_i \circ W_j > \theta$.

5.8 Performance Evaluation

In this section, we evaluate the performance of PassBio. First, we give detailed analysis of both computational and communication complexity. Then, some numeric results are presented for the proposed TPE through simulation.

5.8.1 Complexity Analysis

As shown in Protocol 9, at local side an end-user needs to run the TPE.KeyGen, TPE.Enc and TPE.TokenGen algorithms. The service provider needs to run the TPE.Dec algorithm for every query. It is obvious that the computational bottleneck of these algorithms lies in matrix multiplication or matrix inversion. Thus, in the following analysis, we will focus on matrix multiplication and inversion. Without loss of generality, we assume that the matrices involved in the computation all have the same dimension $n \times n$.

For the function TPE.KeyGen, two random matrices are generated and two matrix inversions need to be calculated. Note that the setup phase is generally a one-time process. That is, TPE.KeyGen needs to be executed by the end-user only once. The function TPE.Enc and TPE.TokenGen will both take 3 matrix multiplications. As a result, they have a complexity of $\mathcal{O}(n^3)$, without optimization for matrix multiplication.

In the function TPE.Dec, the trace of $C_x T_y$ needs to be computed. There is no need to calculate the matrix multiplication before evaluating the trace. Only computing of the diagonal entries is needed. Thus, TPE.Dec has a complexity of $\mathcal{O}(n^2)$.

In terms of communication overhead, assume all the matrix or vector has the same size l. In the registration phase, the end-user needs to submit the encrypted template C_x to the service provider. Thus the communication overhead for registration is $n^2 l$. Similarly, the

communication overhead for the query phase is also $n^2 l$.

5.8.2 Efficiency Improvement

The above complexity analysis shows that the computational bottleneck of both TPE.Enc and TPE.TokenGen lie in matrix multiplication. For resource-constrained devices such as mobile phones, the computation of matrix multiplication with high dimensions is still expensive, if not impossible. In the following, we will introduce two typical techniques that can reduce the computational overhead for mobile devices.

5.8.2.1 Dimension Reduction

The complexity of normal matrix multiplication is $\mathcal{O}(n^3)$, where n is the dimension of the matrices. Thus, a straight forward way to reduce the complexity is to reduce the dimension of the matrices. For applications such as biometric authentication and identification, it is critical to preserve the identification accuracy while reducing the dimension. Several works [92–94] have been devoted to reducing the sizes of biometric templates. In [92], two techniques are introduced to decimate the FingerCode representation. The *tesselation reduction* approach reduces the dimension of FingerCode from the feature generation phase, which is illustrated in Section 5.5.1. Specifically, given a fingerprint image, this approach will reduce the number of sectors of the tessellation. The other approach is to directly apply some general dimension reduction methods such as PCA to the obtained FingerCodes. In this way, the most compact representation of FingerCode is found for a specific dataset.

We note that the above two approaches will both degrade the identification accuracy, however, to a satisfying level. In the experiments [92], the length of FingerCode vary from 640 to 8 in the tesselation approach. For PCA approach, the dimension of FingerCode varies

from 64 to 4. Generally speaking, the shorter the FingerCode is, the worse the accuracy would be. However, the experimental result demonstrated that FingerCode of dimensions 96 (from tesselation reduction) and 8 (from PCA) can achieve a satisfactory accuracy compared to that of the original 640. We also note that the approaches in [92] quantized each entry in FingerCode resulting a reduced accuracy. However, our proposed TPE scheme can be directly utilized to real numbers. Thus, TPE is applicable to the non-quantized case in [92], which has a higher accuracy.

5.8.2.2 Online/Offline Computation

The idea of online/offline computation [60, 95, 96] is to divide a computational expensive process into an online phase and an offline phase. During the offline phase, some pre-computation is done without given the input. During the online phase, given the input, it is relatively easy to padding the offline computation result in order to generate the final result. Typically, the offline computation is carried out when the mobile devices are idle or getting charged. Thus, such an approach can reduce the overall responding time and battery consumption.

Our proposed TPE scheme can utilize such approach to reduce the online computational overhead. For example, in the query phase, an end-user needs to compute $M_2^{-1} Y S_y M_1^{-1}$ given a transformed template Y. Then during the offline phase, the end-user can generate the random matrix S_y and compute $S_y M_1^{-1}$. The computation results can be stored for later usage. When a fresh template Y is generated, the end-user can compute $M_2^{-1} Y S_y M_1^{-1}$ during the online phase. This approach can reduce half of the computational overhead, which is critical for resource-constrained devices.

5.8.3 Numeric Results

In this section, we measure the performance of our proposed TPE scheme through simulation. Since the functions TPE.Setup and TPE.KeyGen are both one-time processes during the registration phase, we mainly focus on the execution time of TPE.TokenGen.

