Efficient cryptographic primitives: Secure comparison, binary decomposition and proxy re-encryption

Feng Li

EFFICIENT CRYPTOGRAPHIC PRIMITIVES: SECURE COMPARISON, BINARY DECOMPOSITION AND PROXY RE-ENCRYPTION

by

FENG LI

A DISSERTATION

Presented to the Faculty of the Graduate School of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

2015

Approved by

ABSTRACT

Data outsourcing becomes an essential paradigm for an organization to reduce operation costs on supporting and managing its IT infrastructure. When sensitive data are outsourced to a remote server, the data generally need to be encrypted before outsourcing. To preserve the confidentiality of the data, any computations performed by the server should only be on the encrypted data. In other words, the encrypted data should not be decrypted during any stage of the computation. This kind of task is commonly termed as query processing over encrypted data (QPED).

One natural solution to solve the QPED problem is to utilize fully homomorphic encryption [1]. However, fully homomorphic encryption is yet to be practical. The second solution is to adopt multi-server setting. However, the existing work is not efficient. Their implementations adopt costly primitives, such as secure comparison, binary decomposition among others, which reduce the efficiency of the whole protocols. Therefore, the improvement of these primitives reults in high efficiency of the protocols. To have a well-defined scope, the following types of computations are considered: secure comparison (CMP), secure binary decomposition (SBD) and proxy re-encryption (PRE). We adopt the secret sharing scheme and paillier public key encryption as building blocks, and all computations can be done on the encrypted data by utilizing multiple servers. We analyze the security and the complexity of our proposed protocols, and their efficiencies are evaluated by comparing with the existing solutions.

ACKNOWLEDGMENTS

I would like to express sincere gratitude to my advisor, Dr. Wei Jiang, for his guidance, deep knowledge, patience, and encouragement during my PhD study. It is a great experience for me to work with him.

I also would like to thank the members of his advisory committee, Dr. Xiaoqing (Frank) Liu, Dr. Dan Lin, Dr. Sriram Chellappan and Dr. Xuerong Meggie Wen for their valuable time and advice to improve the quality of the dissertation.

Thanks to the students of my laboratory, Huchun, Yousef Elmehdwi for numerous instructive discussion and their kind support.

Finally, I would like to thank my parents, my wife (Yan Xu), and my son (Henry Li) for their infinite love, care and encouragement. With their help, I have overcome the difficulities in my study and life and live happily in the beautiful town of Rolla, Missouri.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure Page

LIST OF TABLES

1. INTRODUCTION

Maintaining data is a challenging task, especially for small and medium size businesses, because of high day-to-day operational costs. As a consequence, some data owners may be more interested in outsourcing their data to a remote storage (e.g., cloud). Data outsourcing becomes an essential paradigm for an organization to reduce operation costs on supporting and managing its IT infrastructure. However, data outsourcing raises security challenge such as protecting user privacy and data confidentiality. In general, the sensitive data need to be encrypted before outsourcing. To preserve the confidentiality of the data, any computations performed by the server should only be on the encrypted data. In other words, the encrypted data should not be decrypted during any stage of the computation. This kind of task is commonly termed as query processing over encrypted data (QPED). One solution to solve the QPED problem is to utilize fully homomorphic encryption [1]. However, fully homomorphic encryption is yet to be practical. The second solution is to adopt a multi-server setting where one server has the encryption key and the other server has the encrypted data. Given a query request, the two servers can collaborativly and securely perform the necessary computation to achieve the task without disclosing the actual data to both servers. In addition, the users or clients do not need to participate in any intermediate computations.

To implement multi-server based QPED, we have two orthogonal approaches: one is to use an additive homomorphic encryption scheme (e.g.,Paillier [2]), and the second approach is to adopt a secret sharing scheme (Shamir [1]). In both approaches, any polynomially-bounded query processing tasks can be performed due to the fact that given two encrypted bit $[b_1]$ and $[b_2]$, the multi-server setting allows the computation of $[b_1 + b_2 \mod 2]$ and $[b_1 \cdot b_2]$ without disclosing any information regarding b_1 and b_2. In many cases, binary level operations on bit-wise encrypted data are not efficient due to large circuit size. In other words, it is computationally more efficient to perform secure computation on $[x]$ instead of $[x]_B \equiv [x_{l-1}] \ldots, [x_0]$ where x is a l-bit value. However, for certain primitive computations like division, exponentiation and modulo are much easier to compute when the data in consideration are encrypted bit-wise. Again, if we store the encrypted data in binary format at the first place, the size becomes too big to be practical. In general, to process a QPED task, the task is divided into sub-component. The output from one component becomes input to the subsequent component. For efficiency purposes, one component requires input to be $[x]$, and another component requires to be $[x]_B$. Thus, it is necessary to covert $[x]$ into $[x]_B$ and vice versa. Converting from $[x]_B$ to $[x]$ is straightforward, but converting from $[x]$ to $[x]_B$ is a challenging task which is generally termed as secure binary decomposition (SBD).

The concept of SBD was first introduced by Damgård [3] under the secret sharing based secure multiparty computation scheme. Let x be an l-bit value secretly shared among n parties, each of whom has a secret share $x_i \in \mathbb{Z}_P$ for $1 \leq i \leq n$. We use the notation $[x]_P$ to represent secret shares of x in the group $\mathbb{Z}_P$. An SBD protocol allows the n parties to convert their secret shares of x into secret shares of the individual bits of x. In other words, SBD takes $[x]_P$ as its input and produces $[x]_B \equiv [x_{l-1}] \ldots, [x_0]$ as its output. Since its initial introduction, many improvements to SBD has been proposed to reduce both communication and computation complexities. The most efficient bit-decomposition protocol was proposed in [4]. The protocol transforms the bit-decomposition problem into a post-fix comparison problem. The post-fix comparison can be achieved efficiently using a prefix-product protocol. Using the post-fix comparison, the complexity of bit-decomposition protocol can be reduced to O(1) rounds and O(l) multiplications.

As in other scenario, Alice prefers only the authorized user be able to access the database, and any other unauthorized users can not get data from the database. Then, a new arising challenge is how an authorized user to decrypt Alice's ciphertext. In public key encryption shceme, the owner of secret key corresponding to the public key is the only one who can decrypt the ciphertext legally. One solution to the above new challenge is to transform the ciphertext under one key into a new ciphertext of the same message under another key. However, the transformation requires re-encryption of the message with a new key, implying access to the original plaintext, which is not secure because of an untrusted party to fulfill this transformation. Alice could simply provide her secret key to the user or server, but this would compromise the security of Alice or put unrealistic level of trust on the server. A proxy (server) re-encryption scheme can easily solve the above problem.

The proxy re-encryption (PRE) cryptosystem allows a proxy, given a re-encryption key, to transform ciphertext of a message encrypted under one public key to another ciphertext of the same message under different public key[5, 6], with allowing proxy to learn nothing. The goal of PRE is to securely transform ciphertext from one key to another, without relying on any trusted parties or compromising the security of data owner. Using the PRE scheme, the authorized user, say Bob, can obtain the ciphertexts under his public key, and then decrypt them using his own secret key. Because of the nature of this transformation, PRE scheme can be applied to many applications, such as distributed storage systems, secure email forwarding and digital right management.

Anonymous credential (AC) is a mechanism that allows a user to be authenticated to a service provider without disclosing her real identity. In addition, an AC allows the user to generate a new credential for each authentication request regardless if it is the same or different service providers. In this way, service requests from the same user cannot be linked together. Thus, even if service providers collude, they still cannot discover the user's real identify through locality tracing techniques in a mobile environment. The AC system has been adopted extensively to provide privacy-preserving user authentication. However, the current implementation of AC has limitations, such as ineffective credential revocation, and weak security guarantee against replay attacks. In addition, it lacks other desirable features like traceability to reveal malicious users.

1.1. PROBLEM DEFINITION

Let M and N denote two integer domains where $M \ll N$, and $size(\cdot)$ be a function that returns the number of bits to represent the input parameter. For example, $size(7) = 3$. In addition, the following notations are used extensively (note that M can replace N without altering the main meaning of the notations):

- SBD: secure bit-decomposition
- P_i: a participating party indexed by i
- $[x]_N$: x is secretly shared in $\mathbb{Z}_N$.
- $[x]_{N,B} \equiv [x_{l-1}]_N, \ldots, [x_0]_N$: each bit of x is secretly shared in $\mathbb{Z}_N$, where $l = size(N)$, x_{l-1} denotes the most significant bit, and $x = \sum_{i=0}^{l-1} 2^i x_i$
- $[x]_N^{P_i}$: the secret share of x owned by party P_i. For example, suppose x is secretly shared between P_1 and P_2 in $\mathbb{Z}_N$, then $x = [x]_N^{P_1} + [x]_N^{P_2} \bmod N$
- $[x]_{N,B}^{P_i} \equiv [x_{l-1}]_N^{P_i}, \ldots, [x_0]_N^{P_i}$: the secret shares of each bit of x owned by party P_i. For example, $x_j = [x_j]_N^{P_1} + [x_j]_N^{P_2} \bmod N$ for $0 \leq j \leq l-1$
- $T_B^i \equiv t_i, \ldots, t_0$: a random bit sequence with size $i+1$, where $0 \leq i \leq l-1$ and $t_i \in_R \{0,1\}$

For succinctness, we assume there are three participating parties P_1, P_2 and P_3, and our work can be directly extended to the situation where there are more than three participating parties. Unlike the existing work, the secret shares of x are not distributed among the three parties. Instead, we assume x is shared between P_1 and P_2. The party P_3 is required to perform certain computations like multiplying two secretly shared values. The proposed seure comparison protocol is defined as follows (N_1 and $N_2$are public parameters, $N_2 \ll N_1$):

$$\begin{aligned}(\langle P_1, [u]^{P_1}_{N_1}\rangle, \langle P_2, [u]^{P_2}_{N_1}\rangle) \leftarrow &\text{FAST-CMP}(\langle P_1, [a]^{P_1}_{N_2,B}, [b]^{P_1}_{N_2,B}, t, \pi(.)\rangle, \\ &\langle P_2, [a]^{P_2}_{N_2,B}, [b]^{P_2}_{N_2,B}, b_B, t, \pi(.)\rangle, \langle P_3, \perp\rangle)\end{aligned} \tag{1.1}$$

The above protocol takes private inputs $[a]^{P_1}_{N_2,B}$, $[b]^{P_1}_{N_2,B}$ and $[a]^{P_2}_{N_2,B}$, $[b]^{P_2}_{N_2,B}$ from P_1 and P_2 respectively and returns shares of $f = b \overset{?}{>} a$, for $f \in \{0, 1\}$, $[u]^{P_1}_{N_1}$ to P_1 and $[u]^{P_2}_{N_1}$ to P_2. t , t_B and $\pi(.)$ are parameters known by P_1 and P_2. The proposed comparison protocol to perform the same functionality as post-fix comparison is briefly described as follows (t_B is known by P_1 and P_2):

$$\begin{aligned}(\langle P_1, [u]^{P_1}_{N_1}\rangle, \langle P_2, [u]^{P_2}_{N_1}\rangle) \leftarrow &\text{CMP}(\langle P_1, [a]^{P_1}_{N_2,B}, [b]^{P_1}_{N_2,B}, t_B, \pi(.)\rangle, \\ &\langle P_2, [a]^{P_2}_{N_2,B}, [b]^{P_2}_{N_2,B}, b_B, t_B, \pi(.)\rangle, \langle P_3, \perp\rangle)\end{aligned} \tag{1.2}$$

The secure CMP protocol takes the same private inputs as FAST-CMP protocol. Instead of inputing a flipping bit t, it takes a flipping sequence t_B. The output is the same as FAST-CMP, $[u]^{P_1}_{N_1}$ to P_1 and $[u]^{P_2}_{N_1}$ to P_2. The proposed secure bit-decomposition (SBD) protocol is formally defined as follows (N are public parameters):

$$(\langle P_1, [x]^{P_1}_{N,B}\rangle, \langle P_2, [x]^{P_2}_{N,B}\rangle) \leftarrow \text{SBD}(\langle P_1, [x]^{P_1}_{N}\rangle, \langle P_2, [x]^{P_2}_{N}\rangle, \langle P_3, \perp\rangle) \tag{1.3}$$

The protocol takes private inputs $[x]^{P_1}_{N}$ and $[x]^{P_2}_{N}$ from P_1 and P_2 respectively and returns $[x]^{P_1}_{N,B}$ to P_1 and $[x]^{P_2}_{N,B}$ to P_2. Among the above protocols, the role of P_3 is to assist P_1 and P_2 to perform certain intermediate computations.

Since secure multiplication is one of the dominant factors in analyzing the complexity of secret sharing based protocol (initially discussed in [3, 7]), to analyze the complexity of the proposed CMP and SBD protocols, we adopt the number of secure multiplications as one key variable. In addition, we will also count the number of rounds to execute the protocol and the message sizes communicated among the three parties.

The PRE schemes come in bidirectional and unidirectional varieties[8, 9]. In a bidirectional scheme, the re-encryption is reversible, that is, the re-encryption key can be used to transform ciphertext from Alice (delegator) to Bob (delegatee) (that is, from Alice's public key to Bob's public key), as well as from Bob to Alice. This is not a desired feature since Bob may not allow his ciphertext been re-encrypted and decrypted by other users. Another deficiency is that Alice and Bob must contribute their secret keys to generate the re-encryption key. While in a unidirectional scheme, the re-encryption is one-way, that is, the ciphertext can be re-encrypted from Alice to Bob, but not the reverse, and Bob is not required to reveal his secret key in the process of computing re-encryption key.

The good thing of PRE shceme is that the delegator (Alice) and delegatees (like Bob) are not required to be online when the delegation happens. However, there are some issues we need to address. Colluding among proxy and delegatee may compromise the data owner's secret key; If delegatee wants to decrypt a certain kind of data, he needs to search the corresponding ciphertext in the database. There is no existing unidirectional PRE scheme solving both above issues at the same time. Therefore, we propose a serachable unidirectional PRE scheme satisfying the "Colluding Safe".

In case of the proxy is malicious, the unidirectional PRE scheme should be able to against invalid ciphertexts. Therefore, the validation of ciphertext should be considered when design a secure unidirectional PRE scheme. In the paper, we adopt strongly unforgeable signature technique to implement the validation of re-encrypted ciphertext and then construct efficient and feature-rich unidirectional PRE schemes with simple design.

1.2. OBJECTIVES AND CONTRIBUTIONS

To further improve the efficiency, in this thesis, we propose a new efficient secure comparison protocol to perform the same functionality as the post-fix comparison protocol. Then our new comparison protocol can be applied to implement a more efficient SBD protocol. Our proposed protocol provides a solution matching the O(·) bound while decreasing the hidden constants.

(1). Comparison. In existing bit-decomposition protocols, the comparison of secret data is required as the basic security primitive. We propose a new efficient secure comparison protocol to perform the same functionality as the post-fix comparison protocol

(2). Bit-decomposition. We improved the bit-decomposition protocol presented in Reistad and Toft [4] by applying our efficient secure comparison protocol
.

Table 1.1 compares the complexity of the our proposed solution with those of the existing SBD protocols. All these complexities are analyzed under the passive adversary model.

We construct different proxy re-encryption schemes depends on adversary model assumption. The proposed constructions also preserve multiple features as discussed in Section 3.6. We also design a searchable unidirectional proxy re-encryption scheme in multi-proxy setting under semi-hones proxy model.

Protocols	Rounds	Multiplications
Damgards et al [3]	36	$94l\log l + 93l$
Nishide and Ohta [7]	23	$47l\log l + 93l$
Reistad and Toft [4]	7	$16.5l + 3$
The Proposed SBD	3	$3l$

Table 1.1: Complexity Comparison

1.3. ORGANIZATION

The thesis is organized as follows. Chapter 2 provides an overview of the existing work closely related to the proposed research. Chapter 3 reviews the cryptographic primitives utilized in the proposed protocols as well as the security definition adopted in this dissertation. Chapter 4 presents our new schemes of secure comparison, from which two protocols denoted as FAST-CMP and CMP are derivated. Chapter 5 focus on the binary decomposition of $[x]$. We applied our proposed secure comparison protocol to the SBD protocol achieving more efficient than existing works. Also, the complexity and security are analyzed under different parameter settings. Chapter 6 provides the basic proxy re-encryption scheme achieving collusion safe, and other improved versions with achieving more security features.

2. LITERATURE REVIEW

2.1. SECURE COMPARISON (CMP)

The secure comparison is considered as a basic primitives in implementing distributed protocols, such as binary decomposition protocol. Garay et al [10] developed a protocol for integer comparison based on homomophic encryption , which takes two sets of encrypted bits representing x and y as inputs, and outputs a encrypted bit indicating $x \overset{?}{>} y$. The protocol achieves logarithmic rounds complexity by spliting the bit strings into equally parts, comparing them recursively and then combining results to reach the final result. They also proviede a constant rounds protocol using different apprach: conditional oblivious transfer of Blake and Kolesnikov's [11]. Damgård et al [12, 13] proposed a protocol for securely compare integers based on a new additive homomophic encryption scheme, which is more efficient than previous protocols. However, this efficient comes with small ciphertext space. It is worth to pointing out is that the inputs to these protocol must be in binary representation form which is the main problem we address in this paper. Kolesnikov et al [14] improves the complexity of comparison of non-encrypted walues using Garbled Circuit (GC) by reducing the number of non-Xor gates. Blaken and Koleanikov [15] provides an efficient way to compare two encrypted integer, however, this scheme leaks the result to at least one of the involved parties.

2.2. SECURE BINARY DECOMPOSITION (SBD)

In [3], Damgård et al proposed the first constant rounds bit-decomposition under the secret sharing scheme with $O(l \log l)$ secure multiplications, where l is the bit length. Nishide and Ohta [7] improved the round complexity by removing one secure addition which is the most expensive primitive in bit-decomposition and providing a more accurate way to compute the complexity which is reduced to $O(l)$ secure multiplications. Several other constant-round solutions were proposed by [4, 16, 17, 18] where the bit-decomposition problem was reduced to the post-fix comparison problem. The security of these protocols was inherited from the primitives which means security against the adaptive and active adversaries when the primitives are so under the secret sharing as well as Paillier encryption scheme.

Bogdanov [19] states a framework for secure arithmetic operation including sharing secret multiplication and bit transformation which run simultaneously by three parties. Schoenmakers and Tuyls [20] constructed BITREP gate for securely computing the binary representation of an encrypted integer value with n participants, however, the efficient BITREP gate is only feasible if a cryptosystem such as Paillier is used as the underlying cryptosystem (even not efficient under homomorphic ElGamal cryptosystem as it involving solving a discrete logarithm problem [21] which is not applicable to our problems. As secure comparison is a basic primitive in bit-decomposition, many works focus on the improvement of sensitive data comparison.

Various existing protocols and their improvements under two-party settings [22, 23, 24, 25, 26, 27] or multi-parties environments [3, 4, 7, 28, 29, 30, 31, 32, 33, 34], including addition, multiplication, comparison, equlity texting , enables the design of new efficient bit-decompositon protocol. Without loss of generality, we construct new efficient and secure comparison protocol for our bit-decomposition scheme.

A MPC apprach SPDZ ("SpeedZ"), an extension of protocol proposed by Bendlin et al [34], has been introduced in [35] and improved in [36], which has the advantage that costly operation can be loaded into the preprocessing phase and the cheap secure computation can be loaded into online phase and can therefore very efficient. The SPDZ apprach is based on fully homomophic encryption [37] with an arbitrary number of participants. However, they focus on evaluating functionalities with limited finite field or more mathematically complex offline phase, which can not applied to our scheme with large primes and less workload on both offline and online phase. In our comparison protocol, the inputs are not necessarily binary represented.

In [16], an unconditionally secure and constant-round bit-decomposition protocol using postfix comparison method was introduced. Then Reistad and Toft [4] solve this postfix comparison problem in a more efficient way based on a comparison protocol proposed by Reistad [18]. The new solution, denoted by Bits-RT, has the lowest complexity compared with the other existing solutions. The main steps of Bits-RT [4] are presented in Algorithm 1.

Algorithm 1 $[x]_B \leftarrow$ Bits-RT$([x])$

Input: Secret shares of x from parties
Output: Bit-wise sharing of x: $[x_{l-1}], \ldots, [x_0]$

```
1. $[r] = \sum_{i=0}^{l-1} 2^i [r_i]$ and $[r]_B = ([r_0], \ldots, [r_{l-1}])$
2. $b = \text{Reveal}([x] + [r])$
3. $c' = b + M$
4. $[f] \leftarrow [r]_B \overset{?}{>} b$
5. For $i = 0$ to $l$
   $[c_i] = [f](c'_i - b_i) + b_i$
   EndFor
6. $[c]_B = ([c_l], \ldots, [c_0])$
7. $[x \bmod 2^l] = [x]$
8. For $i = l - 1$ to 1
   $[u_i] = [r \bmod 2^i]_B \overset{?}{>} [c \bmod 2^i]_B$
   $[x \bmod 2^i] = [c \bmod 2^i]_B - [r \bmod 2^i]_B + 2^i [u_i]$
   $[x_i] = ([x \bmod 2^{i+1}] - [x \bmod 2^i]) * 2^{-i}$
   EndFor
9. $[x_0] = [x \bmod 2]$
```

The operations of Bits-RT are over the group $\mathbb{Z}_M$ where $M > 2^{l+\kappa+logn}$, κ is security parameter, and n is the number of parties. To perform bit-decomposition of $[x]$, first a random mask $[r]$ is added to $[x]$ such that $[b] = [x + r \bmod M]$. Then the actual value of b is revealed to every party. This step reveals nothing about x since r is uniformly random. Steps 3-5 determine the correct c value: $c = b$ if $b > r$, $c = b + M$ otherwise. The method to retrieve the individual bits of x is based on the following equation:

$$2^i x_i = (x \bmod 2^{i+1}) - (x \bmod 2^i) \tag{2.1}$$

which can be computed securely based on the secret shares of the actual values according to the equation given below:

$$[x \bmod 2^i] = [c \bmod 2^i]_B - [r \bmod 2^i]_B + 2^i [u_i] \tag{2.2}$$

where $[u_i]$ the secret shares of the the comparison result between $r \bmod 2^i$ and $c \bmod 2^i$. That is $u_i = 1$ if $r \bmod 2^i > c \bmod 2^i$, and $u_i = 0$ otherwise. The comparison using inputs $[r \bmod 2^i]$ and $[c \bmod 2^i]$ for all i is a secure postfix comparison problem. Therefore, the bit-decomposition problem is transformed into the post-fix comparison problem which is the most costly primitive in the bit-decomposition protocol.