Since PassBio is a user-centric biometric authentication scheme, we measure the performance on both mobile phone and personal laptop. In the simulation, we utilize a mobile phone with Android 6.0 operating system, 2.5 GHz Cortex-A72 CPU and 4 GB RAM. We also utilize a personal laptop with macOS 10, 1.6 GHz Intel Core i5 and 4 GB RAM. The java library UJMP [97] and C++ library Armadillo are utilized for the simulation in the mobile phone and personal computer, respectively. We note that the performance relies on the selection of software packages. Our selection does not guarantee the best performance. Through complexity analysis, we know that the most important parameter affecting the performance is the dimension n of the vector. For the simulation on the mobile phone and laptop, we let n vary from 10 to 300 and from 100 to 2000, respectively. Due to the dimension reduction techniques introduced in Section 5.8.2.1, the dimension $n = 300$ is sufficient for most of the biometric templates. We also utilize the online/offline computation mechanism introduced in Section 5.8.2.2 to reduce to online computational overhead.

The numeric result on the laptop is shown in Fig. 5.2. The token generation time for moderate size template (n is around 200) is just around one millisecond with pre-computation. For high-dimensional template with $n = 2000$, the token generation time is less than 1 second with pre-computation. The numeric result on the mobile phone is shown in Fig. 5.3. The simulation results show that it is efficient to generate tokens for templates with moderate size. For example, when $n = 100$, the generation time is approximately 50 ms. When

$n = 300$, the generation time is around 900 ms. It can be observed in both figures that the online/offline mechanism can effectively reduce the online computational overhead. By pre-computation during the offline phase, the online computation time is reduced to about half of the whole processing time.

Protocol 8 Threshold Predicate Encryption (TPE) Scheme

Input: $\mathbf{x} = \{x_1, \ldots, x_n\}, \mathbf{y} = \{y_1, \ldots, y_n\}, \theta$.
Output: $\Lambda = \{0, 1\}$.

$\mathsf{TPE.Setup}() \rightarrow param$:

1: set $param = \{n, \theta\}$.

$\mathsf{TPE.KeyGen}(\lambda) \rightarrow sk$:

1: Randomly generate two non-singular $(n+3) \times (n+3)$ matrices M_1 and M_2 and calculate their inversions M_1^{-1} and M_2^{-1}.
2: Choose a random permutation π: $\mathbb{R}^{n+3} \rightarrow \mathbb{R}^{n+3}$
3: Set $sk = \{M_1, M_2, M_1^{-1}, M_2^{-1}, \pi\}$.

$\mathsf{TPE.Enc}(sk, \mathbf{x}) \rightarrow C_{\mathbf{x}}$:

1: Generate two random number β and r_x.
2: Extend the vector $\mathbf{x}$ to an $(n+3)$-dimensional vector $\mathbf{x'} = (\beta x_1, \beta x_2, \ldots, \beta x_n, -\beta\theta, r_x, 0)$.
3: Permute $\mathbf{x'}$ to obtain $\mathbf{x''} = \pi(\mathbf{x'})$.
4: Transform $\mathbf{x''}$ to a diagonal matrices X with $\mathrm{diag}(X) = \mathbf{x''}$.
5: Generate a random $(n+3) \times (n+3)$ lower triangular matrices S_x with the diagonal entries fixed as 1.
6: Compute $C_x = M_1 S_x X M_2$.

$\mathsf{TPE.TokenGen}(sk, \mathbf{y}) \rightarrow T_{\mathbf{y}}$:

1: Generate two random numbers α and r_y.
2: Extend $\mathbf{y}$ to an $(n+3)$-dimensional vector $\mathbf{y'} = (\alpha y_1, \alpha y_2, \ldots, \alpha y_n, \alpha, 0, r_y)$.
3: Permute $\mathbf{y'}$ to obtain $\mathbf{y''} = \pi(\mathbf{y'})$.
4: Transform $\mathbf{y''}$ to a diagonal matrix Y with $\mathbf{y''}$ being the diagonal.
5: Generate a random $(n+3) \times (n+3)$ lower triangular matrix S_y with the diagonal entries fixed as 1.
6: Compute $T_{\mathbf{y}} = M_2^{-1} Y S_y M_1^{-1}$.

$\mathsf{TPE.Dec}(C_x, T_y) \rightarrow \Lambda = \{0, 1\}$:

1: Compute $I = \mathsf{Tr}(C_x T_y)$, where $\mathsf{Tr}(\cdot)$ denotes the trace of a matrix.
2: Set $\Lambda = 1$ if $I \leq 0$; otherwise set $\Lambda = 0$.

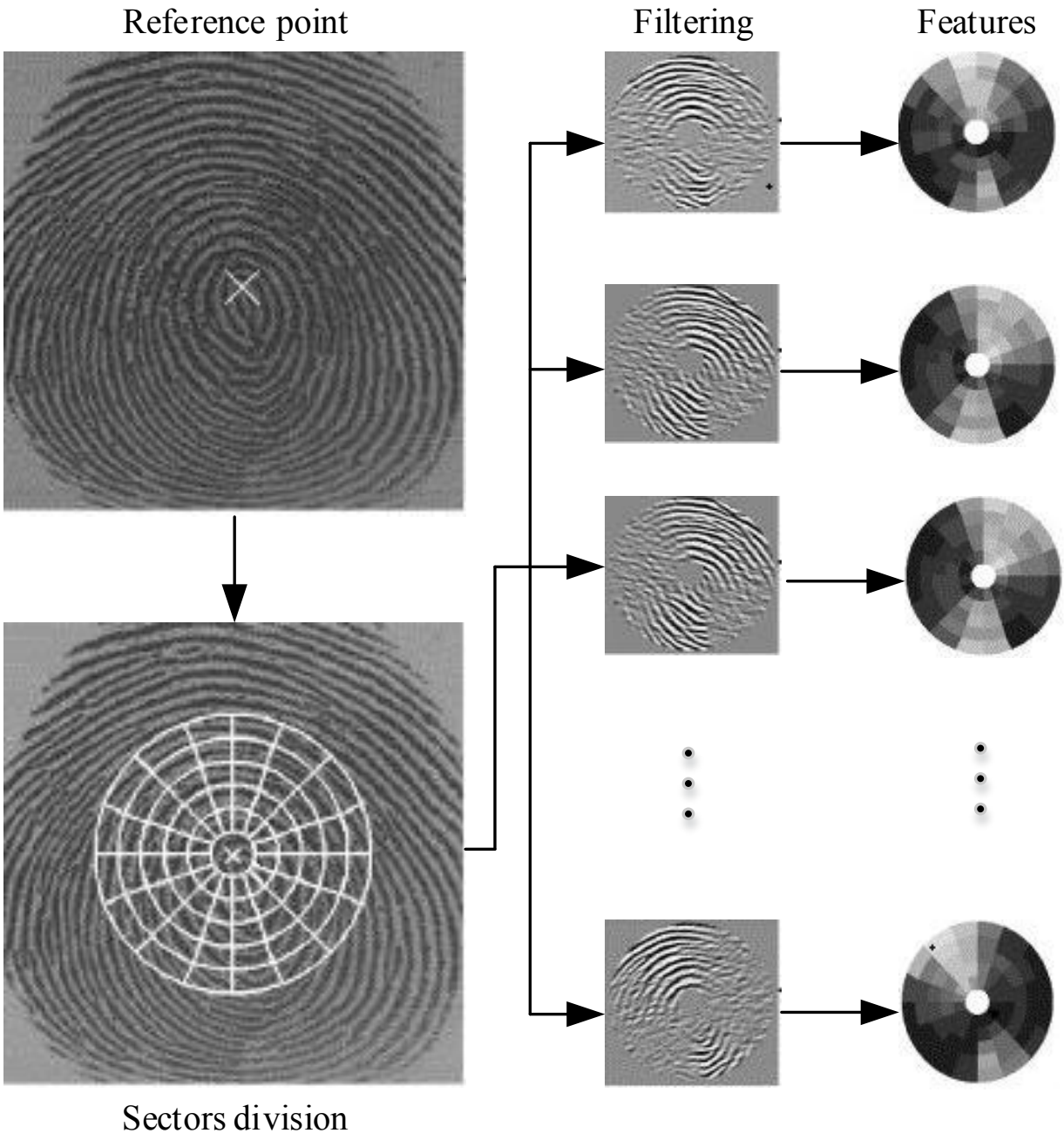

Figure 5.1: Feature extraction of fingerprints: (i) Identify reference point; (ii) Divide region of interest into sectors around reference point; (iii) Filter region of interest; (iv) Extract features.

Protocol 9 Privacy Preserving Biometric Authentication

Input: $\mathbf{x} = \{x_1, \ldots, x_n\}, \mathbf{y} = \{y_1, \ldots, y_n\}, \theta$.
Output: $\Lambda = \{\mathsf{Denied}, \mathsf{Authenticated}\}$.

Setup (End-user U):

1: Set the public parameters as $param = \{n, \theta\}$.
2: Randomly generate two matrices M_1 and M_2 with dimension $(n+5) \times (n+5)$ and a permutation $\pi : \mathbb{R}^{n+5} \to \mathbb{R}^{n+5}$.
3: Set secret key $sk = \{M_1, M_2, M_1^{-1}, M_2^{-1}, \pi\}$.

Registration (End-user U):

1: Generate random numbers β and r_x. Eextend $\mathbf{x}$ to an $(n+5)$-dimensional vector $\mathbf{x}' = (2\beta x_1, 2\beta x_2, \ldots, 2\beta x_n, -\beta \sum_{i=1}^{n} x_i^2, \beta, \beta\theta^2, r_x, 0)$.
2: Permute $\mathbf{x}'$ to obtain $\mathbf{x}'' = \pi(\mathbf{x}')$.
3: Transform $\mathbf{x}''$ to a diagonal matrices X with $\mathbf{x}''$ being the diagonal.
4: Generate a random $(n+5) \times (n+5)$ lower triangular matrix S_x with the diagonal entries fixed as 1. Compute $C_x = M_1 S_x X M_2$.
5: Register the record $\langle ID_U, C_x \rangle$ to the service provider SP, where ID_U is the identity of end-user U.