Reistad and Toft [4] improved the post-fix comparison which reduces the total complexity in both the number of rounds and the number of multiplications. Excluding the preprocessing phase, Bits-RT takes 8 rounds and $16.5l + 3$ multiplications. This protocol is statistically secure against passive adversaries since the post-fix comparison primitive is only statistically secure. In this paper, we proposed a novel way to compare two secrets replacing the post-fix comparison by using only $3l$ secure multiplications as described in next section.

2.3. PROXY RE-ENCRYPTION (PRE)

In proxy cryptosystems [38, 39, 40], a delegator Alice delegates the right to decrypt ciphertexts to the delegatee Bob with the help of proxy by providing Proxy and delegatee pieces of secret information (like secret shares), which requires the delegatee to store additional information for each delegation. PRE scheme is a special kind of proxy cryptosystems where delegatees only need to store their own decryption keys.

Blaze et al. [5, 6] introduced atomic proxy crptography, in which an semi-trusted Proxy is given a re-encryption proxy key that allows him/her to transform a message m encrypted under Alice's public key pk_A into an encryption of the same m under Bob's public key pk_B, without allowing proxy to learn anyting about the plaintext m. The scheme works fine when Alice and Bob are mutually trusted since it is bidirectional and the proxy is semi-honest. However, this assumption is impractical in the real world. Ivan and Dodis [39] proposed an efficient proxy encryption scheme based on RSA, bilinear and DDH asumptions. In the scheme, Alice's secret key s is divided into two shares: s_1, given to the proxy, and s_2, given to Bob. A ciphertext intended for Alice can be partialy decrypted by the proxy via s_1. Then Bob can complete the decryption process by using s_2 and recover the message. However, this scheme is not an exactly proxy re-encryption scheme in which Bob must store the additional secret keys to recover the message and faces the same colluding problem: s can be recovered, $s = s_1 + s_2$ when Bob and Proxy collude; this approach requires that Bob obtain and store an additional secrets for every decryption delegation he accepted which put high requirement of storage capability on the user side.

Ateniese et al.[8, 9] first time showed an unidirectional PRE shcemes, which is Elgamal-based scheme operating over two bilinear map groups[41], solving the above collusion issue. There are two levels encryption: first-level ciphertext can only be decrypted by the owner's secret key, while second-level ciphertext can be transformed into first-level ciphertext with different public key, allowing that to be decrypted by secret keys corresponding to this public key. Ateniese et al.[8, 9] also first time applied the proposed unidirectional PRE scheme to an encrypted file storage system where an untrusted access control server (proxy) managing the storage, without granting full decryption rights to the server. Canetti and Hohenberger[42] construct a PRE scheme achieving security against chosen ciphertext attacks (CCA) based on the Decisional Bilinear Diffie-Hellman (DBDH) assumption in the standard model. But, this scheme is also bidirectional as [5, 6] in that the re-encryption key to convert ciphertexts from Alice's publc key to Bob's public can also be used to convert that in reverse direction.

Libert and Vergnaud [43] proposed a traceable proxy re-encryption schemes, where misbehaving proxies can be identified by the delegator, only against replayable chosen ciphertext attacks (RCCA)[44, 45], a weaker definition of CCA, tolerating a "harmless attacking" of the challenge ciphertext. Deng et al.,[46] presented a CCA-secure PRE without pairings using Schnorr signature[47] to make the original ciphertext be publicly verifiable. However, this unidirectional PRE scheme is sufferring from the collusion attack. Shao and Cao[48] stated another unidirectional PRE scheme without pairings using signature of knowledge, which is secure against chosen ciphertext attack and collusion attack, in the random oracle model, based on Decisional Diffie-Hellman assumption and interger factorization assumption.Chow et al.,[49] identified flaws in security proof of [48] which translate to a real-world chosen-ciphertext attack, then proposed an efficient unidirectional CCA-secure PRE scheme without pairings using the " token-controlled encryption" technique under the standard computational Diffie-Hellman assumption in the random oracle model. The scheme proposed by [42] is a verifiable PRE scheme, in which it appends to the ciphertext a checksum value consisting of an element of bilinear group raised to the random encryption exponent. Like

Table 2.1: Properties of PRE Schemes from the Literature

Schemes	Blaze[6]	Ivan[39]	Ateniese[9]	Canetti[42]	Libert [50]	Yau[62]
Collusion safe	No	No	Yes	No	Yes	No
Non-interactive	No	Yes	Yes	No	Yes	Yes
Unidirectional	No	Yes	Yes	No	Yes	No
Key optimal	Yes	No	Yes	Yes	Yes	Yes
Searchable	No	No	No	No	No	Yes
Verifiable	No	No	No	Yes	Yes	No
CCA secure	No	No	No	Yes	No	No

[42], Libert and Vergnaud [50] present another scheme using a strongly unforgeable one-time signature to tie several ciphertext components altogether and offer a safeguard against CCA.

Boneh et al. [51] first proposed a public key encryption with keyword search (PEKS) scheme enable searching of keywords within encrypted messages. After that, a number of solution have been proposed [52, 53, 54, 55, 56, 57, 58, 59]. Searchable encryption scheme provides a way to search on encrypted datas without leaking any information to the untrusted server[60]. Shao et al. [61] proposed Searchable PRE (PRES) scheme, which encrypts the message and keyword in the same encryption algorithm, while Yau et al., [62] introduced a PRES scheme which keeps the encryption of message and encryption of keyword separate so that we can have the flexibility to select which standard PRE and PRES schemes to be sued for satisfying the requirements of the actual applications.

2.4. PRIVACY-PRESERVING AUTHENTICATION

In this section, we first present the work most related to ours. Then we will provide an overview of other privacy-preserving authentication protocols in mobile ad-hoc network (MANET) and vehicular ad-hoc network (VANET).

(1) The Camenisch Anonymous Credential System. Camenisch et al.[63, 64] introduced the first anonymous credential system using zero-knowledge proof to achieve anonymous authentication. This scheme guarantees user anonymity but

has a credential sharing problem where dishonest users can share their credentials with others, and it is insecure against the replay attack. To solve the credential sharing problem, hardware based solutions have been proposed in [65, 66, 67]. In our proposed approach, we do not assume a user's computing device is equipped with specialized hardware The credential revocation problem was addressed in [68], but the solution only works when the system is adopted as a regular credential system (without randomizing a user's credential for each authentication). There still does not exist a concrete solution to the credential sharing problem in the actual anonymous credential system. There are other extensions to the Camenisch scheme [69, 70], but none of them directly addresses the disadvantages listed in Table 7.1. Due to its popularity and similar setting, the Camenisch scheme will be used as the baseline to evaluate the performance of our proposed randomized authentication (RAU) system.

As listed in Table 7.1, the proposed anonymous credential system has two disadvantages comparing to AC. First, in addition to the service providers, the authentication servers also need to be online to process user authentication request, and multiple messages are communicated among the participating parties. At the first glance, the efficiency of the proposed system may not be as good as AC. However, in Section 7.4, we will empirically show the proposed protocols are very efficient (e.g., a little bit more efficient than AC). Secondly, the proposed system assumes the existence of two mutually independent authentication servers and the two servers do not collude. We believe that collusion will not happen in practice if the two servers are legitimate IT companies.

(2) Privacy-Preserving Authentications in VANET. We can categorize the existing work on privacy-preserving authentication in VANET into two major categories: (i) pseudonym-based and (ii) group-based protocols.

The general goal of the pseudonym-based authentication protocols is to enable vehicles to use different pseudonyms during communication rather than using their real identities. One of the earliest work in this category is by Raya and Hubaux [71]. They suggested that when a vehicle needs to sign a message, it randomly selects a private key from a huge pool of certificates issued by the authority. The message receiver will verify the sender's signature by checking the validity of the corresponding public key certificate. The problem of this protocol is that vehicles need to check a long list of revoked certificates when verifying each received signed-message, which is very time consuming.

Raya, et al. in [72] proposed efficient revocation schemes. However, these schemes violate the location privacy requirement and are subject to a movement tracking attack [73]. In order to reduce the average overhead of message authentication, Calandriello et al. [74] proposed a hybrid scheme, which is also computationally expensive because it needs to check if the group signature is from a revoked vehicle [75]. Other pseudonym-based protocols can be found in s[76, 77, 78, 79, 80, 81, 82], achieving different degrees of improvement over the key revocation problem. However, in most these protocols, the identity management authority is required to maintain the certificates associated with each vehicle so as to retrieve the vehicles' real identities when disputes occur. This allows the authority to track the vehicles' movement; hence, the vehicles' privacy is not fully preserved.

Another category of privacy preserving authentication protocols is group-based[82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93]. The typical idea is to utilize group managers to group and authenticate vehicles, which enables vehicles to anonymously communicate with group members. In general, existing group-based protocols may have certain disadvantages: First, the group manager has all the knowledge about group members and hence is able to track them. In our protocol, this becomes harder since the servers in our scheme need to collude. Secondly, group managers are difficult to select because they serve as trusted parties. There are no theoretical results or comprehensive empirical studies on how to select a trustworthy group leader in VANET. In addition, Groups are also dynamic, and group managers can leave the group at any moment. New group leaders will know the private information within the group. The more dynamic the group becomes, the more private information can be leaked from the group. As a result, we do not adopt the grouped based approach in this paper.

(3) Privacy-Preserving Authentications in MANET. Most related work in MANET is associated with authenticating the messages exchanged in the network without disclosing the actual identities of the source and the destination. Ciszkowski and Koutulski [94] provided an ANAP protocol which identifies the destination using the hash value of a user's pseudonymous. However, the problem of this scheme is how the source of a transmission can get the pseudonymous of the destination node. By assuming that such pseudonymous are public, attackers can pre-compute a table containing pairs of pseudonymous-hashes. In this way, when a packet is captured in the network, a destination node can be immediately discovered. On the other hand, when the pseudonymous is secret, then using hashes does not provide enough strong security [95].

Chou et al. [96] proposed an efficient anonymous communication protocol for peer-to-peer applications over MANETs which uses broadcast-based scheme and probability flooding control to establish multiple anonymous paths within a single query phase. The scheme uses controlled and probabilistic broadcasting to provide anonymity while avoids using step-by-step encryption/decryption and achieves lower computational complexity; however, this approach does not work for privacy-preserving user authentication.

Freudiger et al. [97] pointed out a self-organized anonymous message authentication protocol that a user can use a group of identities including his own to generate a ring signature and the successful verification reveals the only fact that the signature is generated by one of the group identities to authenticate the messages sent from a particular group. Similarly, Ren and Harn [98] proposed a (t, n)-threshold ring signature scheme to achieve anonymous authentication for communication by verifying the ring signature. As other privacy-preserving message authentication protocols, the above schemes cannot be used to anonymously authenticate the identity of a user.

Tsai et al. [99] proposes a secure anonymous authentication protocol to achieve user unlinkability under mobile wireless environment. In the authentication phase, after receiving the credential (certificate) from user, FA (Foreign Agent) forwards the credential with his signature to HA (Home Agent) who issued credentials to users in the initial phase. HA will check the validity of the credential by searching the mapping table and send back the acknowledgment message when the credential is correct. However, in such scheme, the HA can learn useful information to track a user when the FA sends authenticating information to him.

Kotzanikolaou et al. [100] presented an efficient anonymous authentication scheme that provides untraceability and unlinkability of mobile devices while accessing location-based services. The scheme uses standard primitives such as zero-knowledge proofs, MACs and challenge/response. However, there are a couple of drawbacks: First, when a user U generates n different credentials, if each user possesses the same n, which would cause information leakage. If each user possesses different n, they could be tracked by an issuer or SP (Service Provider) in the verification process. Secondly, if more service providers join the network, not only more storage space and secret keys shared between the issuer and each service provider are needed, but also the number of communication messages increase exponentially. Thus, the scheme is not practical.

There are other content-based authentication schemes utilize attribute encryption. For instance, Baden et al. [101] developed Persona which achieves privacy by encrypting private content and prevents misuse of a user's data through authentication under online social network. However, when a user authenticates another user or a group, the users and group need to belong to a certain category, such as “family”, “friend”, and the group size needs to be almost fixed. Attribute encryption is very expensive when there are a large number of users. In a dynamic mobile network such as MANET and VANET, both information and users constantly change, it is impractical or even impossible to apply attribute encryption in this problem domain. In addition, messages associated with location based services like providing traffic flow information are generally not confidential, so to save computation and storage costs, we should not encrypt these information under most situations.

3. CRYPTOGRAPHIC PRIMITIVES AND SECURITY DEFINITIONS

3.1. SECURE MULTIPARTY COMPUTATION (SMC)

Consider a scenario where n number of parties, $P_1, \ldots, P_n$, wish to securely compute a function $f(a_1, \ldots, a_n)$ of their respective inputs, while preserving privacy of their inputs. In the literature, this is referred to as secure multiparty computation (SMC). In a formal defination, SMC allows the evaluation of the function $(b_1, \ldots, b_n) = f(a_1, \ldots, a_n)$ such that the output b_i is known to P_i, making sure the following two conditions hold as well:

- Correctness- the correct value of output is computed
- Privacy - the output is the only new information that is released

Privacy/security is closely related to the amount of inofmation disclosed during the execution of a protocol. In our proposed protocols, our goal is to ensure no information leakage to the participating parties other than what they can deduce from their own inputs and outputs. There are many ways to define information disclosure. To maximize privacy or minimize information disclosure, we adopt the security definitions of secure multiparty computation (SMC) first introduced by Yao's Millionaires problem for which a provably secure solution was developed [102, 103]. Goldreich et al. [104] extended this to multiparty computation scenario and proved that any compuation which can be done in polynomial time by a single party can also be done securely by multiple parties. Since then much work has been done for the multiparty case [105, 106, 107, 108, 109, 110].

Generally, there are three types of adversarial model under SMC: semi-honest, covert and malicious. An adversarial model specifies what an adversary is allowing to do during the execution of secure distributed protocols. In semi-honest model, an adversary follows the prescribed procedures of a protocol, but they are free to perform any computation based on their own inputs, outputs and the messages received during the execution of the protocol. As a result, whatever can be inferred from the private input and output of an attacker is not considered as a privacy violation. In other words, the adversary under semi-honest model is considered as a passive attacker while an adversary under malicious model can be treated as an active attacker who can diverge from the normal execution of a protocol arbitrarily. The covert adversarial model [111] lies between the semi-honest and malicious models. Specifically, an adversary under the covert adversarial model may diverge arbitrarily from the execution of a protocol, however, a honest party is guaranteed to detect such cheating with good probability.

In this thesis, to develop secure and efficient protocols, we assume that parties are semi-honest or passive. That is, the parties follow the prescribed procedures of a protocol, but they are free to perform any computation based on their own inputs, outputs and the messages received during the execution of the protocol. As a result, whatever can be inferred from the private input and output of an attacker is not considered as a privacy violation. The complexities of the existing SBD protocols given in Table 1.1 are all under the semi-honest model. In particular, the most efficient SBD protocol [4] among the existing ones was proved to be statistically secure under the semi-honest model. Therefore, our assumption is consistent with the work we are planning to compare. To prove the security of the proposed protocol under the semi-honest model, we will adopt the formal security definition given below [112]:

Definition 1 *Let T_i be the input of party i, $\prod_i(\pi)$ be i's execution image of the protocol π and s be the result computed from π. π is secure if $\prod_i(\pi)$ can be simulated from $\langle T_i, s\rangle$ and distribution of the simulated image is computationally indistinguishable from $\prod_i(\pi)$.*

In the above definition, an execution image generally includes the input, the output and the messages communicated during an execution of a protocol. To prove a protocol is secure, we generally need to show that the execution image of a protocol does not leak any information regarding the private inputs of participating parties [112].

3.2. SECRET SHARING SCHEMES (SSS)

Given secret $x \in Z_k$, its shares are distributed to each party as follows: for party P_j ($j \in \{1, \dots n-1\}$), uniformly generate values $x_j \in Z_k$, and distribute x_j to P_j and $x_n = (x - \sum_{j=1}^{n-1} x_j) \bmod k$ to P_n. We denote the distribution protocol as $(\langle P_j, x_j^{P_j} \rangle) \leftarrow ShareGen(x, k)$. The secret x also can be reconstructed by adding each party's share together, such as $x = \sum_{j=1}^{n} x_j \bmod k$. We assume that the secret sharing scheme, like Shamir's secret sharing scheme, allows to compute sharings addition without communication. Suppose there are two secrets $[a]_k$, $[b]_k$ and a public constant c. Each party P_j ($j \in \{1, \dots, n\}$) has its own shares $[a]_k^{P_j}$ and $[b]_k^{P_j}$ of a and b respectively. Then the parties can compute: $[a+b \bmod k]_k^{P_j}$ and $[ca \bmod k]_k^{P_j}$, easily. Since the secret sharing scheme is homomorphic considering addition and multiplication by scalar, thus these operations can be done in only one round without communication among parties.

3.2.1. Two Shares Multiplication. We implement the two shares multiplication protocol by modifying a little bit Du-Atallah multiplication protocol[113, 114]. Instead of outputing three shares, we only generate two shares output even involving three parties to compute. The implementation process is shown in Algorithm 2: P_1 and P_2 possess a and b, respectively. Initially, P_3 generates α, β, $\gamma \in Z_N$ and sends α to P_1, β, γ to P_2. Next, P_3 computes $c_1 = \alpha * \beta - \gamma$, and sends c_1 to P_1. Then, P_1 and P_2 communicate with each other and exchange data in a secure channel: P_1 sends $(a + \alpha)$ to P_2 and P_2 sends $(b + \beta)$ to P_1. Now, with all these information, P_1 and P_2 are able to compute their shares $[ab]_N^{P_1}$ and $[ab]_N^{P_2}$ of ab locally, where $[ab]_N^{P_1} = -\alpha b - \gamma \bmod N$ and $[ab]_N^{P_2} = b * (a + \alpha) + \gamma \bmod N$.

Algorithm 2 $(\langle P_1, [s]_N^{P_1}\rangle, \langle P_2, [s]_N^{P_2}\rangle \leftarrow \text{TwoSharesMult}(\langle P_1, a\rangle, \langle P_2, b\rangle)$

```
Input: $a$ from $P_1$ and $b$ from $P_2$.
Output: $[s]_N^{P_1}$ to $P_1$ and $[s]_N^{P_2}$ to$P_2$, where $([s]_N^{P_1} + [s]_N^{P_2}) \bmod N = a \cdot b \bmod N$
    (1). $P_3$ generates three randome numbers: $\alpha$, $\beta$ and $\gamma$, where $\alpha, \beta, \gamma \in Z_N$, and computes:
         $c1 = \alpha * \beta - \gamma$ and $c2 = \gamma$, then sends $\alpha$, $c1$ to $P_1$, and $\beta$, $c2$ to Party $P_2$.
    (2). $P_1$ computes: $(a + \alpha)$, and sends it to Party $P_2$.
    (3). $P_2$ computes: $(b + \beta)$, and sends it to Party $P_1$.
    (4). $P_1$ computes: $[s]_N^{P_1} = -\alpha * (b + \beta) + c1 \bmod N$.
    (5). $P_2$ computes: $[s]_N^{P_2} = b * (a + \alpha) + c2 \bmod N$.
```

3.2.2. Secure Multiplication. Multiplication between two secretly shared values cannot be done locally, and it requires participating parties to collaboratively perform a sequence of operations. For example, given $[a]_N$ and $[b]_N$, we want to compute $[ab]_N$. Suppose there are three parties P_1, P_2 and P_3, and assume a, b and ab are secretly shared between P_1 and P_2, while P_3 is only required to perform certain computations. More specifically, P_1 has $[a]_N^{P_1}$ and $[b]_N^{P_1}$. P_2 has $[a]_N^{P_2}$ and $[b]_N^{P_2}$, where $a = ([a]_N^{P_1} + [a]_N^{P_2}) \bmod N$ and $b = ([b]_N^{P_1} + [b]_N^{P_2}) \bmod N$. Given the above inputs, the parties can compute $[ab]_N$ according to the following equation:

$$\begin{aligned} [ab]_N &= ([a]_N^{P_1} + [a]_N^{P_2})([b]_N^{P_1} + [b]_N^{P_2}) \bmod N \\ &= [a]_N^{P_1} \cdot [b]_N^{P_1} + [a]_N^{P_2} \cdot [b]_N^{P_1} + [a]_N^{P_1} \cdot [b]_N^{P_2} + [a]_N^{P_2} \cdot [b]_N^{P_2} \bmod N \end{aligned}$$

The $[a]_N^{P_1} \cdot [b]_N^{P_1}$ and $[a]_N^{P_2} \cdot [b]_N^{P_2}$ values are computed by P_1 and P_2 locally without incurring any communication costs. The $[a]_N^{P_2} \cdot [b]_N^{P_1}$ and $[a]_N^{P_1} \cdot [b]_N^{P_2}$ can be securely computed using a variation of the Du-Atallah's multiplication protocol [113, 114]. Due to space limitation, we omit the detailed steps here. Following the existing work, we count the entire secure multiplication protocol as one multiplication in the complexity analysis of the proposed protocol.

3.3. PAILLIER CRYPTOSYSTEM

Paillier's public key encryption system was first intorduced in [115]. In the scheme, the public key $N = p * q$, where p and q are large primes with similar size, and they are private information. $\lambda = lcm(p-1, q-1)$, $\mu = (L(g^{\lambda} \bmod n^2))^{-1}$ where function $L(.)$ is defined as $L(u) = \frac{u-1}{N}$. In general, the size of N should be at least 1024 bits. Since p and q have equal size, we assume $g = N+1$, $\lambda = \varphi(N)$ and $\mu = \varphi(N)^{-1} \bmod N$, where $\varphi(N) = (p-1)(q-1)$. The public key is (N, g) and private key is (λ, μ). The encryption function for message $m \in Z_N$ is defined as follows:

$$\begin{aligned} C = E(m,r) &= g^m \cdot r^N \bmod N^2 \\ &= (1+N)^m \cdot r^N \bmod n^2 \end{aligned} \tag{3.1}$$

where r is randomly selected from $Z^*_{N^2}$. Note that the encryption is only based on the public key, and the group $Z^*_{N^2}$ which contains the elements from $Z^*_{N^2} = \{0, 1, 2, \ldots, N^2 - 1\}$ which are co-prime to N^2. The decryption function is:

$$\begin{aligned} m &= D(C) = D(E(m,r)) \\ &= \frac{(E(m,r)^{\lambda} \bmod N^2) - 1}{N} \cdot \mu \bmod N \end{aligned} \tag{3.2}$$

Since r is randomly selected from $Z^*_{N^2}$ each time a message is encrypted, $E(m, r_1) \neq E(m, r_2)$ if $r_1 \neq r_2$. On the other hand, $D(E(m, r_1)) = D(E(m, r_2)) = m$ regardless the value of r_1 and r_2.