Query (End-user U):

1: Generate random numbers α and r_y.
2: Extend $\mathbf{y}$ to an $(n+5)$-dimensional vector $\mathbf{y}' = (\alpha y_1, \alpha y_2, \ldots, \alpha y_n, \alpha, -\alpha \sum_{i=1}^{n} y_i^2, \alpha, 0, r_y)$.
3: Permute $\mathbf{y}'$ to obtain $\mathbf{y}'' = \pi(\mathbf{y}')$.
4: Transform $\mathbf{y}''$ to a diagonal matrix Y with diagonal being y''.
5: Generate a random $(n+5) \times (n+5)$ lower triangular matrix S_y with the diagonal entries fixed as 1. Compute $T_y = M_2^{-1} Y S_y M_1^{-1}$
6: Send the query $\langle ID_U, T_y \rangle$ to SP.

Authentication (Service Provider SP):

1: On receiving a query from the end-user U, retrieve the registered record according to ID_U.
2: Compute $I = \mathsf{Tr}(C_x T_y)$.
3: Set $\Lambda = \mathsf{Authenticated}$ if $I \geq 0$; otherwise set $\Lambda = \mathsf{Denied}$.

Passive attack experiment $\mathsf{Passive}^{\mathsf{mult}}_{\mathcal{A},\mathsf{TPE}}(\lambda)$:

1: Given a security parameter λ, the adversary $\mathcal{A}$ outputs two sequences of messages $M_0 = (m_{0,1}, m_{0,2}, \ldots, m_{0,t})$ and $M_1 = (m_{1,1}, m_{1,2}, \ldots, m_{1,t})$, where the length of each message $|m_{0,i}| = |m_{1,i}|, i = 1, 2, \ldots, t$.
2: The challenger $\mathcal{C}$ runs $\mathsf{TPE.KeyGen}(\lambda)$ to generate the secret key sk.
3: $\mathcal{C}$ chooses a uniform bit $b \in \{0, 1\}$ and computes the ciphertext $c_i = \mathsf{TPE.TokenGen}(m_{b,i}, sk)$. The sequence $C = (c_1, c_2, \ldots, c_t)$ is returned to $\mathcal{A}$.
4: The adversary $\mathcal{A}$ outputs a bit b'.
5: The output of the experiment is 1 if $b = b'$, and 0 otherwise.

Active attack experiment $\mathsf{Active}_{\mathcal{A},\mathsf{TPE}}(\lambda)$:

1: The function $\mathsf{TPE.KeyGen}(\lambda)$ generates a secret key sk.
2: The adversary $\mathcal{A}$ is given oracle access to the function $\mathsf{TPE.TokenGen}(sk, \cdot)$ and outputs two messages m_0 and m_1 of the same length to the challenger $\mathcal{C}$.
3: The challenger $\mathcal{C}$ chooses a uniform bit $b \in \{0, 1\}$, then computes $c = \mathsf{TPE.TokenGen}(sk, m_b)$ and returns to $\mathcal{A}$.
4: $\mathcal{A}$ continues to have oracle access to $\mathsf{TPE.TokenGen}(sk, \cdot)$ and outputs a bit b'. Note however, $\mathcal{A}$ cannot use $\mathsf{TPE.TokenGen}(sk, \cdot)$ to generate tokens for messages somehow related to m_0 and m_1.
5: The output of the experiment is 1 if $b = b'$, and 0 otherwise.

Decryption Oracle $\mathcal{O}$:

1: The oracle $\mathcal{O}$ fixes a vector $\mathbf{x}$ and a number θ.
2: For any submitted vector $\mathbf{y}$, $\mathcal{O}$ generates two positive random numbers α and β and output $\gamma = \alpha\beta(\mathbf{x} \circ \mathbf{y} - \theta)$.

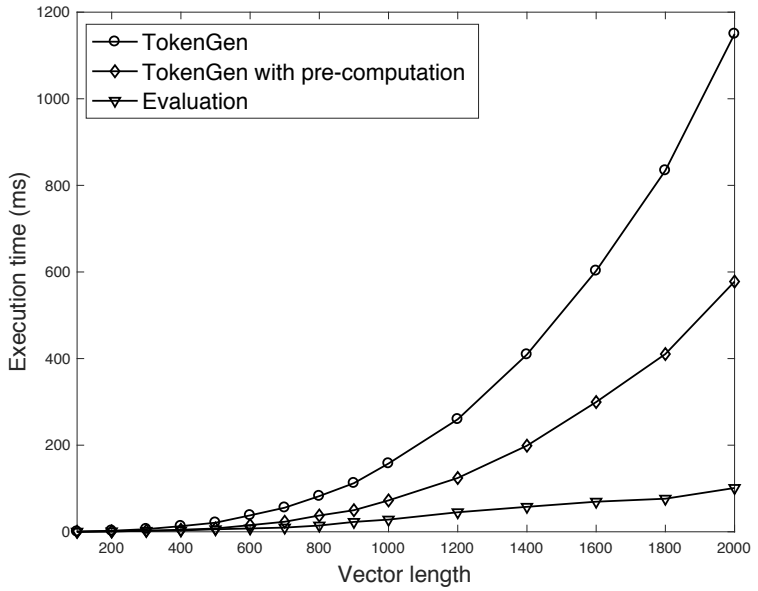

Figure 5.2: Performance of token generation and evaluation simulated on laptop (with vs. without pre-computation)

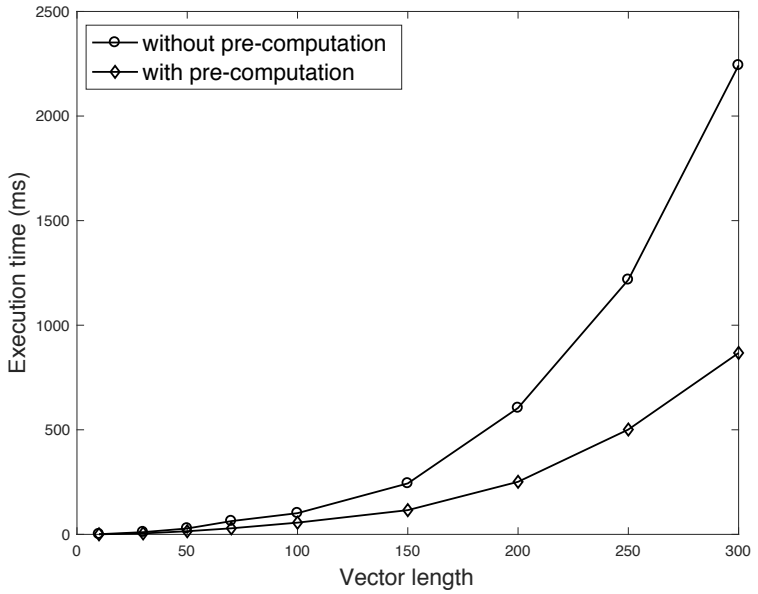

Figure 5.3: Performance of token generation and evaluation on mobile phone (with vs. without pre-computation)

Chapter 6

Conclusion

In this thesis, we aim at designing secure protocols to achieve a trade-off among data security, usability, and complexity in mobile cloud computing environment. Especially, we design secure computation outsourcing schemes for various computational problems to ensure data security while limiting local computational overhead, achieving the trade-off between data security and computational complexity. We design some special encryption techniques enabling the cloud to compute directly over the encrypted data, achieving the trade-off between data security and data usability.

More specifically, first, we designed CASO to securely outsource general scientific computational problems which cover the scope of linear and non-linear problems such as the system of equations (linear or non-linear), linear programming and convex optimization. Second, we proposed ExpSOS to securely outsource exponential operations such as modular exponentiations and scalar multiplications, based on which we utilize ExpSOS to securely outsource advanced cryptographic protocol such as predicate encryption and identity based encryption. Third, we focused on outsourcing a specific protocol named Attribute Based Encryption which is widely adopted in fine-grained access control mechanisms. All the above-proposed outsourcing schemes can provide significant performance gain for the end-users. Besides security, we also provide end-users with the ability to verify the returned results with probability close to 1. We also analyzed the trade-off between security and ef-

ficiency and provided cost-aware outsourcing schemes. At last, we developed novel methods to directly utilize encrypted data. The proposed data encryption scheme serves as an essential building block to construct a privacy-preserving user-centric biometric authentication scheme.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] M. J. Atallah and J. Li, "Secure outsourcing of sequence comparisons," *International Journal of Information Security*, vol. 4, no. 4, pp. 277–287, 2005.

[2] M. Blanton, M. J. Atallah, K. B. Frikken, and Q. Malluhi, "Secure and efficient outsourcing of sequence comparisons," in *Computer Security–ESORICS 2012*. Springer, 2012, pp. 505–522.

[3] M. Blanton and M. Aliasgari, "Secure outsourcing of dna searching via finite automata," in *Data and Applications Security and Privacy XXIV*. Springer, 2010, pp. 49–64.

[4] M. J. Atallah and K. B. Frikken, "Securely outsourcing linear algebra computations," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*. ACM, 2010, pp. 48–59.

[5] D. Benjamin and M. J. Atallah, "Private and cheating-free outsourcing of algebraic computations," in *Privacy, Security and Trust, 2008. PST'08. Sixth Annual Conference on*. IEEE, 2008, pp. 240–245.

[6] C. Wang, K. Ren, and J. Wang, "Secure and practical outsourcing of linear programming in cloud computing," in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 820–828.

[7] Y. N. Seitkulov, "New methods of secure outsourcing of scientific computations," *The Journal of Supercomputing*, vol. 65, no. 1, pp. 469–482, 2013.

[8] K. Zhou and J. Ren, "Linsos: Secure outsourcing of linear computations based on affine mapping," in *2016 IEEE International Conference on Communications (ICC)*. IEEE, 2016, pp. 1–5.

[9] ——, "Caso: Cost-aware secure outsourcing of general computational problems," *arXiv preprint arXiv:1511.02375*, 2015.

[10] S. Hohenberger and A. Lysyanskaya, "How to securely outsource cryptographic computations," in *Theory of Cryptography*. Springer, 2005, pp. 264–282.

[11] X. Chen, J. Li, J. Ma, Q. Tang, and W. Lou, "New algorithms for secure outsourcing of modular exponentiations," in *Computer Security–ESORICS 2012*. Springer, 2012, pp. 541–556.

[12] K. Zhou, M. Afifi, and J. Ren, "Expsos: Secure and verifiable outsourcing of exponentiation operations for mobile cloud computing," *arXiv preprint arXiv:1602.08472*, 2016.