(1) Additive Homomorphic Properties. Let the E and D be the encryption and decryption functions in Paillier public encryption system with public key pk and privated key sk. Without sk, no one can dicover m from $E_{pk}(m)$ in polynomial

time. When the context is clear, we will omit pk and sk from the notations of the encryption and decryption functions. The Paillier encryption system has the following properties[116]:

- The encryption function is additive homomorphic in that the product of the encryptions of m_1 and m_2 produces the encryption of $m_1 + m_2$.

$$E(m_1) * E(m_2) = E(m_1 + m_2) \tag{3.3}$$

- Given a constant c and $E(m)$:

$$E(m)^c = E(c * m) \tag{3.4}$$

- The encryption function has semantic security as defined in [117], i.e., a set of ciphertexts do not provide additional infromation about the plaintext to an adversary. E.g., suppose that C_1 and C_2 are the ciphertexts generated by performing the encryptions of m at different time using the same public key, then there is very high probability that $C_1 \neq C_2$, but $D(C_1) = D(C_2)$ holds.

The encryption scheme is semantically secure [117, 118], i.e., given a set of ciphertexts, an adversary cannot deduce any information about the plaintext. In Proxy re-encryption, we assume that a data owner encryted datas using Paillier public key encryption scheme before outsourcing them to remote storage.

(2) Mathematical Decryption. If a user Bob knows the ciphertext $C = E(m, r)$ and corresponding r which can be generated with Random Number Seed (RNS), Bob can compute $(r^N)^{-1}$ and then retrieve the plaintext message m in such way:

$$m = D'(C) = \frac{(C \cdot (r^N)^{-1} \bmod N^2) - 1}{N} \tag{3.5}$$

The correctness of Eq.(3.5) comes from the lemma 1. First, we have the following lemma. Note that $(1+N)^x \in Z^*_{N^2}$, since $gcd(1+N, N^2) = gcd(1+N, N) = 1$.

Lemma 1 *For any $x \in Z_N$, we have[119, 120]:*

$$(1+N)^x = 1 + xN \bmod N^2 \tag{3.6}$$

Proof 1 *Proof by induction over x. Basis $x = 0$ is obviously true.*
Inductive step: $x \to x+1$:

$$\begin{aligned}(1+N)^x &= 1 + xN \bmod N^2 \Rightarrow \\ (1+N)^{x+1} &= (1+xN)(1+N) \bmod N^2 \\ &= 1 + xN + N + xN^2 \\ &= 1 + (x+1)N \bmod N^2\end{aligned} \tag{3.7}$$

We assume $g = N+1$, then Eq. 3.1 becomes $C = E(m,r) = (1+N)^m \cdot r^N \bmod N^2$. With Lemma 1, we can compute:

$$\begin{aligned}&(C \cdot (r^N)^{-1}) \bmod N^2 \\ &\quad = ((1+N)^m \cdot r^N \cdot (r^N)^{-1}) \bmod N^2 \\ &\quad = (1+Nm) \bmod N^2.\end{aligned} \tag{3.8}$$

Then, m can be easily retrieved. Even though Bob knows what the plaintext message is, he still learn nothing about the secret key (λ, μ) whose privacy is preserved (master key security introduced in [8, 9]).

3.4. STRONGLY UNFORGEABLE SIGNATURE

A signature scheme is secure if it is unforgeable under chosen plaintext attack, that is, an adversary should not be able to compute a signature for a new message given a signature for a few messages of his choice. While the strongly unforgeable signature scheme ensures the adversary cannot compute a new signature for a previously signed message[121, 122, 123, 124]. In our scheme, we assume there is a strongly unforgeable one-time signature scheme $Sig = (\zeta, S, V)$, where ζ is a key generation protocol, S is a message signing protocol while V is a signature verifying protocol. On input of a security parameter k, ζ generates a one-time key pair (sk, vk). For any message m, $V(\sigma, vk, m)$ outputs 1 whenever $\sigma = S(sk, m)$ and 0, otherwise. The security of Sig is defined as playing the following game:

- Setup. The challenger runs ζ, and gives the resulting verification key vk to the adversary and keeps the signing key sk in safe.
- Queries. The adversary continuely issues signature queries $m_1, \ldots, m_w$. To each query m_i, the challenger responds by running S and gives the resulting signature σ_i of m_i to the adversary. The adversary may queries adaptively so that the query m_i depends on the replies to $m_1, \ldots, m_{i-1}$.
- Output. The adversary outputs (m, σ)

The adversary wins the game if σ is a valid signature of m according to V and (m, σ) is not among the pairs (m_i, σ_i). Then, we have the following definition of advantage that the adversary gains:

Definition 2 *$Sig = (\zeta, S, V)$ is a storngly unforgeable one-time signature if the advantage of any PPT adversary Λ:*

$$\begin{aligned}
Adv_\Lambda = |Pr[&(sk, vk) \leftarrow \zeta(k); \\
&(m, st) \leftarrow \Lambda(vk); \\
&\sigma \leftarrow S(sk, m); \\
&(m', \sigma') \leftarrow \Lambda(m, \sigma, vk, st) : \\
&V(\sigma', vk, m') = 1 \wedge (m', \sigma') \neq (m, \sigma)] - \frac{1}{2}| < negl(k)
\end{aligned} \tag{3.9}$$

3.5. COSINE SIMILARITY

(1) Secure multiplication (MULT). This protocol considers P_2 with input $E_{pk}(a)$, $E_{pk}(b)$ and outputs $E_{pk}(a * b)$ to P_2, where a and b are not known to P_1 and P_2. During this process, no information regarding a and b is revealed to P_1 and P_2. The output $E_{pk}(a * b)$ is known only to P_1.

This $MULT$ protocol can be implemented in several ways either using additive homomorphic encryption scheme or garbled circuit evaluation [125]. In each way, P_2 first needs to randomize $E_{pk}(a)$ and $E_{pk}(b)$ to get $E_{pk}(a + r_1)$ and $E_{pk}(b{+}r2)$ and sends the randomized values to P_1. After decryption, P_1 obtains $a+r_1$ and $b+r_2$. Now P_1 can compute $a*b$ is such way if using additive homomorphic encryption:

$$a * b = (a + r_1) * (b + r_2) - a * r_2 - b * r_1 - r_1 * r_2 \tag{3.10}$$

P_1 computes $E_{pk}((a + r_1) * (b + r_2))$ and sends it to P_2. With Eq.(3.10) and the additive homomorphic property of paillier encryption, P_2 can obtain $E_{pk}(a * b)$ at the end.

(2) Secure cosine similarity (SCS). With P_2's input $(E_{pk}(X), E_{pk}(Y))$, the protocol outputs the ciphertext of similarity $E_{pk}(cos(X, Y))$ to P_2. Here X and Y are both l dimensional vectors: $E_{pk}(X) = \langle E_{pk}(x_1), \ldots, E_{pk}(x_l)\rangle$ and $E_{pk}(Y) = \langle E_{pk}(y_1), \ldots, E_{pk}(y_l)\rangle$, where X, Y are normalized vectors, $cos(X, Y) = x_1 \cdot y_1 + \ldots, +x_l \cdot y_l$. First, P_2 computes the component wise multiplication: $E_{pk}(x_i * y_i) \leftarrow MULT(E_{pk}(x_i), E_{pk}(y_i))$, for $i \in \{1, l\}$, and obtains $E_{pk}(cos(X, Y)) \leftarrow \prod_{i=1}^{l} E_{pk}(x_i * y_i)$. Then P_2 communicates with P_1 to find out the highest similarity value and response with requested ciphertext or nothing (by comparing the value with threshold predefined)

Table 3.1: Common Notations

Notation	Meaning
(pk_i, sk_i)	A pair of public key and secret key of user indexed by i
r	Random number
RNS	Random number seed that can be used to compute random number
D	Data space
m	Data from data space D
C_m	Ciphertext of data m
C_A	Ciphertext under A's public key
(sk, vk)	A pair of signing key and verification key
(g_i, N_i)	Public parameters of Paillier PKE scheme indexed by i

3.6. PROXY RE-ENCRYPTION (PRE)

A PRE scheme allows the proxy to transform ciphertext of message m under one public key to another ciphertext of the same m under different public key, with the proxy leaning nothing. For simplicity, in the following context, we use A, B and P to refer Alice, Bob and Proxy in the notations, respectively, as the default setting unless point out, e.g., pk_A denotes Alice's public key. The notations in table 3.1 are used extensively: First, we give the input-output specifications for bidirectional PRE as follows[6].

Definition 3 *A bidirectional PRE is a tuple of algorithms (KeyGen, ReKeyGen, Enc, ReEnc, Dec).*

- *KeyGen* $(1^k) \rightarrow (pk_i, sk_i)$*. On input the security parameter* 1^k*, the algorithm outputs the key pair* (pk_i, sk_i)*, where* $i \in \{A, B\}$*.*
- *ReKeyGen* $(sk_A, sk_B) \rightarrow rk_{A\rightarrow B}$*. A and B input secret keys* sk_A *and* sk_B*, respectively. The algorithm outputs a re-encryptionl key* $rk_{A\rightarrow B}$ *to the P.*
- *Enc* $(pk_A, m) \rightarrow C$*. A inputs a public key* pk_A *and a message* $m \in \{0, 1\}^*$*, the algorithm outputs* C*.*

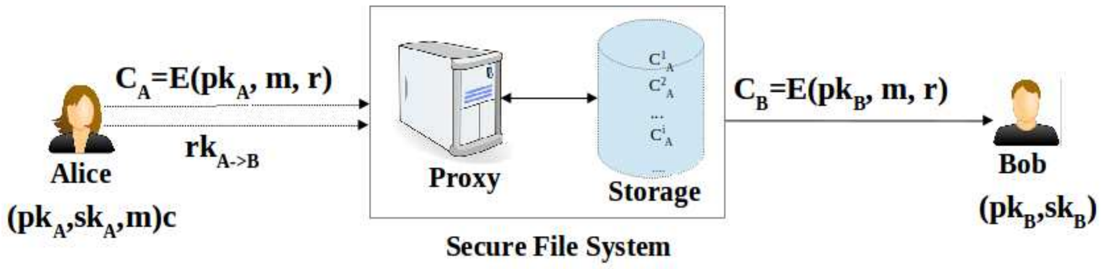


Figure 3.1: Overview of PRE scheme

- *ReEnc* $(rk_{A\to B}, C) \to C'$. *Proxy inputs a re-encryption key* $rk_{A\to B}$ *and a ciphertext* C*, the algorithm outputs a new ciphertext* C' *or the empty symbol* $\perp$ *if* C *is invalid.*

- *Dec* $(sk_B, C') \to m$. B *inputs a secret key* sk_B *and a ciphertext* C'*, the algorithm outputs a message* $m \in \{0,1\}^*$ *or the empty symbol* $\perp$ *if* C' *is invalid.*

The above definition is *bidirectional* in that the re-encryption key to transform ciphertexts from Alice to Bob can also be used to transform ciphertexts from Bob to Alice. We assume that Proxy learns the re-encryption key as defined but nothing else and the secret key holders also learn nothing in ReKeyGen which involves the participation of parties. The I/O specification for *unidirectional* PRE would be changed for ReKeyGen algorithm. Instead of inputing two secret keys, Unidirectional PRE only takes a secret key sk_A and a public key pk_B as input, which makes the scheme be *noninteractive* (properties are explained in detail in the following section). The overview of PRE is shown in figure 3.1

In the paper, we construct Unidirectional PRE with different algorighms and our constructions do not provide ReKeyGen and ReEnc algorithms. We use paillier public key encryption to implement secure U-PRE with simple design. Instead of providing re-encryption key to proxy, Alice just gives the random seed, generating random number used in encryption, to proxy. When Bob requests for it, proxy sends that to him. Although Bob knows the seed, the secret key of Alice is kept safely offline where it is less vulnerable to compromise.

(1) Secure PRE Features. The above definition of PRE scheme, assuming the proxy is semi-honest, which is not secure enough in applying to the distributed file storage system. For example, if the proxy is not satisfying semi-honest assumption, in both bidirectional and unidirectional schemes, the re-encrypted ciphertext may be modified without noticed by Bob; also in the worst case, Alice's secret key may be compromised when the proxy and dishonest Bob collude. Therefore, the varieties of collusion safe and against malicious proxy should be preserved when designing a secure PRE scheme. As we analyzed, to achieve the security of distributed file storage system, a secure PRE scheme should possess the following important properties:

- Collusion safe: By colluding, Bob and the Proxy cannot recover Alices Secret key. For example, in [39], the secret key can be reconstructed by $s = s1+s2$, where s is secret key of Alice and s_1, s_2 are the shares to proxy and Bob, respectively; in [6], the secret key also can be compromised by $a = rk_{A\to B} * b$, where re-encryption key $rk_{A\to B} = a/b$ and a, b are secret key of Alice and Bob, respectively. A secure PRE is able to avoid above situations happen.

- Non-interactive: Re-encryption key $rk_{A\to B}$ can be generated by Alice combining Bob's public key; no trusted third party or interaction is required, that is, users do not need to communicate or reveal their secret keys in order to join the system, which allows content owners to add users to the system without interaction, simply by obtaining their public key.

- Unidirectional: The re-encryption is one-way, that is, the ciphertext of m under Alice's public key can be re-encrypted to the ciphertext of the same m under Bob's public key using re-encryption key $rk_{A\to B}$, but the ciphertext

m' under Bob's public key cannot be re-encrypted to the ciphertext of the same m'under Alice's key using $rk_{A\to B}$.

- Key optimal: The size of Bob's storage for the secret key remains constant, regardless of how many delegations he accepts.

- Searchable**: The ciphertexts from Alice stored in the storage can be searched by Bob (e.g., by keywords) without allowing the proxy learning the query and the query result.

- Non-transitive: The proxy, alone, cannot redelegate decryption rights. For example, given $rk_{A\to B}$ and $rk_{B\to L}$, he cannot produce $rk_{A\to L}$

- Verifiable: Bob is able to verify the validity of re-encrypted ciphertexts to against the malicious proxy.

If the above security property being achieved, the file system is secure even if the following properties are not preserved:

- Non-multihop: Single-hop implys a re-encrypted ciphertext cannot be further re-encrypted[9]; while multi-hop means a ciphertext can be re-encrypted from Alice to Bob to Louis and so on[6]. For example, in a single-hop scheme, ciphertext C_A of message m under Alice's public key can be re-encrypted using $rk_{A\to B}$ to C_B of the same m, but C_B cannot be further re-encrypted.

- Proxy invisibility: This is a feature offered by the original BBS scheme. The proxy in the BBS scheme is transparent in the sense that neither the sender of an encrypted message nor any of the delegatees have to be aware of the existence of the proxy.

- Temporary: [39] suggested applying generic key-insulation techniques to their constructions to form schemes where Bob is only able to decrypt messages from Alice that were authorized during some specific time period i.

**: indicates the feature is preserved under different assumptions

- Original access: Alice can decrypt the re-encrypted ciphertexts sent to Bob.

(2) Adverary Model. We categorized adversaries into three models:

- *Semi-honest proxy model*:

  As stated in the existing papers [6, 39, 61, 126, 127, 128, 129], the proxy is assumed to be semi-honest. That is, the proxy follows the prescribed procedures of a protocol, but he/she is free to perform any computation based on his/her own inputs, outputs and the messages received during the execution of the protocol. As a result, whatever can be inferred from the private input and output of an attacker is not considered as a privacy violation. .

- *Malicious proxy model*

  1. M1: The proxy is malicious with ability of colluding with the dishonest user [8, 9, 42, 46, 48, 49, 62]. A dishonest user, colludes with a malicious proxy, may want to compromise the target's secret key. For example, in [39], Alice's secret key can be reconstructed if the proxy and Bob collude.

  2. M2: The proxy is malicious allowing to modify ciphertext [42, 43, 50] and show an incorrect output. According to the definition of PRE scheme, the proxy learn nothing about the secret information based on the knowledge of re-encryption keys. If being malicious, a proxy may compute the re-encryption incorrectly. For example, in [9], Bob has no way of verifying the correctness of re-encrypted ciphertexts.

  3. M3: The proxy is assumed to behave willingly. For example, the proxy may refuse to provide services of re-encryption, or serve against the protocols.

3.7. DIFFIE-HELLMAN KEY EXCHANGE

In this paper, we will use Diffie-Hellman key exchange protocol[130] to generate a random number shared by two parties Alice and Bob. Let p be a large prime number and g be a primitive root in $\mathbb{Z}_p$. p and g are known by both parties. To share a random number k between Alice and Bob, they can perform the following operations: Alice and Bob generate x_1 and x_2 in $\{1, 2, \ldots, p-2\}$ respectively. Alice sends $g^{x_1} \mod p$ to Bob, and Bob sends $g^{x_2} \mod p$ to Alice. At the end, both parties can compute $k = g^{x_1 x_2} \mod p$. As long as one party follows the protocol, k will be a random number in the group of $\mathbb{Z}_p$.

4. SECURE COMPARISON

4.1. MAIN CONTRIBUTION

In this section, we first present two new secure comparison protocols that are needed for the proposed SBD protocols. Note that the secure comparison protocols presented here are specifically designed to build a more efficient SBD. Thus, their input parameters and structures are very different from a regular secure comparison protocol. In this scheme, the inputs to be compared $[a]_{N_1,B}$ and $[b]_{N_1,B}$ ($l = |a|$ or $|b|$) are given as bitwise sharings $[a_{l-1}]_{N_1}, \ldots, [a_0]_{N_1}$ and $[b_{l-1}]N_1, \ldots, [b_0]_{N_1}$, with $a = \sum_{i=0}^{l-1} a_i 2^i$, $b = \sum_{i=0}^{l-1} b_i 2^i$; the outputs is $[a \overset{?}{>} b]$. Both protocols provide an efficient way to compare two sharing secrets with low computation and communication complexity in constant rounds. Furthermore, we apply our proposed protocols to solving the bit-decomposition problem.

4.2. CUSTOMIZED SECURE COMPARISON PROTOCOLS FOR SBD

We assume P_1 and P_2 have secret shares of a and b in $\mathbb{Z}_{N_1}$, and bit-wise secret shares of a and b in $\mathbb{Z}_{N_2}$. $[a]_{N_2,B} = ([a_{l-1}]_{N_2}, \ldots, [a_0]_{N_2})$ and $[b]_{N_2,B} = ([b_{l-1}]_{N_2}, \ldots, [b_0]_{N_2})$, where $[a]_{N_2,B}, [b]_{N_2,B}$ are by construction the bit-wise sharing of $[a]_{N_2}$ and $[b]_{N_2}$, respectively. where $0 \leq a, b < 2^l$, $l = size(N_1)$, and $N_2 = \lceil 2 \log N_1 \rceil$. P_1 and P_2 want to obtain the secret shares of the comparison result between a and b. The novelty of our secure comparison is that certain computations can be performed on a much smaller domain which leads to a more efficient protocol. The proposed comparison scheme has three main stages:

- Stage 1 - Compute the Difference Vector:
  Given $[a]_{N_2,B}$ and $[b]_{N_2,B}$, compute the secret shares of the difference vector (i.e., the bit-wise xor of a and b, denoted by $e = a \oplus b$). At the end of this

stage, each party obtains new bit-wise shares $[e]_{N_2,B}$ of e. Other than these shares, nothing else is revealed.

- Stage 2 - Find the First Different Bit:
  Find the most significant bit location i such that $a_i \neq b_i$. In this stage, the computation each party performs can be done locally, such that no secret information is revealed.

- Stage 3 - Adjusting the Result:
  The parties will adjust the temporary comparison result based on the flipping bit (sequence).

4.2.1. FAST-CMP. The proposed secure comparison protocol is termed as FAST-CMP whose main steps are highlighted in Algorithm 4. At the end of the protocol, P_1 and P_2 obtain the secret shares of $f = b \overset{?}{>} a$, for $f \in \{0, 1\}$, but not f. Next, we detail the steps involved in each stage.

Stage 1-Computing The Difference Vector. Given $[a]_{N_2,B}$ and $[b]_{N_2,B}$, the purpose of this stage is to compute the bit-wise secret shares of the xor vector e between a and b. Initially, P_1 and P_2 jointly execute the DIFF-VEC protocol which takes 1 round and l multiplications.

$$(\langle P_1, [e]^{P_1}_{N_2,B}\rangle, \langle P_2, [e]^{P_2}_{N_2,B}\rangle) \leftarrow \text{DIFF-VEC}(\langle P_1, [a]^{P_1}_{N_2,B}, [b]^{P_1}_{N_2,B}\rangle, \langle P_2, [a]^{P_2}_{N_2,B}, [b]^{P_2}_{N_2,B}\rangle, \langle P_3, \perp\rangle) \tag{4.1}$$

Example 1 *Assuming $N_1 = 31$ and $N_2 = 11$. Suppose P_1 and P_2 have bit-wise shares of $a_B = 10110$ and $b_B = 10100$, then the outputs of DIFF-VEC are bit-wise secret shares of $e_B = 00010$ to P_1 and P_2.*

Stage 2 - Finding the First Different Bit. Next, inspired by the existing work [10], we compute the γ sequence in a different way to prevent the possible information leaking. The parties compute the Δ sequence and then γ sequence

Algorithm 3 FIRST-DIFF-BIT

```
Input: $\langle P_1, [e]^{P_1}_{N_2,B}, \pi(.)\rangle$, $\langle P_2, [e]^{P_2}_{N_2,B}, \pi(.)\rangle$ and $\langle P_3, \perp\rangle$, where $\pi(.)$ is random permutation only
    known to $P_1$ and $P_2$
Output: $\langle P_3, i^*\rangle$
    1: $P_j$ $(j \in \{1,2\})$
        (a) Compute the $\Delta$ sequence:
            $[\Delta_{l-1}]^{P_j}_{N_2} = [e_{l-1}]^{P_j}_{N_2}$
            $[\Delta_i]^{P_j}_{N_2} = [\Delta_{i+1}]^{P_j}_{N_2} + [e_i]^{P_j}_{N_2}$, for $i = l-2, \dots, 0$ ($l$ is the bit-length of $e$)
        (b) Compute the $\gamma$ sequence:
            $[\gamma_{l-1}]^{P_j}_{N_2} = [\Delta_{l-1}]^{P_j}_{N_2}$
            $[\gamma_i]^{P_j}_{N_2} = [\gamma_{i+1}]^{P_j}_{N_2} + [\Delta_i]^{P_j}_{N_2}$, for $i = l-2, \dots, 0$
        (c) Compute $[u_i]^{P_j}_{N_2} = r_i[\gamma_i - 1]^{P_j}_{N_2}$, for $i = l-1, \dots, 0$ and $r_i \in_R Z^*_{N_2}$
        (d) $[v]^{P_j}_{N_2,B} \leftarrow \pi\left([u]^{P_j}_{N_2,B}\right)$
        (e) Send $[v]^{P_j}_{N_2,B}$ to $P_3$
    2: $P_3$
        (a) $v_i = [v_i]^{P_1}_{N_2} + [v_i]^{P_2}_{N_2} \bmod N_2$, for $i = l-1, \dots, 0$
        (b) Find the unique index $i^*$, where $v_{i^*} = 0$
```

locally which ensures that the value of all positions before d (d denotes the first different bit position start from MSB) is zero and that ofall positions after d is bigger than one. Then the computation $r_i[\gamma_i - 1]_{N_2}$ are locally performed, where r_i are uniformly random number in Z_{N_2}. Then, the value of position d is zero, and values of all the other positions are random number. Combine these together, we get $[u]_{N_2,B}$ sequence. One way to cover the position d is to peform permutation on $[u]_B$ using permutation function $\pi(.)$ first before sending to P_3 since we need P_3 to reconstruct the value of all positions. The detail is depicted in Algorithm 3. At the same time, P_1 uses the same permutation function $\pi(.)$ to permutate a_B, and use t to flip the sequence, where $t \in \{0,1\}$, producing $\tilde{a}_B$ and sends it to P_3. P_3 learns nothing about a_B from $\tilde{a}_B$ because the permutation function $\pi(.)$ and flipping bit are only know by P_1 and P_2. Then, P_3 reconstructs the sequence, finds the uniqe index i^* and sets temporary comparison result as $f = \tilde{a}_{i^*}$ and distributes shares to P_1 and P_2.