[13] C. Gentry, "Fully homomorphic encryption using ideal lattices." in *STOC*, vol. 9, 2009, pp. 169–178.

[14] A. C. Yao, "Protocols for secure computations," in *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*. IEEE, 1982, pp. 160–164.

[15] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Advances in Cryptology–CRYPTO 2010*. Springer, 2010, pp. 465–482.

[16] M. J. Atallah, K. Pantazopoulos, J. R. Rice, and E. E. Spafford, "Secure outsourcing of scientific computations," *Advances in Computers*, vol. 54, pp. 215–272, 2002.

[17] H. Nie, X. Chen, J. Li, J. Liu, and W. Lou, "Efficient and verifiable algorithm for secure outsourcing of large-scale linear programming," in *Advanced Information Networking and Applications (AINA), 2014 IEEE 28th International Conference on*. IEEE, 2014, pp. 591–596.

[18] C. Wang, K. Ren, J. Wang, and K. M. R. Urs, "Harnessing the cloud for securely solving large-scale systems of linear equations," in *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*. IEEE, 2011, pp. 549–558.

[19] X. Chen, X. Huang, J. Li, J. Ma, W. Lou, and D. Wong, "New algorithms for secure outsourcing of large-scale systems of linear equations," *Information Forensics and Security, IEEE Transactions on*, vol. 10, no. 1, pp. 69–78, 2015.

[20] A. B. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption." in *Eurocrypt*, vol. 6110. Springer, 2010, pp. 62–91.

[21] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: Definitions and challenges," *Theory of Cryptography*, pp. 253–273, 2011.

[22] S. Goldwasser, S. D. Gordon, V. Goyal, A. Jain, J. Katz, F.-H. Liu, A. Sahai, E. Shi, and H.-S. Zhou, "Multi-input functional encryption," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2014, pp. 578–602.

[23] J. Katz, A. Sahai, and B. Waters, "Predicate encryption supporting disjunctions, polynomial equations, and inner products," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2008, pp. 146–162.

[24] E. Shen, E. Shi, and B. Waters, "Predicate privacy in encryption systems." in *TCC*, vol. 5444. Springer, 2009, pp. 457–473.

[25] A. Bishop, A. Jain, and L. Kowalczyk, "Function-hiding inner product encryption," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2015, pp. 470–491.

[26] M. Abdalla, F. Bourse, A. De Caro, and D. Pointcheval, "Simple functional encryption schemes for inner products," in *IACR International Workshop on Public Key Cryptography*. Springer, 2015, pp. 733–751.

[27] S. Kim, K. Lewi, A. Mandal, H. W. Montgomery, A. Roy, and D. J. Wu, "Function-hiding inner product encryption is practical." *IACR Cryptology ePrint Archive*, vol. 2016, p. 440, 2016.

[28] P. Datta, R. Dutta, and S. Mukhopadhyay, "Functional encryption for inner product with full function privacy," in *Public-Key Cryptography–PKC 2016*. Springer, 2016, pp. 164–195.

[29] E. J. Candes and T. Tao, "Decoding by linear programming," *IEEE transactions on information theory*, vol. 51, no. 12, pp. 4203–4215, 2005.

[30] S. Pissanetzky, *Sparse matrix technology*. Academic Press, 1984.

[31] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2009.

[32] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in cryptology-EUROCRYPT*. Springer, 1999, pp. 223–238.

[33] Z. Xu, C. Wang, Q. Wang, K. Ren, and L. Wang, "Proof-carrying cloud computation: The case of convex optimization," in *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013, pp. 610–614.

[34] X. Chen, W. Susilo, J. Li, D. S. Wong, J. Ma, S. Tang, and Q. Tang, "Efficient algorithms for secure outsourcing of bilinear pairings," *Theoretical Computer Science*, 2014.

[35] J. Hoffstein, J. C. Pipher, J. H. Silverman, and J. H. Silverman, *An introduction to mathematical cryptography*. Springer, 2008.

[36] J. Li, J. Li, X. Chen, C. Jia, and W. Lou, "Identity-based encryption with outsourced revocation in cloud computing," *Ieee Transactions on computers*, vol. 64, no. 2, pp. 425–437, 2015.

[37] B. Qin, R. H. Deng, S. Liu, and S. Ma, "Attribute-based encryption with efficient verifiable outsourced decryption," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 7, pp. 1384–1393, 2015.

[38] Y. Wang, Q. Wu, D. S. Wong, B. Qin, S. S. Chow, Z. Liu, and X. Tan, "Securely outsourcing exponentiations with single untrusted program for cloud storage," in *Computer Security-ESORICS 2014*. Springer, 2014, pp. 326–343.

[39] K. Zhou and J. Ren, "Secure outsourcing of scalar multiplication on elliptic curves," in *2016 IEEE International Conference on Communications (ICC)*. IEEE, 2016, pp. 1–5.

[40] C. Chevalier, F. Laguillaumie, and D. Vergnaud, "Privately outsourcing exponentiation to a single server: Cryptanalysis and optimal constructions," in *ESORICS*, 2016.

[41] L. Kuppusamy and J. Rangasamy, "Crt-based outsourcing algorithms for modular exponentiations," in *Progress in Cryptology–INDOCRYPT 2016: 17th International Conference on Cryptology in India, Kolkata, India, December 11-14, 2016, Proceedings 17*. Springer, 2016, pp. 81–98.

[42] M. S. Kiraz and O. Uzunkol, "Efficient and verifiable algorithms for secure outsourcing of cryptographic computations," *International Journal of Information Security*, pp. 1–19, 2014.

[43] L. Zhong, "Modular exponentiation algorithm analysis for energy consumption and performance," Citeseer, Tech. Rep., 2000.

[44] S. Kim, K. Lewi, A. Mandal, H. Montgomery, A. Roy, and D. J. Wu, "Function-hiding inner product encryption is practical," Cryptology ePrint Archive, Report 2016/440, 2016. http://eprint. iacr. org, Tech. Rep.

[45] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Advances in cryptology*. Springer, 1985, pp. 47–53.

[46] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Advances in Cryptology CRYPTO 2001*. Springer, 2001, pp. 213–229.

[47] T. Matsumoto, K. Kato, and H. Imai, "Speeding up secret computations with insecure auxiliary devices," in *Advances in Cryptology–CRYPTO'88*. Springer, 1990, pp. 497–506.

[48] P. de Rooij, "On schnorr's preprocessing for digital signature schemes," *Journal of Cryptology*, vol. 10, no. 1, pp. 1–16, 1997.

[49] V. Boyko, M. Peinado, and R. Venkatesan, "Speeding up discrete log and factoring based schemes via precomputations," in *Advances in Cryptology–EUROCRYPT'98*. Springer, 1998, pp. 221–235.

[50] P. Q. Nguyen, I. E. Shparlinski, and J. Stern, "Distribution of modular sums and the security of the server aided exponentiation," in *Cryptography and Computational Number Theory*. Springer, 2001, pp. 331–342.

[51] M. Van Dijk, D. Clarke, B. Gassend, G. E. Suh, and S. Devadas, "Speeding up exponentiation using an untrusted computational resource," *Designs, Codes and Cryptography*, vol. 39, no. 2, pp. 253–273, 2006.

[52] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 24, no. 1, pp. 131–143, 2013.

[53] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Advances in Cryptology–EUROCRYPT 2005*. Springer, 2005, pp. 457–473.

[54] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM conference on Computer and communications security*. Acm, 2006, pp. 89–98.

[55] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Security and Privacy, 2007. SP'07. IEEE Symposium on*. IEEE, 2007, pp. 321–334.

[56] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Public Key Cryptography–PKC 2011*. Springer, 2011, pp. 53–70.

[57] M. Ambrosin, M. Conti, and T. Dargahi, "On the feasibility of attribute-based encryption on smartphone devices," in *Proceedings of the 2015 Workshop on IoT challenges in Mobile and Industrial Systems*. ACM, 2015, pp. 49–54.

[58] X. Wang, J. Zhang, E. M. Schooler, and M. Ion, "Performance evaluation of attribute-based encryption: Toward data privacy in the iot," in *Communications (ICC), 2014 IEEE International Conference on*. IEEE, 2014, pp. 725–730.

[59] M. Brown, D. Hankerson, J. López, and A. Menezes, *Software implementation of the NIST elliptic curves over prime fields*. Springer, 2001.

[60] S. Hohenberger and B. Waters, "Online/offline attribute-based encryption," in *Public-Key Cryptography–PKC 2014*. Springer, 2014, pp. 293–310.

[61] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of abe ciphertexts." in *USENIX Security Symposium*, no. 3, 2011.

[62] J. Li, X. Chen, J. Li, C. Jia, J. Ma, and W. Lou, "Fine-grained access control system based on outsourced attribute-based encryption," in *Computer Security–ESORICS 2013*. Springer, 2013, pp. 592–609.

[63] J. Li, C. Jia, J. Li, and X. Chen, "Outsourcing encryption of attribute-based encryption with mapreduce," in *Information and Communications Security*. Springer, 2012, pp. 191–201.

[64] A. Beimel, "Secret-sharing schemes: a survey," in *Coding and cryptology*. Springer, 2011, pp. 11–46.

[65] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[66] A. De Caro and V. Iovino, "jpbc: Java pairing based cryptography," in *Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011*. Kerkyra, Corfu, Greece, June 28 - July 1: IEEE, 2011, pp. 850–855. [Online]. Available: \url{http://gas.dia.unisa.it/projects/jpbc/}

[67] E. F. Brickell, "Some ideal secret sharing schemes," in *Advances in Cryptology EUROCRYPT 89*. Springer, 1989, pp. 468–475.

[68] A. K. Jain and K. Nandakumar, "Biometric authentication: System security and user privacy." *IEEE Computer*, vol. 45, no. 11, pp. 87–92, 2012.

[69] A. K. Jain, K. Nandakumar, and A. Ross, "50 years of biometric research: Accomplishments, challenges, and opportunities," *Pattern Recognition Letters*, vol. 79, pp. 80–105, 2016.