Algorithm 4 FAST-CMP

```
Input: $\langle P_1, [a]^{P_1}_{N_2,B}, [b]^{P_1}_{N_2,B}, t, \pi(.)\rangle$, $\langle P_2, [a]^{P_2}_{N_2,B}, [b]^{P_2}_{N_2,B}, b_B, t, \pi(.)\rangle$ and $\langle P_3, \perp\rangle$, where $\pi(.)$ is
    random permutation and $t$ is a random bit, known only to $P_1$ and $P_2$
Output: $\langle P_1, [u]^{P_1}_{N_1}\rangle$ and $\langle P_2, [u]^{P_2}_{N_1}\rangle$
    1: $P_2$:
        (a) $b'_B \leftarrow \pi(b_B)$
        (b) For $i = l-1, \ldots, 0$ do
                if $t = 0$, $\tilde{b}_i = b'_i$
                else, $\tilde{b}_i = 1 - b'_i$
            End for
        (c) Sends $\tilde{b}_B$ to $P_3$.
    2: $P_1$, $P_2$ and $P_3$:
        (a) $(\langle P_1, [e]^{P_1}_{N_2,B}\rangle, \langle P_2, [e]^{P_2}_{N_2,B}\rangle) \leftarrow$
            DIFF-VEC $(\langle P_1, [a]^{P_1}_{N_2,B}, [b]^{P_1}_{N_2,B}\rangle, \langle P_2, [a]^{P_2}_{N_2,B}, [b]^{P_2}_{N_2,B}\rangle, \langle P_3, \perp\rangle)$
        (b) $\langle P_3, i^*, f\rangle \leftarrow$FIRST-DIFF-BIT $(\langle P_1, [e]^{P_1}_{N_2,B}, \pi(.)\rangle, \langle P_2, [e]^{P_2}_{N_2,B}, \pi(.)\rangle, \langle P_3, \tilde{b}_B\rangle)$
    3: $P_3$:
        (a) $(\langle P_1, [f]^{P_1}_{N_1}\rangle, \langle P_2, [f]^{P_2}_{N_1}\rangle) \leftarrow ShareGen(f, N_1)$
    4: $P_j$ $(j \in \{1, 2\})$:
        (a). $[u]^{P_j}_{N_1} = (1-t)[f]^{P_j}_{N_1} + t[1-f]^{P_j}_{N_1} \bmod N_1$
```

The major goal of this stage is to find the first bit that is different from MSB to LSB between a_B and b_B. Due to flipping and permutation P_1 and P_2 performs before ahead, with temporary comparison result, the possibility that P_3 learns the final comparison result is no more than $\frac{1}{2}$.

Example 2 *Suppose we have P_1 and P_2's bit-wise shares of $e_B = 00010$, then the Algorithm 3 outputs the result as followings: first, computes the γ sequence: $\gamma_B = \{0, 0, 0, 1, 2\}$, then chooses $r_B = \{6, 4, 2, 7, 9\}$ randomly. At the end, P_1 and P_2 get the shares of small domain of permuted u_B sequence, $v_B = \{9, 0, 1, 7, 9\}$, using $\pi(reverse)$. P_3 reveals and finds the uniqe index i^*. As in Example 2, $i^* = 3$, then $f = \bar{a}_3 = 0$.*

Stage 3-Adjusting the result. P_1 and P_2 adjust their shares of comparison result according to the flipping bit t. P_1 and P_2 adjust their shares as $[1-f]^{P_1}_{N_1}$

and $[1-f]^{P_2}_{N_1}$, respectively, if $t = 1$.

In our application, P_3 keeps f for future use. Only P_3 knows the temporary comparison result (not final decision since the result may be flipped) which will be used in Algorithm 5 to perform selection. Knowing f, P_3 cannot learn the final comparison result because f is flipped by flipping bit t which is known by P_1 and P_2. At then end, P_1 and P_2 can adjust the final comparson result according to the value of t.

All steps involved are presented in Algorithm 4. This protocol can also be used for equality check problem such that all values revealed by P_3 are random numbers (Here, we assume the $a \neq b$). The complexity of comparison protocol is 1 round and l multiplications contributed by stage 1, all the other steps can be done locally.

The goal of Algorithm 5 is to select from $\tilde{c'}_B, \tilde{b}_B$ based on the value of f and flipping bit value t. The selection result is c_B in different domains (N_1 and N_2), $c_{N_1,B} = ([c]^{P_1}_{N_1,B} + [c]^{P_2}_{N_1,B}) \bmod N_1$, $c_{N_2,B} = ([c]^{P_1}_{N_2,B} + [c]^{P_2}_{N_2,B}) \bmod N_2$, and :

$$c_B = t\beta_B + (1-t)\alpha_B \tag{4.2}$$

where,

$$\begin{aligned} \alpha_B &= f\tilde{b}_B + (1-f)\tilde{c'}_B \\ \beta_B &= f\tilde{c'}_B + (1-f)\tilde{b}_B \end{aligned} \tag{4.3}$$

Example 3 *The purpose of Algorithm 5 is to determine what c is. Continued from Example 2, suppose P_1 has $c'_B = 110011$, $b_B = 10100$. In step 1, P_1 permutes and flips (since $t = 0$, there is no flipping) in such way: $\bar{c'}_B = \pi(c'_B) = 111001$ and $\bar{b}_B = \pi(b_B) = 00101$. In step 2, as in Example 1. P_1 and P_2 compute the difference vector; as in Example 2, P_3 obtans the temporary comparison result: $f = 0$. In step 3, P_3 decides how to output the right result. He would choose*

Algorithm 5 SELECT

```
Input: $\langle P_1, [r]^{P_1}_{N_2,B}, [x+r]^{P_1}_{N_2,B}, b, b', t, \pi(.)\rangle$, $\langle P_2, [r]^{P_2}_{N_2,B}, [x+r]^{P_2}_{N_2,B}, t, \pi(.)\rangle$, and $\langle P_3, \perp\rangle$, where $t$
    is random bit and $\pi(.)$ is random permutation known only to $P_1$ and $P_2$, and $b = x+r \bmod N_2$
    and $b' = b + N_2$
Output: $\langle P_1, [c]^{P_1}_{N_1,B}, [c]^{P_1}_{N_2,B}\rangle$ and $\langle P_2, [c]^{P_2}_{N_1,B}, [c]^{P_2}_{N_2,B}\rangle$, where $c = b$ if $b > r$ and $c = b'$ otherwise
 1: $P_1$:
      (a). $\tilde{b} \leftarrow \pi(b)$ and $\tilde{b}' \leftarrow \pi(b')$
      (b). If $t = 1$, then $\tilde{b}_i = 1 - \tilde{b}_i$ and $\tilde{b}'_i = 1 - \tilde{b}'_i$, for $0 \le i \le l-1$
      (c). Send $\tilde{b}$ and $\tilde{b}'$ to $P_3$
 2: $P_1$, $P_2$ and $P_3$:
      (a) $(\langle P_1, [e]^{P_1}_{N_2,B}\rangle, \langle P_2, [e]^{P_2}_{N_2,B}\rangle) \leftarrow$
          DIFF_VEC($\langle P_1, [r]^{P_1}_{N_2,B}, [x+r]^{P_1}_{N_2,B}\rangle, \langle P_2, [r]^{P_2}_{N_2,B}, [x+r]^{P_2}_{N_2,B}\rangle, \langle P_3, \perp\rangle$)
      (b) $\langle P_3, i^*\rangle \leftarrow$ FIRST_DIFF_BIT($\langle P_1, [e]^{P_1}_{N_2,B}, \pi(.)\rangle, \langle P_2, [e]^{P_2}_{N_2,B}, \pi(.)\rangle, \langle P_3, \perp\rangle$)
 3: $P_3$:
      (a) $f = \tilde{b}_{i^*}$
      (b) $\alpha = f * \tilde{b} + (1-f) * \tilde{b}'$ and $\beta = (1-f) * \tilde{b} + f * \tilde{b}'$
      (c) For $i = l-1, \ldots, 0$ do
              $([\alpha_i]^{P_1}_{N_1}, [\alpha_i]^{P_2}_{N_1}) \leftarrow$ ShareGen($\alpha_i, N_1$)
              $([\alpha'_i]^{P_1}_{N_2}, [\alpha'_i]^{P_2}_{N_2}) \leftarrow$ ShareGen($\alpha_i, N_2$)
              $([\beta_i]^{P_1}_{N_1}, [\beta_i]^{P_2}_{N_1}) \leftarrow$ ShareGen($\beta_i, N_1$)
              $([\beta'_i]^{P_1}_{N_2}, [\beta'_i]^{P_2}_{N_2}) \leftarrow$ ShareGen($\beta_i, N_2$)
          End for
 4: $P_j$ ($j \in \{1, 2\}$):
          $[c]^{P_j}_{N_1,B} = (1-t) * \pi^{-1}([\alpha]^{P_j}_{N_1,B}) + t * \pi^{-1}([\beta]^{P_j}_{N_1,B})$
          $[c]^{P_j}_{N_2,B} = (1-t) * \pi^{-1}([\alpha']^{P_j}_{N_2,B}) + t * \pi^{-1}([\beta']^{P_j}_{N_2,B})$
```

$\alpha_B = \bar{c'}_B = 111001$ *and* $\beta_B = \bar{b}_B = 00101$ *corresponding to the value of* f*. In step 4, based on the knowledge of* t*,* P_1 *and* P_2 *choose shares of* $c_B = \alpha_B = 111001$.

4.2.2. CMP. we extend our secure comparison shceme to obtain CMP protocol solving post-fix comparison problem without using pre-fix product method and LSB-gate technique as shown in Algorithm 6. All stages are described in the following:

- Stage 1-Compute the difference vector from $[a]_{N_2,B}$ and $[b]_{N_2,B}$:

  This Stage perfomes the same operation as that in FAST-CMP.

- Stage 2-Find the first different bit:

  Instead of P_1 sending $\tilde{b}_B$ to P_3, P_2 sends $\tilde{a}_B$ radomized using permutation function $\pi(.)$ and flipping sequence t_B. P_3 learns nothing about b_B from $\tilde{a}_B$ because the permutation function $\pi(.)$ and flipping sequence t_B are only know by P_1 and P_2. Then, P_3 reveals the sequence $[v]_{N_2,B}$ until finding the unique index i^* satisfying $v_{i^*} = 0$ and sets $w = \tilde{a}_{i^*}$, and sends $[w]_{N_1}^{P_1}$ to P_1 and $[w]_{N_1}^{P_2}$ to P_2. P_3 also computes a new sequence $\tilde{v}_B$, where $\tilde{v_{i^*}} = 1$, and $\tilde{v_i} = 0$, if $i \neq i^*$, and sends bit sharings of $\tilde{v}_B$ to P_1 and P_2.

- Stage 3-Adjust the result:

  P_j ($j \in \{1,2\}$) performs multiplication on $[\tilde{v}]_{2,B}^{P_j}$ and fliping sequence t_B to find out the bit $[\tilde{z}]_2^{P_j}$ indicating flipped or not flipped and choose $t' \in \{0,1\}$, then send $[h]_2^{P_j} = t'(1 - [\tilde{z}]_2^{P_j}) + (1-t')[\tilde{z}]_2^{P_j}$ to P_3. P_3 contributes to reveal h and sends new shares $[h]_{N_1}^{P_j}$ back to P_j. At last, P_j combine t', $[w]_{N_1}^{P_j}$ and $[h]_{N_1}^{P_j}$ to compute the comparison result.

  If we want to compute all post-fix bits string comparison as section 2.2 , stages-2 and Stage-3 are included in a loop increasing l multiplications. Thus the overall complexity is $2l$ multiplications.

Example 4 *Suppose we have the following inputs: P_1 and P_2's shares of $[e']_B = 0100$; $a_B = 1101$ from P_2; $t_B = 0111$ and $\pi(reverse)$ from P_1 and P_2. The Algorithm 6 runs as follows: first, P_2 permutes and flips a_B using $\pi(reverse)$ and flipping sequence t_B and obtains $\bar{a}_B = 1010$. P_3 finds the unique index $i^* = 2$ and sets $w = \bar{a}_2 = 0$ and $\bar{v}_B = 0100$. Next, P_1 and P_2 need to find out if $\bar{a}_2$ is flipped or not by checking the "summation" of the result of bit-wise multiplying $\bar{v}_B$ with t_B. Here, $\bar{a}_2$ is flipped, therfore, the output should be adjusted as shares of $c = 1 - w = 1$.*

Algorithm 6 CMP

Input: $\langle P_1, [a]^{P_1}_{N_2,B}, [b]^{P_1}_{N_2,B}, t_B, \pi(.)\rangle$, $\langle P_2, [a]^{P_2}_{N_2,B}, [b]^{P_2}_{N_2,B}, b_B, t_B, \pi(.)\rangle$ and $\langle P_3, \perp\rangle)$, where $\pi(.)$ is random permutation and t_B is a random binary string only known to P_1 and P_2

Output: $\langle P_1, [u]^{P_1}_{N_1}\rangle, \langle P_2, [u]^{P_2}_{N_1}\rangle$ where $u \equiv b \overset{?}{>} a$ and $u \in \{0,1\}$

```
1: $P_2$
    (a) ${a'}_B \leftarrow \pi(b_B)$.
    (b) $\tilde{a}_B = a'_B \oplus t_B$, where $\oplus$ denotes bit-wise xor operation.
    (c) Sends $\tilde{a}_B$ to $P_3$.
2: $P_1$, $P_2$ and $P_3$:
    (a) $(\langle P_1, [e]^{P_1}_{N_2,B}\rangle, \langle P_2, [e]^{P_2}_{N_2,B}\rangle) \leftarrow$
        DIFF-VEC$(\langle P_1, [a]^{P_1}_{N_2,B}, [b]^{P_1}_{N_2,B}\rangle, \langle P_2, [a]^{P_2}_{N_2,B}, [b]^{P_2}_{N_2,B}\rangle, \langle P_3, \perp\rangle)$
    (b) $(\langle P_3, i^*, w\rangle) \leftarrow$FIRST-DIFF-BIT $(\langle P_1, [e]^{P_1}_{N_2,B}, \pi(.)\rangle, \langle P_2, [e]^{P_2}_{N_2,B}, \pi(.)\rangle, \langle P_3, \perp\rangle)$
3: $P_3$
    (a) $\langle [w]^{P_1}_{N_1}, [w]^{P_2}_{N_1}\rangle \leftarrow ShareGen(w, N_1)$ .
    (b) Generate a new sequence $\tilde{v}_B$, where $\tilde{v_{i^*}} = 1$, and $\tilde{v_i} = 0$, if $i \neq i^*$, for $i = \tilde{l} - 1, \ldots, 0$
    (c) $\langle [\tilde{v}_i]^{P_1}_2, [\tilde{v}_i]^{P_2}_2\rangle \leftarrow ShareGen(\tilde{v}_i, 2)$, for $i = \tilde{l} - 1, \ldots, 0$
4: $P_j$ $(j \in \{1,2\})$
    (a) $[z_i]^{P_j}_2 = t_i \cdot [\tilde{v}_i]^{P_j}_2$, for $i = \tilde{l} - 1, \ldots, 0$
    (b) $[\tilde{z}]^{P_j}_2 = \sum_{i=0}^{l-1} [z_i]^{P_j}_2$
    (c) Send $[h]^{P_j}_2 = t'([1-\tilde{z}]^{P_j}_2) + (1-t')[\tilde{z}]^{P_j}_2$ to $P_3$, where $t' \in \{0,1\}$ is jointly selected.
5: $P_3$:
    (a) $h = ([h]^{P_1}_2 + [h]^{P_2}_2) \bmod 2$
    (b) $(\langle [h]^{P_1}_{N_1}, [h]^{P_2}_{N_1}\rangle) \leftarrow ShareGen(h, N_1)$.
6: $P_j$ $(j \in \{1,2\})$
    (a) $[\tilde{z}]^{P_j}_{N_1} = (t'([1-h]^{P_j}_{N_1}) + (1-t')[h]^{P_j}_{N_1}) \bmod N_1$
    (b) $[u]^{P_j}_{N_1} = ([\tilde{z}]^{P_j}_{N_1}([1-w]^{P_j}_{N_1}) + (1-[\tilde{z}]^{P_j}_{N_1})[w]^{P_j}_{N_1}) \bmod N_1$
```

4.3. COMPLEXITY AND SECURITY ANALYSIS

FAST-CMP takes 1 round and l multiplications to compare two sharing secrets. CMP also takes 1 round and l multiplications to implement post-fix comparison without using pre-fix product method and LSB-gate technique . In our secure comparison schemeds, the communication among parties occurs in all stages. During stage 1, the exclusive-or involves the interaction among parties for sharing secret multiplication. The security of this stage comes with the secure multiplication which is assumed to be secure, therefore, stage 1 is secure and no information is leaked. In stage 2, the generation of the γ sequence and permutation the sequence are all locally computed; the parties jointly generate uniformly random numbers and reveal them. u_i is also uniformly random since r_i is a uniformly random value from a finite field jointly generated by the parties and is independent of γ_i. Therefore, u sequence is secure and no information is leaked. During stage 3, the parties send their sequences of shares ot party P_3 and then P_3 performs the reconstruction process to reveal the "anchor". Although P_3 gets the revealing value, it does not know the position that d is located since the sequences are permuted by the parties based on the pre-determined one-way permutation function and thenP_3 has no chance of finding the being compared and comparator. Therefore, P_3 gains nothing valuable in stage 3.

5. EFFICIENT PROTOCOL FOR BINARY DECOMPOSITION

Suppose an l-bit value s is secretly shared among n parties, each of whom has a secret share $s_i \in \mathbb{Z}_P$ for $1 \leq i \leq n$. A secure bit-decomposition (SBD) protocol allows the n parties to convert their secret shares of s into secret shares of the individual bits of s. SBD has many important applications in secure multiparty computation, especially when the intermediate computation result of a secure computation cannot be disclosed, and the subsequent computation requires the input to be secret shares of individual bits of the intermediate result. The first SBD protocol was introduced by Damgård et al. Since then, more SBD protocols have been developed to reduce both communication and computation complexities. To our knowledge, among the existing SBD protocols, the most efficient protocol requires O(1) rounds of communication and O(l) multiplications. To further improve the efficiency of SBD, this paper introduces a novel and more efficient SBD protocol. The complexity of the proposed protocol matches the big O($\cdot$) bound while decreasing the hidden constants. More specifically, the proposed protocol is five times more efficient in the number of required multiplications and twice more efficient in round complexity. The reduced complexity comes from the fact that certain secure operations can be done with smaller share sizes without sacrificing any security guarantee.

5.1. MAIN CONTRIBUTION

We improved the bit-decomposition protocol presented by Reistad and Toft [4] by using our efficient comparison primitive In our SBD protocol, we no longer use post-fix comparison in which prefix-product dominate the overall complexity as described in section 2.2 and therefore no need of transforming bit-decomposition problem to post-fix problem, which gives us more efficiency in both computation and communication aspects. In addition, in the case of a or b is public, the dominate factor that affects complexity is removed and followed by further improved efficiency. At the end, we address the complexity comparison between our SBD and Bits-RT protocol. For security reason, We assume that the parties are connected by perfectly secure channels in a synchronous network.

5.2. SECURE BINARY DECOMPOSITION (SBD)

The secure comparison protocol can be apllied into many practical problems, and bit-decomposition problem is one of them. Therefore, we introduce a new improvement bit-decomposition protocol using the secure comparison scheme proposed in Section 4.2.

First, in our SBD protocol, we use SELECT to achieve the same functinality as step (4) and (5), which are comparison and selection phase, of Bits-RT protocol. Second, To achieve the post-fix comparison as in step (8) of Bits-RT protocol, we use CMP with small domain modulo N_2. With small domain in comparison protocol, the communication complexity can be significantly reduced. In addition, to obtain more efficiency, we put the DIFF-VEC in CMP protocol outside the loop. Therefore, the CMP is modified a little bit, denoting as CMP*. The input specification should be changed correspondingly: the output of DIFF-VEC is considered as part of the input to CMP*.

$$\begin{aligned}(\langle P_1, [u]^{P_1}_{N_1}\rangle, \langle P_2, [u]^{P_2}_{N_1}\rangle) \leftarrow \text{CMP*}(&\langle P_1, [e \bmod 2^i]^{P_1}_{N_2,B}, T^i_B, \pi(.)\rangle, \\ &\langle P_2, [e \bmod 2^i]^{P_2}_{N_2,B}, (r \bmod 2^i)_B, T^i_B, \pi(.)\rangle, \langle P_3, \perp\rangle)\end{aligned} \tag{5.1}$$

T_B is, a flipping sequence, used to flip individual bit (bit-wise flipping) of a sequence, hiding the sequence from P_3 so that P_3 learns nothing since T_B are only know by P_1 and P_2. The protocols hierarchy is shown in figure 5.1

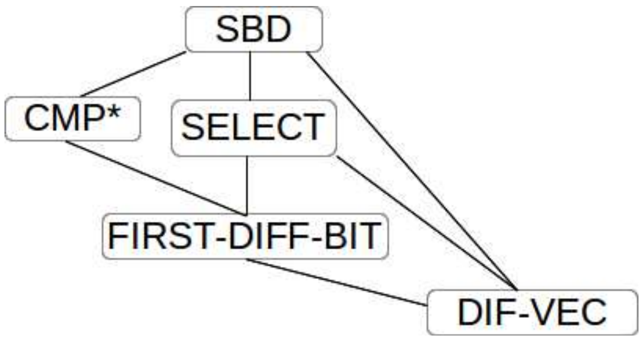


Figure 5.1: The protocol hierarchy

Example 5 *Suppose we have public parameters:* $N_1 = 31$, $N_2 = 11$, $l = 5$. *We assume* $x = 29(11101)$, *and* $x^{P_1} = 12$, $x^{P_2} = 17$; *we use the function that reverse the order of l-bit string as permutation function:* $\pi(reverse)$ *(e.g.,* $01101 \leftarrow \pi(10110)$*).*

(1). Step 1:(a)-(b): P_2 *choose* $r = 22(10110)$, *and sends* $X^{P_2} + r$ *to* P_1. *Then* P_1 *computes* $b = x^{P_1} + x^{P_2} + r \bmod 31 = 20(10100)$ *and* $c' = b + N_1 = 51(110011)$.

(2). Step 1:(c): P_1 *and* P_2 *choose* $t = 0$, *and* $T_B = \{T_B^{l-1}, \dots, T_B^1\} = \{T_B^4, T_B^3, T_B^2, T_B^1\} = \{0111, 101, 10, 1\}$.

(3). Step 2: as in Example 3, P_1 *and* P_2 *get the shares of* $c_B = 111001$ *in different domains.*

(4). Step 3:(a): P_1 *and* P_2 *gets the shares of* $e'_B = 111001 \oplus 01101 = 110100$.

(5). Step 3:(b)-(d): As in Example 4, $u_4 = 1$, *we gets* $u_3 = 1$, $u_2 = 0$ *and* $u_1 = 0$ *by running CMP* repeatedly. Now, we can compute locally:* $[x \bmod 2^5] = [x] = [29]$ $[x \bmod 2^4] = [(0011)]_2 - [(0110)]_2 + 2^4 \cdot [u_4] = [13]$; $[x \bmod 2^3] = [5]$; $[x \bmod 2^2] = [1]$; $[x \bmod 2^1] = [1]$. *Then, we compute locally:* $[x_4] = \frac{[x \bmod 2^5] - [x \bmod 2^4]}{2^4} = 1$. $[x_3] = 1$; $[x_2] = 1$; $[x_1] = 0$; $[x_0] = [29 \bmod 2] = 1$.

Algorithm 7 SBD

```
Input: $\langle P_1, [x]_{N_1}^{P_1}, \pi(.)\rangle$, $\langle P_2, [x]_{N_1}^{P_2}, \pi(.)\rangle$ and $\langle P_3, \perp\rangle$, where $\pi(.)$ is random permutation only
  known to $P_1$ and $P_2$. $N_1$ and $N_2$ are public parameters where $N_2 = \lceil 2\log N_1\rceil$
Output: $\langle P_1, [x]_{N_1,B}^{P_1}\rangle$ and $\langle P_2, [x]_{N_1,B}^{P_2}\rangle$
  $P_2$:
  (a) $r \in_R Z_{N_1}$, $t \in_R \{0,1\}$, and flipping sequences $T_B^i \equiv (T_{i-1}^i, \ldots, T_0^i)$, where $T_j^i \in_R \{0,1\}$ for
      $0 \le j \le i-1$ and $1 \le i \le l-1$
  (b) $\langle [r_i]_{N_1,B}^{P_1}, [r_i]_{N_1,B}^{P_2}\rangle \leftarrow$ ShareGen$(r_i, N_1)$, for $i = l-1, \ldots, 0$
      $\langle [r_i]_{N_2,B}^{P_1}, [r_i]_{N_2,B}^{P_2}\rangle \leftarrow$ ShareGen$(r_i, N_2)$, for $i = l-1, \ldots, 0$
  (c) Send $([x]_{N_1}^{p_2} + r) \bmod N_1$, $t$ and $T_B$ to $P_1$
  $P_1$:
  (a) $b \leftarrow [x]_{N_1}^{p_1} + [x]_{N_1}^{p_2} + r \bmod N_1$ and $b' = b + N_1$
  (b) $\langle [b_i]_{N_1,B}^{P_1}, [b_i]_{N_1,B}^{P_2}\rangle \leftarrow$ ShareGen$(b_i, N_1)$, for $i = l-1, \ldots, 0$
      $\langle [b_i]_{N_2,B}^{P_1}, [b_i]_{N_2,B}^{P_2}\rangle \leftarrow$ ShareGen$(b_i, N_2)$, for $i = l-1, \ldots, 0$
  (c) $\langle [b'_i]_{N_1,B}^{P_1}, [b'_i]_{N_1,B}^{P_2}\rangle \leftarrow$ ShareGen$(b'_i, N_1)$, for $i = l-1, \ldots, 0$
      $\langle [b'_i]_{N_2,B}^{P_1}, [b'_i]_{N_2,B}^{P_2}\rangle \leftarrow$ ShareGen$(b'_i, N_2)$, for $i = l-1, \ldots, 0$
  $P_1, P_2$ and $P_3$
  (a) $(\langle P_1, [c]_{N_1,B}^{P_1}, [c]_{N_2,B}^{P_1}\rangle, \langle P_2, [c]_{N_1,B}^{P_2}, [c]_{N_2,B}^{P_2}\rangle) \leftarrow$ SELECT$(\langle P_1, [r]_{N_2,B}^{P_1},$
          $[b]_{N_2,B}^{P_1}, b_B, b'{}_B, t, \pi(.)\rangle, \langle P_2, [r]_{N_2,B}^{P_2}, [b]_{N_2,B}^{P_2}, t, \pi(.)\rangle, \langle P_3, \perp\rangle)$
  $P_j$ $(j \in \{1,2\})$
  (a) $(\langle P_1, [e]_{N_2,B}^{P_1}\rangle, \langle P_2, [e]_{N_2,B}^{P_2}\rangle) \leftarrow$ DIFF-VEC$(\langle P_1, [r]_{N_2,B}^{P_1}, [c]_{N_2,B}^{P_1}\rangle, \langle P_2, [r]_{N_2,B}^{P_2}, [c]_{N_2,B}^{P_2}\rangle, \langle P_3, \perp$
      $\rangle)$
  (b) $[x \bmod 2^l]_{N_1}^{P_j} = [x]_{N_1}^{P_j}$
  (c) For $i = l-1, \ldots, 1$ do
        $(\langle P_1, [u]_{N_1}^{P_1}\rangle, \langle P_2, [u]_{N_1}^{P_2}\rangle) \leftarrow$ CMP*$(\langle P_1, [e \bmod 2^i]_{N_2,B}^{P_1}, T_B^i, \pi(.)\rangle,$
          $\langle P_2, [e \bmod 2^i]_{N_2,B}^{P_2}, (r \bmod 2^i)_B, T_B^i, \pi(.)\rangle, \langle P_3, \perp\rangle)$
        $[x \bmod 2^i]_{N_1}^{P_j} = [c \bmod 2^i]_{N_1,B}^{P_j} - [r \bmod 2^i]_{N_1,B}^{P_j} + 2^i [u_i]_{N_1}^{P_j}$
        $[x_i]_{N_1}^{P_j} = ([x \bmod 2^{i+1}]_{N_1}^{P_j} - [x \bmod 2^i]_{N_1}^{P_j}) * 2^{-i}$
      End for
  (d) $[x_0]_{N_1}^{P_j} = [x \bmod 2]_{N_1}^{P_j}$
```

5.3. COMPLEXITY AND SECURITY ANALYSIS

5.3.1. Complexity Analysis.

5.3.1. Complexity Analysis. Bits-RT protocol in section 2.2 takes 5 rounds and $24l + 12$ multiplications for preprecessing phase, and for online phase, it takes 2 rounds and $7l + 3$ multiplications for comparison between a random bit strings and a public value in step (4), and 5 rounds and $9.5l$ multiplications for post-fix compariosn in step (8). Therefore, the overall online complexity is

Table 5.1: Complexity of Bit-decomposition Protocol

Protocols	Rounds	Multiplications
Compare	2	$7l + 3$
Post-fix Compare	5	$9.5l$
Bits (online)	7	$16.5l + 3$

(a) Bits-RT protocol [4]

Protocols	Rounds	Multiplications
SELECT	1	l
DIFF-VEC	1	l
CMP	1	l
SBD	3	$3l$

(b) Our SBD protocol

Table 5.2: Complexity Comparison

Protocols	Rounds	Multiplications
Bits-RT [4] (online)	7	$16.5l + 3$
SBD	3	$3l$

7 rounds and $16.5 + 3$ multiplications as shown in Table 5.1(a). While our proposed secure comparison scheme reduces the complexity of comparison between two secrets, and further improved the bit-decomposition protocol. The proposed SELECT protocol takes 1 rounds and $1l$ multiplications and CMP* takes 2 rounds and $2l$ multiplications as dipicted in section 4.2. By applying our secure comparison scheme, SBD protool takes 3 rounds and $3l$ multiplications to solve bit decomposition problem, detailied in Table 5.1 (b). The complexity comparison of our approaches with Bits-RT protocol is shown in Table 5.2

Table 5.3: Message Size Comparison

Protocol	Parallelized steps	Message Size
Bits-RT	$4 + 16.5k$	$(28.5k + 6)l^2 + 18.5\kappa kl$
SBD	$10 + 2k$	$(12 + 2\log l)l^2 + (18 + 4k)l\log l$

Communication message size of Bits-Toft protocol is around $(28.5k+6)l^2 + 18.5\kappa kl$ while our proposed protocol is $(12 + 2\log l)l^2 + (18 + 4k)l\log l$, where k is number of communication steps involved in one multiplication. In our protocol, we use shares in different finite fields, such as N_1 and N_2. Therefore, our proposed protocol is also more efficient cosidering communication complexity as shown in Table 5.3.

5.3.2. Security Analysis. In our secure comparison scheme, the communication among parties occurs in all stages. During stage 1, the exclusive-or involves the interaction among parties for sharing secret multiplication. The security of this stage comes with the secure multiplication which is assumed to be secure, therefore, stage 1 is secure and no information is leaked. In stage 2, the generation of the γ sequence and permutation the sequence are all locally computed; the parties jointly generate uniformly random numbers and reveal them. u_i is also uniformly random since r_i is a uniformly random value from a finite field jointly generated by the parties and is independent of γ_i. Therefore, u sequence is secure and no information is leaked. During stage 3, the parties send their sequences of shares ot party P_3 and then P_3 performs the reconstruction process to reveal the "anchor". Although P_3 gets the revealing value, it does not know the position that d is located since the sequences are permuted by the parties based on the pre-determined one-way permutation function and thenP_3 has no chance of finding the being compared and comparator. Therefore, P_3 gains nothing valuable in stage 3.

6. PROXY RE-ENCRYPTION

6.1. MAIN CONTRIBUTION

Most existing proxy re-encryption (PRE) schemes are implemented by pairings. These pairings are useful for realizing a scheme's construction.A bilinear mapping function can be used to check a ciphertext's validity. If the proxy is malicious, a PRE scheme should be able to check the correctness of ciphertexts. Therefore, the validation of ciphertext should be considered when designing a secure unidirectional PRE scheme. In the paper, it adopts strongly unforgeable signature technique to implement the validation of re-encrypted ciphertext and then construct efficient and feature-rich unidirectional PRE schemes with simple design (no pairings). At last, the paper measures the implementation of PRE in a secure file system. The secure file system uses an untrusted access control server to manage accesses to encrypted files stored on a distributed storage.

6.2. PROPOSED PROXY RE-ENCRYPTION (PRE) SCHEMES

A paillier public key encryption (PKE) scheme was used to encrypt all of the data from the proposed construction by data owner. It was then stored in a distributed file storage. Suppose an authorized user (e.g., Bob) has the right to access this data. Before he can access this information, however, he needs to obtain the secret information used to perform decryption. This information is either Alice's secret key or random number seed (RNS) generating random number to randomize Paillier encryption. Alice, however, does not want to share her secret key with others due to security concerns. Therefore, Bob can only access to data through mathematical decryption with RNS as described in Section **??**.

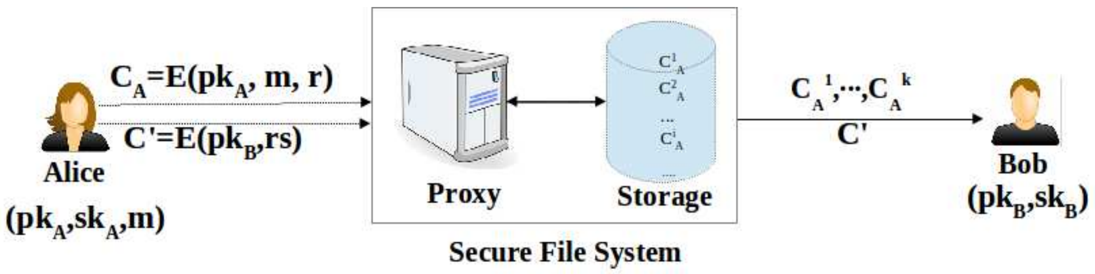


Figure 6.1: Overview of our single-proxy basic construction

Assume a PKE scheme given to the proxy was used to encrypt the RNS under a user's public key. This PKE can be any type of secure public key encryption scheme (If the Paillier PKE scheme is used, the prime products must be different for security reasons.). In Lemma 1, Bob must submit a request to the proxy for the secret information if he wants to access the data. The proxy in this proposed construction does not perform any computation on the ciphertexts to respond to the requests. This is the main difference form the existing PRE schemes. The existing schemes require the proxy to perform re-encryption on the ciphertext when receives the access request from users.

6.2.1. Single-Proxy Basic Construction. The file system in this single-proxy scheme has only one proxy to control access to the encrypted data in storage. An overview of the single-proxy PRE's basic construction can be seen in Figure 6.1.

Assume the proxy is in a semi-honest model, and only one proxy (P) controls access to the data. One simple solution is available to construct a PRE scheme. In the solution, Alice send the RNS, which can be used to decrypt ciphertext encrypted by Paillier PKE scheme, to the proxy in plaintext. If the authorized user (Bob) requests RNS for decryption, the proxy will pass RNS to him. This case, however, contains only one proxy. Thus, the proxy learns the valuable information RNS used to generate the random number (r) which is integrated in the ciphertext. instead of sending plaintext of RNS, Alice provides the encryption of that under Bob's public key to the proxy. This solution can keep the data in storage secure and protect the privacy of data owner

An assumption similar to the Paillier PKE is made here. Assume the public key is $N = p * q$, where p and q are large primes with equal size. The random number (r) generated by RNS is within the range $\{0, N^2\}$, and the message (m) is in $\{0, N-1\}$. The construction's detail are described in the following algorithms.

- Setup (k): Input of the security parameter k, and choose two large prime numbers (p and q) with the same size.
- KeyGen (k): Set $N = pq$ and $g = N + 1$, $\lambda = \varphi(N)$, where $\varphi(N)$ is Euler's totient function. Set $\mu = \varphi(N)^{-1}$. The public key is $pk = (N, g)$, and the private key is $sk = (\lambda, \mu)$.
- Enc (pk_A, m): Alice encryptes the message (m) in such a way that $C = g_A^m r^{N_A}$. Alice sends all encrypted data to the storage and sends encrypted RNS rs: $C' = E(PK_B, rs)$ to P.
- Dec1 (sk_A, C): Alice can use Eq.(3.2) to decrypt the ciphertext C with sk_A
- Dec2 (C, C', sk_B): Bob uses input from data ciphertext C and RNS ciphertext C'. He decrypts C' to obtain rs and computes r corresponding to C, and further uses Eq.(3.5) to decrypt the ciphertext by $m = L(C(r^{N_A})^{-1}) \bmod N_A$

This construction is secure under the assumption of a semi-honest proxy model. It achieves colluding safe under adversary model M1, but fails to achieve *verifiable* and *CCA secure* under adversary model M2 in malicious proxy models. Alice keeps secret key used to decrypt all ciphertexts safe, even if the Proxy and Bob collude. Bob, however, has no way of verifying the correctness of ciphertext of RNS if proxy is malicious with ability of compromising the re-encryption. The Paillier PKE achieves CPA; it does not achieve CCA. Therefore, this construction is not CCA secure. The RNS is encrypted under Bob's public key, allowing only

Bob to decrypt it. This means other users cannot obtain the RNS, even if the proxy offers the ciphertext to them. Therefore, the RNS cannot be distributed further without Alice's permission (assuming Bob is honest). In Enc step, Since the Paillier PKE scheme is adopted to encrypt the RNS. The malicious proxy has ability of modifying the ciphertext without been caught since the paillier PKE is chosen plaintext attack (CPA)-secure. A scheme against adversary model M2 was developed to avoid this situation. This scheme is discussed in the following section.

6.2.2. Single-Proxy Construction Against M2. Unlike the Single-Proxy Basic Construction, this proxy is under adversary model M2. Alice uses a strongly unforgeable signature (as described in Section 3.4) to sign the ciphertext and thus validate the RNS. She then provides both the ciphertext and the signature to the proxy. The proxy's malicious behavior can now be detected.

Assume a strongly unforgeable, one-time signature scheme $Sig = (\zeta, S, V)$ exists, where ζ is a key generation protocol, S is a message signing protocol, and V is a signature verifying protocol and a one-time key pair (sk, vk). The construction detail can be seen as follows:

- Setup (k): This setup is the same as that described in Section 6.2.1.
- KeyGen (k): The keyGen is the same as that described in Section 6.2.1.
- Enc (pk_A, m_i):Alice proceeds as follows to encrypt a message $m_i \in Z_N$ under public key pk_A, where $0 \leq i \leq |D|$, $|D|$ is the database size:
  - (1). Select a one-time signature key pair $(sk, vk) \leftarrow \zeta(\lambda)$.
  - (2). Generates random number r_i using RNS rs and computes: $C_i = E(pk_A, m_i) = g_A^{m_i} r_i^{N_A} \bmod N_A^2$.
  - (3). Encrypts the RNS rs with Bob's public key pk_B: $C' = E(pk_B, rs) = g_B^{rs} r^{N_B}$.

(4). Computes a one-time signature: $\sigma = S(sk, C')$ on C'.

Alice stores ciphertexts to the storage and sends (C', σ) to P.

- Dec1 (sk_A, C): This decription is the same as that described in Section 6.2.1
- Dec2 (C, C', σ, sk_B): With input of data ciphertext C, RNS ciphertext C' and the signature on C', Bob first checks the validity of the ciphertext of RNS by testing the following condition:

$$V(\sigma, vk, C') = 1 \tag{6.1}$$

If valid, Bob decrypts C' to get rs, computes r_i corresponding to C_i and then gets the plaintext message m_i by using Eq.(3.5): $m_i = \frac{(C_i(r_i^{N_A})^{-1} \bmod N_A^2) - 1}{N}$; $\perp$, otherwise.

With strongly unforgeable signature, the validity of ciphertext of RNS can be verified by Bob and the malicious behavior of proxy can be detected. In one case that the proxy altered the ciphertext of RNS, Bob will find this behavior by verifying the Alice's signature. In the other case that the proxy modified ciphertext of data, Bob can detect this behavior by checking if decryption result is within the message domain. This construction further achieved *verifible* feature and against adversary model M2. Bob is able to check the validity of RNS by through testing condition 6.1. Further, the signature prevents the proxy from altering the ciphertext willingly without being caught since the signature is strongly unforgeable. We can detect the behavior of modifying ciphertexts of data, However, theses ciphertexts are vulnerable to chosen-ciphertext attack (CCA). Then, we need further design a more secure scheme to protect the data in storage.

6.2.3. Single-Proxy Construction Against CCA . Although we can detect the proxy's malicious behavior of modifying ciphertext of data, we have no way of preventing that happens in the above schemes. Here, we provide a more secure PRE scheme preventing modification over ciphertext of data. By applying the similar mechanism to protect the RNS as in Section 6.2.1, we further protect data by adopting signature scheme. Therefore, all the ciphertexts in the storage were signed by Alice using strongly unforgeable signature scheme before they were stored. The detail of construction is described as follows:

- Setup (k): This setup is the same as that described in Section 6.2.1.
- KeyGen (k): The keyGen is the same as that described in Section 6.2.1.
- Enc (pk_A, m_i):To encrypt a message $m_i \in Z_N$ under public key pk_A, where $0 \leq i \leq |D|$, $|D|$ is the database size, Alice proceeds as follows:

  (1). Select a one-time signature key pair $(sk, vk) \leftarrow \zeta(\lambda)$.

  (2). Generates random number r_i using the seed rs and computes: $C_i = E(pk_A, m_i) = g_A^{m_i} r_i^{N_A} \bmod N_A^2$.

  (3). Encrypts the seed rs with Bob's public key pk_B: $C' = E(pk_B, rs) = g_B^{rs} r^{N_B}$.

  (4). Computes a one-time signature: $\sigma_r = S(sk, C')$ on C' and $\sigma_i = S(sk, C_i)$ on C_i.

  Alice stores ciphertext and their signatures (C_i, σ_i) to the storage and sends (C', σ_r) to P.

- Dec1 (sk_A, C): This decription is the same as that described in Section 6.2.1
- Dec2 ($C, \sigma, C', \sigma_r, sk_B$): With input of data ciphertext C, RNS ciphertext C' and signatures on C and C', Bob first checks the validity of the ciphertexts by testing the following conditions:

$$V(\sigma, vk, C') = 1 V(\sigma_i, vk, C_i) = 1 \qquad (6.2)$$

Table 6.1: Properties of proposed single-proxy constructions

Constructions	Basic	Against M2	Against CCA
Collusion safe	Yes	Yes	Yes
Non-interactive	Yes	Yes	Yes
Unidirectional	Yes	Yes	Yes
Key optimal	Yes	Yes	Yes
Non-transitive	Yes	Yes	Yes
Verifiable	No	Yes	Yes
CCA secure	No	No	Yes

If valid, Bob decrypt C' obtains rs and computes r_i corresponding to C_i and then gets the plaintext message m by Eq.(3.5): $m = \frac{(C_i(r_i^{N_A})^{-1} \bmod N_A^2)-1}{N}$; $\perp$, otherwise.