[70] S. Rane, Y. Wang, S. C. Draper, and P. Ishwar, "Secure biometrics: concepts, authentication architectures, and challenges," *IEEE Signal Processing Magazine*, vol. 30, no. 5, pp. 51–64, 2013.

[71] N. P. Smart and F. Vercauteren, "Fully homomorphic encryption with relatively small key and ciphertext sizes." in *Public Key Cryptography*, vol. 6056. Springer, 2010, pp. 420–443.

[72] M. Naehrig, K. Lauter, and V. Vaikuntanathan, "Can homomorphic encryption be practical?" in *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*. ACM, 2011, pp. 113–124.

[73] W. K. Wong, D. W.-l. Cheung, B. Kao, and N. Mamoulis, "Secure knn computation on encrypted databases," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. ACM, 2009, pp. 139–152.

[74] Y. Elmehdwi, B. K. Samanthula, and W. Jiang, "Secure k-nearest neighbor query over encrypted data in outsourced environments," in *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*. IEEE, 2014, pp. 664–675.

[75] S. Choi, G. Ghinita, H.-S. Lim, and E. Bertino, "Secure knn query processing in untrusted cloud environments," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 11, pp. 2818–2831, 2014.

[76] B. Wang, Y. Hou, and M. Li, "Practical and secure nearest neighbor search on encrypted large-scale data," in *Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on*. IEEE, 2016, pp. 1–9.

[77] Q. Wang, S. Hu, K. Ren, M. He, M. Du, and Z. Wang, "Cloudbi: Practical privacy-preserving outsourcing of biometric identification in the cloud," in *European Symposium on Research in Computer Security*. Springer, 2015, pp. 186–205.

[78] A. K. Jain, S. Prabhakar, L. Hong, and S. Pankanti, "Fingercode: a filterbank for fingerprint representation and matching," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 2. IEEE, 1999, pp. 187–193.

[79] ——, "Filterbank-based fingerprint matching," *IEEE transactions on Image Processing*, vol. 9, no. 5, pp. 846–859, 2000.

[80] A. K. Jain, S. Prabhakar, and L. Hong, "A multichannel approach to fingerprint classification," *IEEE transactions on pattern analysis and machine intelligence*, vol. 21, no. 4, pp. 348–359, 1999.

[81] S.-S. Choi, S.-H. Cha, and C. C. Tappert, "A survey of binary similarity and distance measures," *Journal of Systemics, Cybernetics and Informatics*, vol. 8, no. 1, pp. 43–48, 2010.

[82] J. Katz and Y. Lindell, *Introduction to modern cryptography*. CRC press, 2014.

[83] K. Zhou and J. Ren, "Secure fine-grained access control of mobile user data through untrusted cloud," in *Computer Communication and Networks (ICCCN), 2016 25th International Conference on*. IEEE, 2016, pp. 1–9.

[84] K. Zhou, M. Afifi, and J. Ren, "Expsos: Secure and verifiable outsourcing of exponentiation operations for mobile cloud computing," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2518–2531, 2017.

[85] K. Zhou and J. Ren, "Secure fine-grained access control of mobile user data through untrusted cloud," in *Computer Communication and Networks (ICCCN), 2016 25th International Conference on*. IEEE, 2016, pp. 1–9.

[86] J. Yuan and S. Yu, "Efficient privacy-preserving biometric identification in cloud computing," in *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013, pp. 2652–2660.

[87] C. Hahn and J. Hur, "Poster: Towards privacy-preserving biometric identification in cloud computing," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1826–1828.

[88] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on*. IEEE, 2000, pp. 44–55.

[89] E. Shi, J. Bethencourt, T. H. Chan, D. Song, and A. Perrig, "Multi-dimensional range query over encrypted data," in *Security and Privacy, 2007. SP'07. IEEE Symposium on*. IEEE, 2007, pp. 350–364.

[90] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," *Theory of cryptography*, pp. 535–554, 2007.

[91] S. Tu, M. F. Kaashoek, S. Madden, and N. Zeldovich, "Processing analytical queries over encrypted data," in *Proceedings of the VLDB Endowment*, vol. 6, no. 5. VLDB Endowment, 2013, pp. 289–300.

[92] T. Bianchi, S. Turchi, A. Piva, R. D. Labati, V. Piuri, and F. Scotti, "Implementing fingercode-based identity matching in the encrypted domain," in *Biometric Measurements and Systems for Security and Medical Applications (BIOMS), 2010 IEEE Workshop on*. IEEE, 2010, pp. 15–21.

[93] N. Kambhatla and T. K. Leen, "Dimension reduction by local principal component analysis," *Dimension*, vol. 9, no. 7, 2006.

[94] T. Mandal, Q. J. Wu, and Y. Yuan, "Curvelet based face recognition via dimension reduction," *Signal Processing*, vol. 89, no. 12, pp. 2345–2353, 2009.

[95] S. S. Chow, J. K. Liu, and J. Zhou, "Identity-based online/offline key encapsulation and encryption," in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*. ACM, 2011, pp. 52–60.

[96] J. K. Liu and J. Zhou, "An efficient identity-based online/offline encryption scheme." in *ACNS*, vol. 5536. Springer, 2009, pp. 156–167.

[97] "Universal java matrix package," https://ujmp.org/.