In this construction, the ciphertexts of data and the ciphertext of RNS are signed by Alice, the data owner, using strongly unforgeable signature before they were distributed. Because the signature is unforgeable, the adversary can do nothing to compromise the ciphertext and further the data. Therefore, this construction is CCA secure. Further analysis can be seen in section 6.3.

6.2.4. Multi-Proxy Construction with Searchable Feature. To meet some certain needs, like searchable feature, we assume there are multiple semi-honest proxies (here is two proxies P_1 and P_2, where P_2 controls the distributed file storage, and P_1 stores the random number seed (RNS)). We first gives the basic construction without strong security, which will be further improved in the following section. To decrypt the ciphertexts from P_2, Bob receives the plaintext of RNS, which is used to generate random number, a randomization factor in the Paillier PKE scheme, from P_1. With RNS, Bob computes random number corresponding to the ciphertext, then obtains the message by decrypting the ciphertext following Eq.(3.5).

- Setup (k): This setup is the same as that described in Section 6.2.1.
- KeyGen (k): The keyGen is the same as that described in Section 6.2.1.

- Enc (pk_A, m): Alice encrypts message m in such a way $C = g_A^m r^{N_A}$ and stores C in the storage and sends RNS rs to P_1.
- Dec1 (sk_A, C): With sk_A, Alice can decrypt the ciphertext C by computing as Eq.(3.2)
- Dec2 (C, rs): With input of ciphertext of data C and RNS rs, Bob is able to compute r corresponding to C, and further decrypts the ciphertext by computing $m = L(C(r^{N_A})^{-1}) \bmod N_A$ as Eq.(3.5)

Since we assume P_1 and P_2 are semi-honest, they follows the protocol and learns nothing about the data and secret key of Alice except the RNS and the ciphertext of data. Although, the RNS can be used to decrypt the ciphertext indirectly, the data will be in a safe box as long as P_1 and P_2 are not colluding. Therefore, the security of the above construction is semantic secure which comes from the Paillier PKE scheme.

Followed the same assumption as the above multi-proxy construction, we further design a searchable PRE scheme. In practice, the delegatee Bob may have to search for a certain kind email, for example, emails related to classes, to decrypt in email forwarding scenario. In this case, the construction should achieve searchability. Originated from [62], we keep the encryption of message and encryption of keyword separate so that we can have the flexibility to select which standard PRE and PRES (Searchable PRE) schemes to be used for satisfying the requirements of the actual applications. Here, we constructed an efficient PRES scheme based on Paillier PKE scheme.

Each ciphertext of message is associated a ciphertext of keyword w encrypted under the data owner Alice's public key pk_A. When the Bob chooses to decrypt one of the ciphertext, he/she encrypts the query q with pk_A and sends to P_2 to request this ciphertext. Then P_2 computes the difference C_{diff} between the keyword and the query, and based on the communication result with P_1, P_2 response with requested ciphertext or nothing.

- Setup (k): This setup is the same as that described in Section 6.2.1.
- KeyGen (k): The keyGen is the same as that described in Section 6.2.1.
- Enc (pk_A, w): Alice encrypts keyword w in such a way $C_{key} = g_A^w r^{N_A}$ associated with the ciphertext of data and sends RNS rs to P_1.
- Query (pk_A, q): Bob generates the query by encrypting word q with Alice's public key, and sends $C_q = E(pk_A, q)$ to P_2.
- Test (C_q, C_{key}): With input of query C_q and keyword ciphertext C_{key}, P_2 computes $C_{diff} = C_{key} * C_q^{-1}$, then randomizes C_{diff} and gets C'_{diff}. P_2 sends C'_{diff} to P_1 who can decrypt it. P_2 response with requested ciphertext C_{data} or nothing based on the communication result with P_1.
- Dec1 (sk_A, C_{data}): With sk_A, Alice can decrypt the ciphertext C_{data} by computing Eq.(3.2)
- Dec2 (C_{data}, rs'): With input of C_{data} and rs' (different from rs, rs' is used to generate random number for encrypting data). Then, Bob generates r corresponding to C_{data}, and further decrypts the ciphertext by computing $m = L(C_{data}(r^{N_A})^{-1}) \bmod N_A$ as Eq.(3.5)

The computation of C_{diff} can be based on the exact match or similarity. In the exact match based, the matching requirement is that the query and the keyword are the same, that is, C_{diff} is the ciphertext of 0, while in similarity based, the matching requirement is that the matched one is with highest similarity. Here, we provide protocols to compute cosine similarity which is a popular way of measuring the similarity (see section 3.5). With cosine similarity, the highest value means the best match. In order to better using cosine similarity method, we preprocess the keyword and query into vectors with the same length l and encrypt them component wise using Paillier PKE.

6.3. SECURITY ANALYSIS

In this section, we mainly analyze the security of our proposed CCA-secure construction in section 6.2.3. The following game is played by a challenger c and an adversary Λ:

- KeyGen: The challenger c executes the keygen algorithm n_h times resulting a set of public/private key pairs (pk_h, sk_h) and runs the keygen algorithm for n_c times again to obtain a set of corrupted public/private key paris (pk_c, sk_c). Then c sends the sk_c and all $pk = (pk_h \cup pk_c) = pk_{i i\in[1,n_h+n_c]}$.
- Step 1: Adaptively query decryption oracle: the adversary adaptively sends decryption request $(pk, C)$to c, the decryption oracle responds with the decryption of C using the sk with respect to pk
- Challenge: After the adversary Λ stops the step 1, it chooses two equal-length plaintexts m_0 and m_1 from plaintext domain and a target public key pk^*, then sends them to the challenger c. The challenger c randomly select $b \leftarrow_R \{0, 1\}$, and sends C^* depending on pk^* and m_b to the adversary Λ.
- Step 2: The adversary Λ continues to issue decryption queries as in step 1.
- Guess: Λ outputs the guess $b' \in \{0, 1\}$.

The public key in input from the adversary should meet the following constraints:

(1). All public keys in the input should be generated by the challenger

(2). The target public key pk^* should be from pk_h sets.

(3). After the challenge step, Λ cannot query the decryption oracle with C^*

Definition 4 *CCA-secure scheme: We define the adversary Λ's advantage in attacking a scheme as*

$$Adv_{CCA}^{\Lambda} = |Pr[b = b'] - \frac{1}{2}| \tag{6.3}$$

where the probability is taken over the random b chosen by the challenge and b' chosen by the adversary. The scheme is said to be $(t, n_u, n_c, q_d, \epsilon) - CCA$ secure, if for any polynomial t-time adversary Λ who queries at most q_d times over the decryption queries, we have $Adv_{CCA}^{\Lambda} \leq \epsilon$.

Theorem 1 *If the signature is strongly unforgeable and assumptions for Paillier PKE holds, our CCA-secure construction is $(t, n_u, n_c, q_d, \epsilon) - CCA$ secure.*

6.4. SECURE DISTRIBUTED STORAGE SYSTEM

In PRE scheme, an untrusted Proxy is given a transformation key to transform ciphertexts under Alice's public key into another ciphertext that Bob can decrypt with his secret key, without the Proxy knowing the plaintext. Because of the nature of this transformation, PRE scheme can be applied to many applications, such as access to distributed storage [9]. A storage system that uses an untrusted proxy (access control server) to manage accesses to encrypted data stored on a distributed, untrusted storage. We use proxy re-encryption to allow for access control without granting full decryption rights to the proxy.

Overview: In the storage system, a data owner stores all the encrypted data to the distributed, untrusted storage and selects which users, say delegatee user, should have right to access to the data and gives the appropriate delegation rights to the proxy by distributing the encrypted RNS under delegatee's public key to the proxy. The delegatee users on client machines wish to get access to the encrypted confidential data stored in the storage.

Access Control using CCA-secure PRE: The delegatee users download the encrypted content from the storage, then communicate with the Proxy to get the RNS used to protecting the data. Because the proxy does not possess the corresponding secret key, it cannot be corrupted so as to gain access to the RNS necessary to access encrypted data. The delegatee has the ability to verify the validity of ciphertext of RNS, and the ciphertext as well, using delegator's public key, which preserves the confidentiality of data. Moreover, the secret key used to decrypt the data remains offline, which cannot be corrupted in any kind of attack.

The architecture has significant advantages over systems with trusted access control servers. The key material stored on the proxy cannot be used to access stored data directly, which reduces the need to absolutely trust the proxy and diminishes the proxy's value to attackers. The secret key itself only required by a content owner when encrypts the confidential data at the begining and can therefore be stored safely offline where it is less vulnerable to compromise.

7. PRIVACY-PRESERVING USER AUTHENTICATION

The protection of user privacy when performing web-based transactions has become an important factor in the acceptance and widespread use of on-line location based services, such as navigation, site recommendation, finding the nearest restaurant or gas station. Service providers generally control user access to their on-line resources and impose accountability of user actions. Under these situations, a user needs to prove her identity or the possession of a certificate which may contain a pseudo identity of the user or the necessary attributes required for accessing certain location based services. Since a user's physical location can reveal the actual identity of the user, it is in the best interest of the user not to be tracked by service providers or peer users. In many non-critical scenarios, a service provider may only need to know whether the user is authenticated or not, but does not need to know the user's actual identity. Thus, user privacy should be preserved during authentication in that their identities should be kept private in order to avoid unlawful tracing and user profiling. More specifically, regarding location based services, these entities may cause privacy concerns: (i) the authentication server(s) and (ii) service providers. The authentication servers and the service providers may obtain the behavior pattern or track the user locations according to user authentication records to access particular services. We refer to the privacy concern caused by the server as the server-wise privacy and the privacy concern caused by peer users as the peer-wise privacy. Ideally, we should preserve both server-wise for each mobile user.

7.1. MAIN CONTRIBUTION

To overcome the shortcomings in the well-known AC system [63, 64], we propose a novel anonymous credential or privacy-preserving authentication system (namely RAU) that truly preserves user privacy while still ensures traceability. The proposed protocols of the system are designed based on homomorphic encryption [2], and they allow each user to self-generate any number of randomized authenticated identities to prove his or her legal status when communicating with the service providers of location based services. In fact, users will be able to easily use a fresh pseudo or randomized identity for each newly established communication. These randomized identities can be verified through the collaboration of a pair of authentication servers while each authentication server would not know the real identity of the authentication requester. In this way, we achieve the server-wise privacy preservation. For traceability, the pair of authentication servers need to collaboratively execute a protocol to reveal the identity of a malicious user without disclosing the identities of other trustworthy users. We summarize the advantages of our proposed authentication protocol as follows.

- Under our anonymous authentication system, users' real identities are hidden from each individual party including authentication servers and service providers.

- Our system achieves a set of desired security and privacy properties such as *unforgeability*, *unlinkability* and *traceability*. It is robust against various types of attacks (discussed in Section 7.3).

- Our approach no longer has the key revocation problem neither the costly group management. Specifically, users using the proposed protocol no longer need to preload a huge number of keys (i.e., pseudonyms) or rely on others (i.e., peers or infrastructure) to generate the pseudonyms. Our experimental study demonstrates the proposed protocol is very efficiency.

- Our protocol does not require users to be equipped with high performance computing equipment since almost all computations are outsourced to the

Table 7.1: System comparison

Properties	**AC**	**RAU**
Against the replay attack	×	√
Ease of credential revocation	×	√
Privacy-preserving traceability	×	√
Preventing credential sharing	×	√
Non-interactive credential issuers	√	×
No collusion issues during authentication	√	×

servers and the users only need to generate several encryptions and random numbers from a pre-defined domain.

- Since anonymity revocation needs not to be done as a real-time application (due to court orders), our protocol provides reasonable computation time (as presented in Section 7.4).
- Our approach does not have the credential sharing problem, and it is secure against the replay attack.

Table 7.1 summarizes the advantages of our approach compared with the baseline AC system proposed in [63, 64].

7.2. PROPSED RANDOMIZED AUTHENTICATION (RAU)

In this section, we present the proposed privacy-preserving authentication system. We first discuss the system setup and give an overview of the related protocols. Then we elaborate on the details in each protocol of the authentication system.

The proposed authentication system consists of three kinds of entities: authentication servers, users and service providers. In the current version, the system has two authentication servers, namely Registration Server (RS), and Verification

Server (VS). The two servers collaborate with each other to conduct privacy-preserving user authentications, and hence none of them is able to track the user alone.

In addition, we assume that the users and the service providers can communicate with the authentication servers.

When designing each specific protocol, we aim to achieve the following security requirements of the anonymous authentication system:

- User authentication: we can verify if a user has a legitimate right to obtain the service from a service provider.
- Preserving user anonymity: the real identity of a legitimate user should not be known by other entities (e.g., peer users and service providers), and these entities should not be able to track a user's behavior by linking multiple authentication messages to the same user.
- Providing traceability: if necessary and under lawful request, the two authentication servers will be able to collaboratively reveal the real identity of a malicious user.

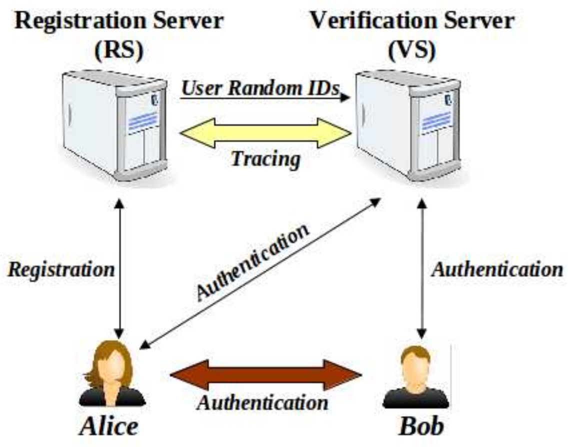


Figure 7.1: An Overview of the Data Flow

Table 7.2: List of Notations

Notation	**Meaning**
RAU	Randomized Authentication
RS	Registration Server
VS	Verification Server
E(x,r)	Encrypt x using RS' public key
D(y)	Decrypt the ciphertext y
ID_u	Real identity of user u
RID_u	Randomized identity of user u
r_u^i	Random number generated at i^{th} round

The proposed authentication system has three main phases: (1) user registration, (2) user authentication, and (3) identity tracing. Figure 7.1 illustrates an overview of the data flow in the system. At the beginning, users register at the RS server. The RS server shares part of the information of users' pseudo identities with the VS server. Whenever users want to communicate with others, they can randomly generate pseudo identities which can be verified by the VS server. If there is any dispute, the two servers will conduct a tracing protocol to figure out the real identity of a malicious user. The detailed steps in each phase will be presented in the following subsections. For clarity, Table 7.2 lists the frequently used notions in this paper.

7.2.1. User Registration. To begin with, the RS server generates its own public-private key pair using the Paillier encryption scheme, and the public key is known by all entities in the network. Users will always communicate with the servers through a secure channel. Specifically, a session key between a user u and a server can be generated using any well-known method, e.g., public key infrastructure utilizing the servers' digital certificate. The remaining communication between the server and the user will be encrypted using the session key only known to them. Similarly, the communication between the two authentication servers is also via a secure channel.

User registration is an on-going process. That is, a new user can join the system at any moment. To register, a user u sends certain identification information (ID_u) such as driver license number* to the registration server (RS) via the secure channel. If needed, the RS server can further verify u's identification information via a third party (e.g., an agency who performs background check for credit card applications or visas). How to achieve robust identify verification is out of the scope of this paper, but the RS server can use any existing solutions. *Note that a user's identification or identifiable information is only needed to achieve traceability; otherwise, any value can be used as the seed to generate the initial randomized ID.*

The RS server computes an initial randomized authentication ID (RID_u^0) for user u as follows:

$$\mathrm{RID}_u^0 = E(\mathrm{ID}_u, r_u^0) \tag{7.1}$$

where $E(\mathrm{ID}_u, r_u^0)$ is a Paillier encryption of the identity of u with a random number r_u^0 using the RS' public key. RID_u^0 is sent to both user u and the verification server (VS). Since RID_u^0 is encrypted using the RS server's public key, only the RS server is able to decrypt it and reveal the real identity of the user. The actual identity of the user is always kept secret from the verification server during the lifetime of the user. After user u is registered, both the RS and VS servers store the user's initial randomized authentication ID $\mathrm{RID}_\mathrm{u}^0$ in their local databases DB_{rs} and DB_{vs} respectively. The plain texts of the real identities are discarded by the RS server to prevent attackers from hacking the system and stealing the sensitive information. The registration protocol is illustrated in Figure 7.2. Note that all messages are encrypted using the corresponding session keys between the communicating parties. For clarity, we only include the content of the messages in the figure.

*Here we use driver license number for illustration only. In practice, we can use more complex information to verify a user's identity to prevent an adversary from guessing.

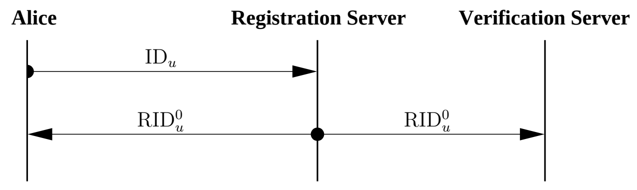


Figure 7.2: User Registration

At the end of the registration, a user creates an account with the RS server which can be authenticated via normal password or public key authentication system. This account will allow the user to communicate with RS for generating a new randomized ID. (See Section **??** for more details.)

7.2.2. User Authentication. Here we present the authentication protocol for a service provider Bob to verify if Alice is a legitimate user. The authentication protocol consists of three phases: (i) identity validation, (ii) ownership validation, and (iii) generation of randomized authentication ID.

(1) Identity Validation. Suppose Alice uses her randomized authentication ID $\mathrm{RID}^{\mathrm{i}}_{\mathrm{u}_1}$ to initiate the identity validation process with Bob. The following steps will be performed.

(a) Generating a shared random number

After establishing a secure channel, Alice executes the Diffie-Hellman key exchange protocol with Bob to mutually generate a shared random number $k_{u_1u_2}$. The protocol guarantees that the probability of other two users obtaining the same random number $k_{u_1u_2}$ is close to zero as long as one of the users follows the protocol. In other words, $k_{u_1u_2}$ is unique for each pair of users each time they execute the protocol. The use of this random number is to prevent the replay attack (discussed in Section 7.3).

(b) Setting up a pending request by Alice

Before sending $\mathrm{RID}^{\mathrm{i}}_{\mathrm{u}_1}$ to Bob, Alice will first register a pending authentication request at the VS server by sending the message: $p_{u_1} = [\mathrm{RID}^{\mathrm{i}}_{\mathrm{u}_1}, k_{u_1u_2}]$. The

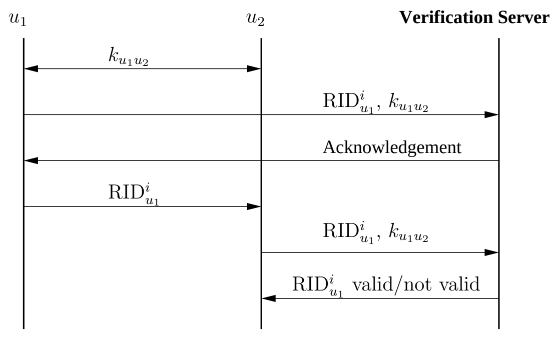
VS server will search its database to look for $\mathrm{RID}^{\mathrm{i}}_{\mathrm{u_1}}$. If $\mathrm{RID}^{\mathrm{i}}_{\mathrm{u_1}}$ exists, then the VS server will check if there is a duplication of this $\mathrm{RID}^{\mathrm{i}}_{\mathrm{u_1}}$ in the pending request table (to avoid the replay attack, described in Section 7.3). If there is no such randomized ID in the pending request table, the VS server will record this pending request. On the other hand, if it does not exist in the database or there is a duplication in the pending request table, the VS server will deny the authentication request.

(c) Exchanging the randomized ID

Upon receiving the acknowledgment of successful registration of the authentication request from the VS server, Alice sends $\mathrm{RID}^{\mathrm{i}}_{\mathrm{u_1}}$ to Bob.

(d) Verifying the randomized ID

For Bob to verify the received $\mathrm{RID}^{\mathrm{i}}_{\mathrm{u_1}}$, Bob forwards this randomized ID together with the random number $k_{u_1u_2}$ to the VS server. If the VS server finds a pending authentication request that matches the message sent by Bob, the VS server will inform Bob that this is a valid ID. Otherwise, the VS server will inform Bob that authentication fails.



u_1
u_2
Verification Server
$k_{u_1u_2}$
$\mathrm{RID}^i_{u_1}$, $k_{u_1u_2}$
Acknowledgement
$\mathrm{RID}^i_{u_1}$
$\mathrm{RID}^i_{u_1}$, $k_{u_1u_2}$
$\mathrm{RID}^i_{u_1}$ valid/not valid

Figure 7.3: Identity Validation

The main steps of identity validation are shown in Figure 7.3. In addition, Alice can perform concurrent authentication sessions with other service providers. For example, suppose Alice wants to contact with three other service providers (e.g., u_3, u_4 and u_5), Alice can ask RS to generate three new randomized authentication IDs (see Section **??** for technical details) and start three additional sessions with VS and continues with the rest of the steps discussed above with each user. Note that each session is established with a different session key generated between each pair of users at step (a) of the identity validation protocol.

(2) Ownership Validation. Once the service provider Bob confirms the validity of $\mathrm{RID}^{\mathrm{i}}_{\mathrm{u}_1}$, Bob may need to further verify whether Alice is the real owner of the randomized ID by checking if $\mathrm{RID}^{\mathrm{i}}_{\mathrm{u}_1}$ is currently stored at VS (i.e., has yet to be used by Alice). This step prevents an adversary to use $\mathrm{RID}^{\mathrm{i}}_{\mathrm{u}_1}$ to authenticate his or herself with other service providers to prevent common attacks such as the replay attack and the credential sharing problem (analyzed in Section 7.3). Note that this step is optional and not necessary if the user and the servers' systems are reasonably secure, the security of the communication channels cannot be tempered, or service providers are trustworthy.

(a) Generating a random challenge

Bob selects two random values c and r, and sends the following value v_1 to Alice.

$$v_1 = \left(\mathrm{RID}^{\mathrm{i}}_{\mathrm{u}_1}\right)^c * E(0, r) \tag{7.2}$$

where c is a challenge for Alice to solve, and r can be any random number just for performing the encryption of 0. The purpose of multiplying with $E(0, r)$ is to randomize $\left(\mathrm{RID}^{\mathrm{i}}_{\mathrm{u}_1}\right)^c$, so that it is computationally infeasible for an adversary to compute the discrete log of $\left(\mathrm{RID}^{\mathrm{i}}_{\mathrm{u}_1}\right)^c$ to obtain c (Section **??** provides detailed security analysis on this regard).

(b) Solving the random challenge

Only if Alice is the owner of the real identity, Alice will be able to compute the encrypted value of the challenge c. Specifically, Alice first encrypts the multiplicative inverse of her real identity $\mathrm{ID}_{u_1}^{-1}$ used during the registration phase. Then, Alice computes a value v_2 by computing $v_1^{\mathrm{ID}_{u_1}^{-1}}$. According to the additive homomorphic property of the underlying encryption scheme, value v_2 is equal to the encrypted value of c as deduced below:

$$\begin{aligned} v_2 &= v_1^{\mathrm{ID}_{u_1}^{-1}} \\ &= \left(\left(\mathrm{RID}^{\mathrm{i}}_{\mathrm{u_1}}\right)^c * E(0,r)\right)^{\mathrm{ID}_{u_1}^{-1}} \\ &= E(c * \mathrm{ID}_{u_1} * \mathrm{ID}_{u_1}^{-1}, r') \\ &= E(c, r') \end{aligned}$$

(c) Verifying Ownership

Then, Alice sends v_2 along with k_{u_1,u_2} to VS who will ask the RS to decrypt $v_2 = E(c, r')$ and obtain a decrypted value $D(E(c, r'))$. In addition, Bob needs to send c and k_{u_1,u_2} to VS in parallel. VS will forward the v_2 and c with the same k_{u_1,u_2} to RS. Then, RS will decrypt $v_2 = E(c, r')$ and get c. Since $E(c, r')$ or c does not contain any identity information about Alice, the RS server does not know whose identity that Bob is trying to verify. At last, RS will check if $D(E(c, r'))$ equals to c. If yes, RS informs VS that the validation succeeded, and no, otherwise. Then VS will notify Alice and Bob the corresponding validation result.

Figure 7.4 depicts the main messages exchanged during this validation phase. For the above scheme to work, $\mathrm{ID}_{\mathrm{u_1}}$ needs to have a multiplicative inverse in $\mathbb{Z}_N$. Since $N = pq$, and p and q are very large prime numbers, this requirement can be easily satisfied in our problem domain.

(3) Generation of Randomized Authentication ID. In our system, each randomized authentication ID is only used once so that a user social behaviors or locality patterns will not be tracked by any party. After Alice obtained services from Bob, Alice needs to acquire a new randomized authentication identity $\mathrm{RID}^{\mathrm{i}}_{\mathrm{u_1}}$ from the RS server for the next service request.

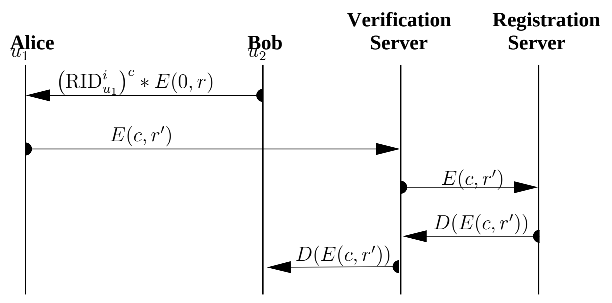


Figure 7.4: Ownership Validation

(a) Generating a new randomized ID

After Alice informs RS its intention to obtain a new randomized ID, the RS server can produce one based on the last randomized ID associated with Alice according to Equation 7.3. (As stated at the end of Section 7.2.1, a user creates a on-line account with the RS server at the end of the registration phase. This account links the user and his or her last randomized ID together. Thus, the user does not need to send his or her old randomized ID to generate a new one.)

$$\mathrm{RID^i_{u_1}} = \mathrm{RID^{i-1}_{u_1}} * E(0, r^i_{u_1}) \tag{7.3}$$

The $r^i_{u_1}$ value is randomly generated for this i^{th} request from u_1 (or Alice). Based on the addition property of the homomorphic encryption (Equation **??**), the new randomized ID is again the encryption of the real identity which can be deduced as:

$$\mathrm{RID^i_{u_1}} = E(\mathrm{ID_{u_1}}, r^{i-1}_{u_1}) * E(0, r^i_{u_1}) = E(\mathrm{ID_{u_1}} + 0, r')$$

It is worth mentioning that by leveraging this addition property, the generation of new randomized ID using the above Equation is more efficient and secure than directly encrypting the real identity again since there is no need for the RS server to keep the real identity after the registration phase. This is why the RS server should delete a user's real identity for added security protection, under the assumption that the private key is kept in a temper-proof storage such as a secure co-processor. As a consequence, even if the RS server is hacked, and private key is still safely protected from the attackers. Without the private key, the attackers cannot decrypt the randomized IDs to discover a user's real identity.

(b) Storing the new randomized ID
The RS server sends the newly generated randomized ID to both Alice and VS. When the VS server receives the new copy of randomized IDs, it will not be able to link each new ID to its previous version.

Since correlating a user's requests to location based services can reveal the user's real identity, it is important to note that the main purpose of generating a new set of randomized authentication IDs is to prevent VS and service providers from tracking to which services a user has been authenticated. Because each randomized ID is only used once, service requests cannot be linked to the same user. As a result, the user's real identity is protected.

Although RS knows the number of times a user has been authenticated, the RS server still cannot trace the locations of the user since the server does not know which service providers have been in contact with the user. Without the location information, the RS server cannot discover the real identity of the user when the user only use non-identifiable information in the user registration phase. (Recall that when a user registers with the RS server as discussed in Section 7.2.1, the user can submit either her real identity or some random information. The real identity is merely used for the purpose of traceability which is application-specific.)

7.2.3. Identity Tracing. In some applications, disputes may occur due to various reasons. Sometimes a third-party law enforcement authority may want to know immediately the real identity of a suspect user. Sometimes there may be a need to discover the authentication history of a suspect user. Thus, we propose both real-time identity tracing and historical identity tracing.

The real-time identity tracing is easy to achieve. The law enforcement authority submits the tracing request that contains the suspect user's randomized authentication ID to either the VS server or the RS server. If the request is received by the VS server, the VS server will forward the suspect user's randomized ID to the RS server. Upon receiving the suspect user's randomized ID, the RS server uses its private key to decrypt the randomized ID and reports the real identity to the law enforcement authority.

In terms of historical identity tracing, the law enforcement authority captured one randomized ID of the suspect user and wants to know the authentication history of the user to figure out the user's behavior in the network. The law enforcement authority sends the randomized ID of the suspect user to both RS and VS server. The RS server maintains a list of authentication history of all users. For example, each user has a list of randomized authentication IDs that have been or are planning to be used. The VS server maintains all valid authentication IDs and their targeted service providers or peer users like Bob in our example.

First, the RS server finds a match in a user's list. If there is a match, the list of randomized IDs will be provided to the law enforcement authority who will subsequently send these IDs to VS. The VS will return the authority the service providers who have provided services to the user with these randomized ID. Based on the location of the service providers, the authority may learn where the suspect has been before. To provide this kind of historical tracing, the only thing needs to be changed is that the RS and VS servers need more memory space to store previously used randomized IDs. In addition, when the VS server performs identity validation, it needs to make sure, old IDs cannot be used again. These modifications can be easily incorporated into our current scheme.

7.2.4. Credential or Identity Revocation. Identity revocation is very efficient in our system. Once a suspect user is confirmed to be malicious, the RS and the VS server just need to remove this user's randomization ID from their databases. Any subsequent authentication request for this malicious user will fail as no matching record will be found by the server any more.

7.3. SECURITY ANALYSIS

In this section, we analyze the properties of our proposed authentication system and discuss its robustness against various types of attacks The proposed authentication protocols have the following three properties: (i) unforgeability, (ii) full privacy preservation, and (iii) traceability. These properties can be analyzed based on the fact that the Paillier encryption scheme is semantically secure [2]. That is, for any $m \in \mathbb{Z}_N$, $c_1 = E(m, r_1)$ and $c_2 = E(m, r_2)$ are computationally indistinguishable (i.e., $c_1 \neq c_2$) if r_1 and r_2 are randomly chosen and N is sufficiently large (e.g., 1024-bit). Please note that even though $c_1 \neq c_2$, $D(c_1) = D(c_2) = m$ always holds because the randomness used in the encryption function is removed by the decryption function.

In addition, we also need to prove the challenge c issued by Bob in the ownership validation protocol cannot be discovered by a malicious user who does not possess the corresponding secret or the real identity. Our proof is based on the formal definition of computational indistinguishability.

Combining with the semantic security of the Paillier encryption function adopted in this paper, we are able to prove it is computationally infeasible for a malicious user to discover c. Next we will present this proof first.

(1) Discoverability of the Challenge in Ownership Validation. Equation 7.2 can be rewritten as $v = (E(x, r_1))^c * E(0, r_2)$. As discussed previously, since we do not know if it is hard to compute the discrete log of $(E(x, r_1))^c$ to discover c, the purpose of multiplying $(E(x, r_1))^c$ with $E(0, r_2)$ is to make the discovery of c from v computationally infeasible without knowing the secret value x. The following claim holds:

Claim 1 *Suppose the encryption key size is sufficient large (i.e., N is at least 1024-bit). Without knowing the secret value x, a malicious user is computationally infeasible to discover c from v where $v = (E(x, r_1))^c * E(0, r_2)$.*

Our proof follows from the semantic security property of E (the Paillier encryption function). Suppose the claim is not true, then there exists a probabilistic polynomially bounded algorithm A that can discover c from v. Now A can be used to distinguish the two cipher texts $v = (E(x, r_1))^c * E(0, r_2) = E(cx, r)$ and $v' = E(x, r')$ in polynomial time with hundred percent certainty. Thus, we can claim that v and v' are computationally distinguishable. On the other hand, since E is semantically secure [2], v and v' are computationally indistinguishable. This contradicts that A can distinguish between v and v'. Therefore, the assumption of the existence of A is invalid. We can conclude that the above claim is true and c cannot be discovered by a malicious user without knowing the secret value x. The proposed ownership validation protocol works correctly and securely.

(2) Unforgeability. Our authentication protocol guarantees that no one can use the randomized IDs that does not belong to him/her. Under the assumptions that the private key is kept securely at the RS server side, the only option left for the attacker to impersonate legitimate users is to exploit their randomized authentication IDs. There are several possible ways for an attacker to obtain a randomized authentication ID of a user. However, we show in the following that the attacker would not be able to use this ID as its own for authentication purpose.

(3) The Replay Attack. Although an attacker (e.g., a malicious service provider) can obtain another user's valid authentication ID during authentication, the attacker cannot directly use the received authentication ID using another parallel authentication session since the ID is currently in use, and the VS server can detect it during the step (b) of the identify validation phase.

If the attacker does not use the randomized ID in parallel with Alice, the attacker still cannot use it to perform user authentication successfully because each

ID is allowed to be used only once and is discarded after the use by the VS server. Even if the attacker tries to re-randomize the received ID using a new random number, the resulting ID will not match any valid authentication ID stored in the VS server. This is because the attacker does not know the randomization seed used by the real owner of the ID, and hence the attacker will not be able to generate the same series of randomized IDs that match the real ones.

(4) The Replay Attack with New Randomized IDs Previous discussion is focused on the used randomized IDs. We now discuss the case when an attacker steals new or never used randomized IDs from a user, the VS or RS server. Since these IDs have not been used by the real owner, the attacker will be able to go through the user authentication phase, but will be caught at the ownership validation phase. This is because the attacker does not know the real identity of the ID owner or the initial seed used in the user registration phase; hence, the attacker cannot discover the random number included in the challenge message according to Claim 1 proved previously.

(5) Full Privacy Preservation. Our authentication protocol provides full privacy preservation in that it guarantees both server-wise and peer-wise privacy for the users in terms of both anonymity and unlinkability. Considering the peer-wise privacy, under the proposed protocol, a user always self-generates a new randomized authentication ID when establishing a new communication session. Since the encryption scheme we adopted is semantically secure [2], it is computationally infeasible for peer users to know the real identity of others and to link different communication sessions or randomized authentication IDs to the same user as long as the size of the encryption key is large enough (such as 1024 bit).

As for the VS server, it does not have the secret key to decrypt the randomized IDs stored in its database, and hence it does not know the real identity of the user who submits authentication request (again here we assume the encryption key size is sufficiently large). Due to the fact that the IDs are randomized, VS cannot link different RIDs to the same user. As for the RS server, since it does not handle any authentication request that contains randomized IDs during the authentication phase, the RS server does not know which user or service provider is sending the authentication request. Therefore, our protocol prevents the RS server from tracking the locations of the users.

(6) Traceability. Traceability refers to the ability to reveal a user's real identity requested by the law authorities. This is a seemingly conflicting requirement with respect to the privacy preservation goal of our system. We achieve this by proposing the collaborative identity tracing protocol as presented in Section 7.2.3. The identity tracing protocol is capable of revealing a suspect user's real identity and his/her whole authentication history to the law authorities without violating the privacy of other legitimate users.

(7) Preventing Credential Sharing. The credential sharing problem under the proposed RAU system can be described as follows. Suppose a legitimate user Alice possesses a valid randomized authentication ID $\mathrm{RID}_{\mathrm{u}_1}^{\mathrm{i}}$ and Bob is not a legitimate user (e.g., Bob did not pay service fees). Credential sharing means that Alice gives $\mathrm{RID}_{\mathrm{u}_1}^{\mathrm{i}}$ to Bob so that Bob can obtain the needed service using $\mathrm{RID}_{\mathrm{u}_1}^{\mathrm{i}}$.

To prevent credential sharing, the RAU system needs to require that Alice provides her real identify or personal identifiable information, such as social security number (SSN), during user registration phase (discussed in Section 7.2.1). Here we assume that the personal information (denoted by x) for Alice is extremely important and cannot be shared with other users. Under this assumption, even if $\mathrm{RID}_{\mathrm{u}_1}^{\mathrm{i}}$ is given to another user Carl, Carl cannot pass the ownership validation phase without knowing x. As a consequence, credential sharing problem can be easily solved under the RAU system assuming that Alice does not share her personal identifiable information x.

(8) Different Configurations of RAU. We have discussed all the features and functionalities of the RAU system. To be more clear, here we summarize the

choices that need to be made when setting up the RAU system to achieve the desired features. The choices are related to non-identifiable or identifiable information for user registration and whether or not to perform ownership validation.

- Identifiable vs. non-identifiable information

  As stated in Section 7.2.1, a user can provide either identifiable or non-identifiable information during user registration phase. A user needs to provide identifiable personal information such as SSN to achieve

  (1) *traceability*

  (2) *prevention of credential sharing*

- Ownership validation

  As acknowledged in the previous section, the ownership validation phase is optional. The ownership validation is required if we want to eliminate

  (1) *the replay attack*

  (2) *the credential sharing problem*

7.4. EXPERIMENTAL STUDY

Please note that our protocol does not require the users be equipped with high performance computing equipment. The following hardware specification is used to simulate the servers, but not the hardware carried by a mobile user. We implemented the RAU authentication protocol in C language with GMP library, and run the tests on a PC with Intel Xeon CPU X5675 @3.07GHZ x6 and 12GB memory. We evaluate the efficiency of the total authentication process in terms of communication and computation costs. We did not include the transmission and propagation delays since they depend on specific network configurations.

(1) User Registration. The main computation cost involved in user registration is the generation of the initial randomized ID for the new user. Each

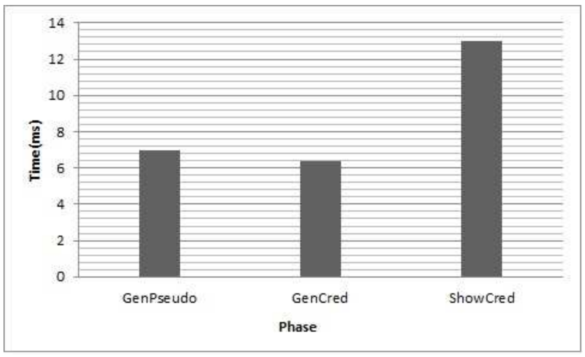


Figure 7.5: Running time of AC protocol

randomized ID is 2048 bits. The communication complexity of this phase is 4 steps:

- A user sends his or her real identity to RS,
- RS sends the received identity to background check authority,
- The authority sends back the verification result to RS,
- RS generates the initial random ID for the user.

The message complexity is $4l$ ($l = 1024$) where l denotes the encryption key size. The computation complexity is 1 exponentiation. In the experiment, we observed that the randomized ID generation time is less than 1.9ms per user.

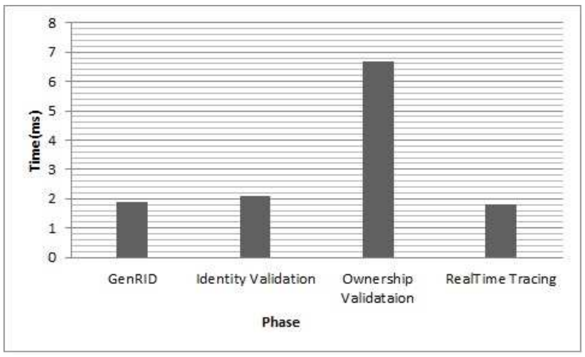


Figure 7.6: Running Time of RAU protocol

(2) User Authentication. Recall that the user authentication protocol consists of three phases: identity validation, ownership validation and randomized ID generation. Note that not every phase is required for each user authentication. The ownership validation phase is optional. Thus, the efficiency of the identity validation phase is important. As reported later in this section, all three phase combined take about 11ms. In addition, the average latency for a 4G network (e.g., Verizon) is 30ms. Identity tracing is very efficient, and it only requires several encryption operations.

The AC protocol [63] uses zero-knowledge proof to achieve privacy-preserving user authentication. Being the best among the exiting solutions, that scheme achieves most criteria for anonymous authentication defined in this paper. Here we compare its efficiency with our protocol. In order to have the same security guarantee as our proposed RAU protocol, we need to include the running time for both credential issuing and verification phases. We implemented the AC protocol under the same computing environment as RAU.

The running time of each phase of both protocols are given in Figure 7.5 and Figure 7.6. (The running time of historical tracing in our protocol is not shown here since it is about 3490ms, which is hard to present clearly on the figure with other phases.) The running time for both protocols are also compared in Table 7.3. According to the table, 10.7ms running time includes all three phases of RAU. RAU$^\alpha$ represents the situation where the ownership validation phase is omitted. RAU$^\beta$ represents the situation where the request for and the generation of new randomized ID, denoted by RID_{new}, are performed offline. As we can see, the proposed protocol is considerably more efficient than the AC protocol.

The communication complexity of showing credential to verifier in the AC protocol using zero-knowledge proof contains 5 steps:

1. a user sends an initial request to the verifier

Table 7.3: Running time

Protocol	Time
AC	13ms
RAU	10.7 ms
RAU$^\alpha$	4 ms
RAU$^\beta$	2.1ms

2. the user sends his or her anonymous credential to the verifier (if necessary, the user should wait for the acknowledgment of the initial request from the verifier which adds one more step)

3. the verifier sends a challenge to the user

4. the user sends the response of the challenge to the verifier

5. the verifier sends an acknowledgment to the user.

The message complexity is $14l$ bits. The computation complexity of showing credential is 9 exponentiations.

Our scheme improves both communication and computation complexities. The communication complexity of our RAU scheme contains 6 steps:

1. Alice and Bob mutually generate a session key

2. Alice sends a pending authentication request to VS

3. Alice sends an authentication request to Bob (if necessary, Alice should wait for the acknowledgment from VS before communicating with Bob which adds one more step)

4. Bob sends the authentication message to VS

5. Alice requests for a new randomized ID, RID_{new}

6. RS sends RID_{new} to Alice, and in parallel, RS sends RID_{new} to VS.

Table 7.4: Complexity RAU vs. AC

Protocol	Rounds	Message size	exponentiations
AC	5	$14l$	9
RAU$^\alpha$	6	$9l + 512$	1
RAU$^\beta$	4	$7l + 512$	1

The message complexity is $9l + 512$ bits, and the computation only takes one exponentiation. The last two steps can be done offline which makes our protocol more efficient. Table 7.4 summarizes the results.

Our scheme also solves the credential sharing problem which is the major concern in the AC protocol. As described in Section 7.3, each RID can only be used once, so an old RID is useless for a malicious user. On the other hand, when a new RID is stolen and used, the malicious user can be caught by executing the ownership validation phase.

Please note that for the ownership validation phase, the communication complexity is 5 steps with $8l$ bits message complexity, and the computation cost is 2 exponentiations. Without the ownership validation phase, our protocol offers the same functionality as the AC protocol. Thus, to have a fair comparison, Table 7.4 does not include the complexity of the ownership validation phase.

8. CONCLUSIONS

We provides an efficient constant rounds solution for the bit-decomposition problem based on secure arithmetic computation with improved rounds and multiplication complexity compared to the previous solutions. The complexity reduced to $3l$ (constant factor is reduced) secure multiplications. Unlike Bits-RT protocol, there is no need of the LSB gate which is statistical secure in the comparison primitive and result in statistical security of post-fix comparison primitive and hence, our protocol achieves more secure. In addition, we analyze the communication complexity of the proposed protocol with that protocol. We emphasize that, apart from the computation complexity gain, the communication message size is reduced by using smaller share size. In this thesis, we proposed several efficient protocols of cryptographic primitives to implement the computation over the encrypted data, including:

- secure comparison (CMP)
- secure binary decomposition (SBD)
- proxy re-encryption (PRE)

We also construct efficient and feature-rich unidirectional proxy re-encryption schemes with simple design (no pairings involved as the existing works) against multiple adversary models. We adopt the strongly unforgeable signature technique to implement the validation of re-encrypted ciphertext. In our proposed constructions, the delegatee Bob has to submit request of the secret to proxy if Bob desires to access the data, and the proxy does not perform any computation on the ciphertexts to response the requests. This is the main difference form the existing PRE schemes where the proxy is required to perform re-encryption on the ciphertext when the delegatee requests to access data. We measure our implementation of PRE in a secure file system which uses an untrusted access control server to manage accesses to encrypted files stored on a distributed storage.

BIBLIOGRAPHY

[1] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[2] P. Paillier. Public key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology - Eurocrypt '99 Proceedings, LNCS 1592*, pages 223–238, Prague, Czech Republic, 1999. Springer-Verlag.

[3] Ivan Damgård, Matthias Fitzi, Eike Kiltz, Jesper Nielsen, and Tomas Toft. Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. *Theory of Cryptography*, pages 285–304, 2006.

[4] Tord Reistad and Tomas Toft. Linear, constant-rounds bit-decomposition. In *Information, Security and Cryptology–ICISC 2009*, pages 245–257. Springer, 2010.

[5] Matt Blaze and Martin Strauss. Atomic proxy cryptography. In *Proc. EuroCrypt'97*. Citeseer, 1998.

[6] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In *Advances in CryptologyEUROCRYPT'98*, pages 127–144. Springer, 1998.

[7] Takashi Nishide and Kazuo Ohta. Multiparty computation for interval, equality, and comparison without bit-decomposition protocol. In *Public Key Cryptography–PKC 2007*, pages 343–360. Springer, 2007.

[8] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. In *In Internet Society (ISOC)*, pages 29–43. NDSS 2005, 2005.

[9] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security (TISSEC)*, 9(1):1–30, 2006.

[10] Juan Garay, Berry Schoenmakers, and José Villegas. Practical and secure solutions for integer comparison. In *Public Key Cryptography–PKC 2007*, pages 330–342. Springer, 2007.

[11] Ian F Blake and Vladimir Kolesnikov. Strong conditional oblivious transfer and computing on intervals. In *Advances in Cryptology-ASIACRYPT 2004*, pages 515–529. Springer, 2004.

[12] Ivan Damgård, Martin Geisler, and Mikkel Krøigaard. Efficient and secure comparison for on-line auctions. In *Information Security and Privacy*, pages 416–430. Springer, 2007.

[13] Ivan Damgard, Martin Geisler, and Mikkel Kroigard. A correction to'efficient and secure comparison for on-line auctions'. *International Journal of Applied Cryptography*, 1(4):323–324, 2009.

[14] Vladimir Kolesnikov, Ahmad-Reza Sadeghi, and Thomas Schneider. Improved garbled circuit building blocks and applications to auctions and computing minima. In *Cryptology and Network Security*, pages 1–20. Springer, 2009.

[15] Ian F Blake and Vladimir Kolesnikovs. One-round secure comparison of integers. *Journal of Mathematical Cryptology*, 3(1):37–68, 2009.

[16] Tomas Toft. Constant-rounds, almost-linear bit-decomposition of secret shared values. In *Topics in Cryptology–CT-RSA 2009*, pages 357–371. Springer, 2009.

[17] Tord Ingolf Reistad and Tomas Toft. Secret sharing comparison by transformation and rotation. In *Information Theoretic Security*, pages 169–180. Springer, 2009.

[18] Tord Reistad. Multiparty comparison - an improved multiparty protocol for comparison of secret-shared values. pages 325–330, 2009.

[19] Dan Bogdanov. how to securely perform computations on secret shared data. *Master's thesis, University of Tartu*, 2007.

[20] Berry Schoenmakers and Pim Tuyls. Efficient binary conversion for paillier encrypted values. In *Advances in Cryptology-EUROCRYPT 2006*, pages 522–537. Springer, 2006.

[21] Berry Schoenmakers and Pim Tuyls. Practical two-party computation based on the conditional gate. *Advances in Cryptology-ASIACRYPT 2004*, pages 129–145, 2004.

[22] Yan Huang, Jonathan Katz, and Dave Evans. Efficient secure two-party computation using symmetric cut-and-choose. In *Advances in Cryptology - CRYPTO'2013*, August 2013.

[23] Payman Mohassel and Ben Riva. Garbled circuits checking garbled circuits: More efficient and secure two-party computation. In *Advances in Cryptology - CRYPTO'2013*, August 2013.

[24] Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. In *Advances in Cryptology - CRYPTO'2012*, August 2012.

[25] Andreas Holzer, Martin Franz, Stefan Katzenbeisser, and Helmut Veith. Secure two-party computations in ansi c. In *The 19th ACM Conference on Computer and Communications Security*, October 2012.

[26] S. Dov Gordon, Jonathan Katz, Vladimir Kolesnikov, Fernando Krell, Tal Malkin, Mariana Raykova, and Yevgeniy Vahlis. Secure two-party computation in sublinear (amortized) time. In *The 19th ACM Conference on Computer and Communications Security*, October 2012.

[27] Wilko Henecka, Stefan Koegl, Ahmad-Reza Sadeghi, Thomas Schneider, and Immo Wehrenberg. Tasty: Tool for automating secure two-party computations. In *The 17th ACM Conference on Computer and Communications Security*, October 2010.

[28] Rosario Gennaro, Michael O Rabin, and Tal Rabin. Simplified vss and fast-track multiparty computations with applications to threshold cryptography. In *Proceedings of the seventeenth annual ACM symposium on Principles of distributed computing*, pages 101–111. ACM, 1998.

[29] Chao Ning and Qiuliang Xu. Multiparty computation for modulo reduction without bit-decomposition and a generalization to bit-decomposition. In *Advances in Cryptology-ASIACRYPT 2010*, pages 483–500. Springer, 2010.

[30] Yan Huang, David Evans, Jonathan Katz, and Lior Malka. Faster secure two-party computation using garbled circuits. In *The 20th USENIX Security Symposium*, August 2011.

[31] Sanjam Garg and Amit Sahai. Adaptively secure multi-party computation with dishonest majority. In *Advances in Cryptology - CRYPTO'2012*, August 2012.

[32] S. Dov Gordon, Tal Malkin, Mike Rosulek, and Hoeteck Wee. Multi-party computation of polynomials and branching programs without simultaneous interaction. In *Advances in Cryptology - EUROCRYPT'2013*, May 2013.

[33] Sanjam Garg, Vipul Goyal, Abhishek Jain, and Amit Sahai. Concurrently secure computation in constant rounds. In *Advances in Cryptology - EUROCRYPT'2012*, April 2012.

[34] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In *Advances in Cryptology–EUROCRYPT 2011*, pages 169–188. Springer, 2011.

[35] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Advances in Cryptology–CRYPTO 2012*, pages 643–662. Springer, 2012.

[36] Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P. Smart. Practical covertly secure mpc for dishonest majority - or: Breaking the spdz limits. In *ESORICS*, pages 1–18, 2013.

[37] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 6110 of *Lecture Notes in Computer Science*, pages 24–43. Springer, 2010.

[38] Masahiro Mambo and Eiji Okamoto. Proxy cryptosystems: Delegation of the power to decrypt ciphertexts. *IEICE transactions on fundamentals of electronics, Communications and computer sciences*, 80(1):54–63, 1997.

[39] Anca-Andreea Ivan and Yevgeniy Dodis. Proxy cryptography revisited. In *NDSS*, 2003.

[40] Markus Jakobsson. On quorum controlled asymmetric proxy re-encryption. In *Public Key Cryptography*, pages 112–121. Springer, 1999.

[41] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology*, pages 10–18. Springer, 1985.

[42] Ran Canetti and Susan Hohenberger. Chosen-ciphertext secure proxy re-encryption. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 185–194. ACM, 2007.

[43] Benoît Libert and Damien Vergnaud. Tracing malicious proxies in proxy re-encryption. In *Pairing-Based Cryptography–Pairing 2008*, pages 332–353. Springer, 2008.

[44] Ran Canetti, Hugo Krawczyk, and Jesper B Nielsen. Relaxing chosen-ciphertext security. In *Advances in Cryptology-CRYPTO 2003*, pages 565–582. Springer, 2003.

[45] Manoj Prabhakaran and Mike Rosulek. Rerandomizable rcca encryption. In *Advances in Cryptology-CRYPTO 2007*, pages 517–534. Springer, 2007.

[46] Robert H Deng, Jian Weng, Shengli Liu, and Kefei Chen. Chosen-ciphertext secure proxy re-encryption without pairings. In *Cryptology and Network Security*, pages 1–17. Springer, 2008.

[47] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology-Crypto89 Proceedings*, pages 239–252. Springer, 1990.

[48] Jun Shao and Zhenfu Cao. Cca-secure proxy re-encryption without pairings. In *Public Key Cryptography–PKC 2009*, pages 357–376. Springer, 2009.

[49] Sherman SM Chow, Jian Weng, Yanjiang Yang, and Robert H Deng. Efficient unidirectional proxy re-encryption. In *Progress in Cryptology–AFRICACRYPT 2010*, pages 316–332. Springer, 2010.

[50] Benoit Libert and Damien Vergnaud. Unidirectional chosen-ciphertext secure proxy re-encryption. *Information Theory, IEEE Transactions on*, 57(3):1786–1802, 2011.

[51] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *Advances in Cryptology-Eurocrypt 2004*, pages 506–522. Springer, 2004.

[52] Brent R Waters, Dirk Balfanz, Glenn Durfee, and Diana K Smetters. Building an encrypted and searchable audit log. In *NDSS*, volume 4, pages 5–6, 2004.

[53] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. In *Advances in Cryptology–CRYPTO 2005*, pages 205–222. Springer, 2005.

[54] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. On the integration of public key data encryption and public key encryption with keyword search. In *Information Security*, pages 217–232. Springer, 2006.

[55] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. Public key encryption with keyword search revisited. In *Computational Science and Its Applications–ICCSA 2008*, pages 1249–1259. Springer, 2008.

[56] Giovanni Di Crescenzo and Vishal Saraswat. Public key encryption with searchable keywords based on jacobi symbols. In *Progress in Cryptology–INDOCRYPT 2007*, pages 282–296. Springer, 2007.

[57] Chunxiang Gu, Yuefei Zhu, and Heng Pan. Efficient public key encryption with keyword search schemes from pairings. In *Information Security and Cryptology*, pages 372–383. Springer, 2008.

[58] Hyun Sook Rhee, Jong Hwan Park, Willy Susilo, and Dong Hoon Lee. Improved searchable public key encryption with designated tester. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, pages 376–379. ACM, 2009.

[59] Hyun Sook Rhee, Jong Hwan Park, Willy Susilo, and Dong Hoon Lee. Trapdoor security in a searchable public-key encryption scheme with a designated tester. *Journal of Systems and Software*, 83(5):763–771, 2010.

[60] Dawn Xiaoding Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on*, pages 44–55. IEEE, 2000.

[61] Jun Shao, Zhenfu Cao, Xiaohui Liang, and Huang Lin. Proxy re-encryption with keyword search. *Information Sciences*, 180(13):2576–2587, 2010.

[62] Wei-Chuen Yau, Raphael C-W Phan, Swee-Huay Heng, and Bok-Min Goi. Proxy re-encryption with keyword search: new definitions and algorithms. In *Security Technology, Disaster Recovery and Business Continuity*, pages 149–160. Springer, 2010.

[63] Jan Camenisch and Els Van Herreweghen. Design and implementation of the *idemix* anonymous credential system. In *ACM Conference on Computer and Communications Security*, pages 21–30, 2002.

[64] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT*, pages 93–118, 2001.

[65] Ernie Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In *Proceedings of the 11th ACM conference on Computer and communications security*, pages 132–145. ACM, 2004.

[66] Emanuele Cesena, Hans Löhr, Gianluca Ramunno, Ahmad-Reza Sadeghi, and Davide Vernizzi. Anonymous authentication with tls and daa. In *Trust and Trustworthy Computing*, pages 47–62. Springer, 2010.

[67] Christian Wachsmann, Liqun Chen, Kurt Dietrich, Hans Löhr, Ahmad-Reza Sadeghi, and Johannes Winter. Lightweight anonymous authentication with tls and daa for embedded mobile devices. In *Information Security*, pages 84–98. Springer, 2011.

[68] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Advances in CryptologyCRYPTO 2002*, pages 61–76. Springer, 2002.

[69] Jan Camenisch and Els Van Herreweghen. Design and implementation of the idemix anonymous credential system. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 21–30. ACM, 2002.

[70] Jan Camenisch and Thomas Groß. Efficient attributes for anonymous credentials. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 345–356. ACM, 2008.

[71] M. Raya and J. P. Hubaux. Securing vehicular ad hoc networks. In *Journal of Computer Security*, pages 39–68, 2007.

[72] Maxim Raya, Daniel Jungels, Panos Papadimitratos, Imad Aad, and Jean-Pierre Hubaux. Certificate revocation in vehicular networks. *Laboratory for computer Communications and Applications (LCA) School of Computer and Communication Sciences, EPFL, Switzerland*, 2006.

[73] Xiaoting Sun. Anonymous, secure and efficient vehicular communications. Master's thesis, The University of Waterloo, Waterloo, Ontario, Canada, 2007.

[74] Giorgio Calandriello, Panos Papadimitratos, Jean-Pierre Hubaux, and Antonio Lioy. Efficient and robust pseudonymous authentication in vanet. In *Proceedings of the fourth ACM international workshop on Vehicular ad hoc networks*, pages 19–28. ACM, 2007.

[75] Yipin Sun, Rongxing Lu, Xiaodong Lin, Xuemin Shen, and Jinshu Su. An efficient pseudonymous authentication scheme with strong privacy preservation for vehicular communications. *Vehicular Technology, IEEE Transactions on*, 59(7):3589–3603, 2010.

[76] G.Calandriello, P. Papadimitratos, J.P. Hubaux, and A. Lio. Efficient and robust pseudonymous authentication in vanet. In *Proc. of the 4th ACM international workshop on Vehicular ad hoc networks*, pages 19–28, 2007.

[77] A. Studer, E. Shi, F. Bai, and A. Perrig. Tacking together efficient authentication, revocation, and privacy in VANETs. In *Proceedings of the 6th Annual IEEE communications society conference on Sensor, Mesh and Ad Hoc Communications and Networks*, SECON'09, pages 484–492, Piscataway, NJ, USA, 2009. IEEE Press.

[78] J. Sun, C. Zhang, Y. Zhang, and Y. Fang. An identity-based security system for user privacy in vehicular ad hoc networks. *IEEE Transactions on Parallel and Distributed Systems*, 21(9):1227–1239, 2010.

[79] K.-A. Shim. Cpas: An efficient conditional privacy-preserving authentication scheme for vehicular sensor networks. In *IEEE Transaction on Vehicular Technology*, pages 1874–1883, 2012.

[80] C. Zhang, P.-H. Ho, and J. Tapolcai. On batch verification with group testing for vehicular communications. *Wireless Networks*, 17(8):1851–1865, 2011.

[81] J. Zhang, Y. Cui, and Z. Chen. Spa: Self-certified pkc-based privacy-preserving authentication protocol for vehicular ad hoc networks.

[82] C. Zhang, R. Lu, X. Lin, P.H. Ho, and X. Shen. An efficient identity-based batch verification scheme for vehicular sensor networks. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 246–250. IEEE, 2008.

[83] K. Sha, Y. Xi, W. Shi, L. Schwiebert, and T. Zhang. Adaptive privacy-preserving authentication in vehicular networks. In *Communications and Networking in China, 2006. ChinaCom'06. First International Conference on*, pages 1–8. IEEE, 2006.

[84] R. Lu, X. Lin, H. Zhu, P.-H. Ho, and X. Shen. Ecpp:efficient conditional privacy preservation protocol for secure vehicular communications. In *Proc. of IEEE Conference on Computer Communications*, pages 1229 – 1237, 2008.

[85] C. D. Jung, C. Sur, Y. Park, and K.-H. Rhee. A robust conditional privacy-preserving authentication protocol in vanet. *Social Informatics and Telecommunications Engineering*, 17:35–45, 2009.

[86] D. Chaum and E. V. Heijst. Group signatures. In *Advanced CryptologyEurocryptS*, pages 257–265, 1991.

[87] X. Lin, X. Sun, P.-H. Ho, and X. Shen. Gsis: A secure and privacy-preserving protocol for vehicular communications. *IEEE Transactions on Vehicular Technology*, 56(6):3442 – 3456, 2007.

[88] Y. Hao, C. Yu, C. Zhou, and W. Song. A distributed key management framework with cooperative message authentication in vanets. *Selected Areas in Communications, IEEE Journal on*, 29(3):616–629, 2011.

[89] A. Squicciarini, D. Lin, and A. Mancarella. Paim: Peer-based automobile identity management in vehicular ad-hoc network. In *Proc. of the IEEE Computer Software and Applications Conference (COMPSAC)*, 2011.

[90] L.Y. Yeh, Y.C. Chen, and J.L. Huang. Paacp: A portable privacy-preserving authentication and access control protocol in vehicular ad hoc networks. *Computer Communications*, 34(3):447–456, 2011.

[91] Z. Tan. A lightweight conditional privacy-preserving authentication and access control scheme for pervasive computing environments. *Journal of Network and Computer Applications*, 2012.

[92] Z. Tan. A privacy-preserving mutual authentication protocol for vehicle ad hoc net. *Journal of Convergence Information Technology*, 5(7), 2010.

[93] C. Gamage, B. Gras, B. Crispo, and A.S. Tanenbaum. An identity-based ring signature scheme with enhanced privacy. In *Securecomm and Workshops*, pages 1–5, 2006.

[94] Tomasz Ciszkowski and Zbigniew Kotulski. Anap: Anonymous authentication protocol in mobile ad hoc networks. *arXiv preprint cs/0609016*, 2006.

[95] RIFA-POUS Helena, Emmanouil A Panaousis, and Christos Politis. Recipients' anonymity in multihop ad-hoc networks. *IEICE TRANSACTIONS on Information and Systems*, 95(1):181–184, 2012.

[96] Chao-Chin Chou, David SL Wei, C-CJ Kuo, and Kshirasagar Naik. An efficient anonymous communication protocol for peer-to-peer applications over mobile ad-hoc networks. *Selected Areas in Communications, IEEE Journal*, 25(1):192–203, 2007.

[97] Julien Freudiger, Maxim Raya, and Jean-Pierre Hubaux. Self-organized anonymous authentication in mobile ad hoc networks. In *Security and Privacy in Communication Networks*, pages 350–372. Springer, 2009.

[98] Jian Ren and Lein Harn. An efficient threshold anonymous authentication scheme for privacy-preserving communications. 2013.

[99] Jia-Lun Tsai, Nai-Wei Lo, and Tzong-Chen Wu. Secure anonymous authentication protocol with unlinkability for mobile wireless environment. In *Anti-Counterfeiting, Security and Identification (ASID), 2012*, pages 1–5. IEEE, 2012.

[100] Panayiotis Kotzanikolaou, Emmanouil Magkos, Nikolaos Petrakos, Christos Douligeris, and Vassilis Chrissikopoulos. Fair anonymous authentication for location based services. In *Data Privacy Management and Autonomous Spontaneous Security*, pages 1–14. Springer, 2013.

[101] Randy Baden, Adam Bender, Neil Spring, Bobby Bhattacharjee, and Daniel Starin. Persona: an online social network with user-defined privacy. In *ACM SIGCOMM Computer Communication Review*, volume 39, pages 135–146. ACM, 2009.

[102] Andrew C Yao. Protocols for secure computations. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 160–164. IEEE, 1982.

[103] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *In SFCS*, pages 162–167. IEEE Computer Society, 1986.

[104] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 218–229. ACM, 1987.

[105] Donald Beaver. Foundations of secure interactive computing. In *Advances in CryptologyCRYPTO91*, pages 377–391. Springer, 1992.

[106] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[107] Ronald Cramer, Ivan Damgård, and Jesper B Nielsen. *Multiparty computation from threshold homomorphic encryption*. Springer, 2001.

[108] Yehuda Lindell. General composition and universal composability in secure multiparty computation. *Journal of cryptology*, 22(3):395–428, 2009.

[109] Assaf Ben-David, Noam Nisan, and Benny Pinkas. Fairplaymp: a system for secure multi-party computation. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 257–266. ACM, 2008.

[110] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography: principles and protocols*. CRC Press, 2007.

[111] Yonatan Aumann and Yehuda Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. *Journal of Cryptology*, 23(2):281–343, 2010.

[112] Oded Goldreich. *Foundations of Cryptography: Volume 2, chapter Encryption Schemes*, volume 2. Cambridge university press, 2004.

[113] Wenliang Du and Mikhail J Atallah. Secure remote database access with approximate matching. In *In First Workshop on E-Commerce Security and Privacy*, 2000.

[114] Dan Bogdanov, Sven Laur, and Jan Willemson. Sharemind: A framework for fast privacy-preserving computations. In *Computer Security-ESORICS*, pages 192–206. Springer, 2008.

[115] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in cryptologyEUROCRYPT99*, pages 223–238. Springer, 1999.

[116] Hu Chun, Yousef Elmehdwi, Feng Li, Prabir Bhattacharya, and Wei Jiang. Outsourceable two-party privacy-preserving biometric authentication. In *Proceedings of the 9th ACM symposium on Information, computer and communications security*, pages 401–412. ACM, 2014.

[117] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.

[118] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*, volume 2. Cambridge university press, 2009.

[119] Tobias Volkhausen. Paillier cryptosystem: A mathematical introduction. `http://www2.cs.uni-paderborn.de/cs/ag-bloemer/lehre/proseminar_WS2005/mate` 2006. Online: accessed 10-May-2006.

[120] Michael OKeeffe. The paillier cryptosystem. *A Look Into The Cryptosystem And Its Potential Application, college of New Jersey*, 2008.

[121] Jee Hea An, Yevgeniy Dodis, and Tal Rabin. On the security of joint signature and encryption. In *Advances in CryptologyEUROCRYPT 2002*, pages 83–107. Springer, 2002.

[122] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *Advances in Cryptology-Eurocrypt 2004*, pages 207–222. Springer, 2004.

[123] Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. *SIAM Journal on Computing*, 36(5):1301–1328, 2006.

[124] Dan Boneh, Emily Shen, and Brent Waters. Strongly unforgeable signatures based on computational diffie-hellman. In *Public Key Cryptography-PKC 2006*, pages 229–240. Springer, 2006.

[125] Yan Huang, David Evans, Jonathan Katz, and Lior Malka. Faster secure two-party computation using garbled circuits. In *USENIX Security Symposium*, volume 201, 2011.

[126] Lidong Zhou, Michael A Marsh, Fred B Schneider, and Anna Redz. Distributed blinding for distributed elgamal re-encryption. In *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference on*, pages 824–824. IEEE, 2005.

[127] Benoît Libert and Damien Vergnaud. Multi-use unidirectional proxy re-signatures. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 511–520. ACM, 2008.

[128] Xiaohui Liang, Zhenfu Cao, Huang Lin, and Jun Shao. Attribute based proxy re-encryption with delegating capabilities. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, pages 276–286. ACM, 2009.

[129] Matthew Green, Susan Hohenberger, and Brent Waters. Outsourcing the decryption of abe ciphertexts. In *USENIX Security Symposium*, page 3, 2011.

[130] Whit Diffie and Martin Hellman. New directions in cryptography. *IEEE*, IT-22(6):644–654, 1976.

VITA

Feng Li was born in China. He received his MS degree in College of Computer and Communication Engineering from China University of Petroleum (East China) in 2011. He joined Missouri University of Science and Technology (formerly the University of Missouri-Rolla) in August, 2011, and received the degree of Doctor of Philosophy in Computer Science in August, 2015.