2013

On remote data integrity checking of the cloud storage

Xinyu Fan

University of Wollongong

Recommended Citation
UNIVERSITY OF WOLLONGONG

COPYRIGHT WARNING

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site. You are reminded of the following:

Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material. Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.
On Remote Data Integrity Checking of the Cloud Storage

A thesis submitted in fulfillment of the requirements for the award of the degree

Master of Computer Science (by Research)

from

UNIVERSITY OF WOLLONGONG

by

Xinyu Fan

School of Computer Science and Software Engineering
December 2013
Dedicated to

my parents ... 
with love and gratitude
Declaration

This is to certify that the work reported in this thesis was done by the author, unless specified otherwise, and that no part of it has been submitted in a thesis to any other university or similar institution.

________________________________________________________
Xinyu Fan
December 2, 2013
Cloud computing offers different types of computational services to end users via computer networks. Nowadays it has become a trend that individuals and IT enterprises store data remotely to the cloud in a flexible on-demand manner, which has become a popular way of data outsourcing. This has reduced the burden for storage management and maintenances and costs on hardware and software, with great advancement of universal data access and convenience to users. In fact, cloud storage has become one of the major services in cloud computing where user data are stored and maintained by cloud servers. It allows users to access their data via computer networks at anytime and from anywhere.

Despite the great benefits provided by cloud computing, data security is a very important yet challenging problem that must be solved. One of the major concerns of data security is data integrity in a remote storage system. Although storing data in the cloud is attractive, it does not usually offer any guarantee on data integrity and retrievability.

Unfortunately, many Remote Integrity Checking (RIC) schemes in the literature are insecure. In this thesis, we will provide a cryptanalysis against a well-known RIC scheme. Our analysis approach can also be applied to other similar RIC schemes. We also provide a solution to the problem.

It is also very important that an auditing process should not introduce new vulnerabilities of unauthorized information leakage towards their data security. The previous efforts in RIC accommodate several security features including data integrity and confidentiality, which mainly ensure secure maintenance of data. However, they do not cover the issue of data privacy, which means that the communication flows (RIC proofs) from the cloud server should not reveal any useful information to the adversary. Intuitively, by “privacy”, we mean that an adversary should not be able to distinguish which file has been uploaded by the client and maintained
by the cloud server. We refer it as *Indistinguishability* (or IND, for short). We believe that it is very important to consider such privacy issues adequately in protocol designs. We refer to this security property as IND-privacy.

In this thesis, we also provide the definition of data privacy for RIC protocols and demonstrate how data privacy can be achieved. We demonstrate that a well-known privacy-preserving RIC protocol do not provide IND-Privacy. Actually, we can conclude that all current RIC protocols do not provide IND-privacy. We also show that with a witness distinguishability proof, we are able to achieve IND-Privacy in RIC. As a instantiation, we present a concrete RIC protocol, which capture the security property of IND-privacy.
Acknowledgement

First of all, I am very grateful to my supervisor, Dr Guomin Yang, who gave me this opportunity to study cryptography. He was always very patient explaining the concepts and guiding me on the right way during my research. I would like to express my gratefulness to my co-supervisor, Prof. Yi Mu, who provided me with his guidance to improve my knowledge in cryptography and problem solving skills. I would also like to thank Dr Yong Yu for his invaluable discussions.

My great gratitude also goes to my parents who always supported my study. Especially, when I encountered problems, their encouragement always made me move forward.

I want to thank the administrative staff and the technical support staff of School of Computer Science and Software Engineering, for providing the good study environment and professional supports.
Contents

Abstract v
Acknowledgement vii
Publications viii

1 Introduction 1
  1.1 Background ........................................... 1
  1.2 Previous Solutions ................................ 2
  1.3 Motivation and Contributions ..................... 4
  1.4 Organisation of This Thesis ....................... 6

2 Preliminaries 7
  2.1 Foundations of Algebra ............................. 7
  2.2 Bilinear Groups ..................................... 8
  2.3 Complexity Assumptions ............................ 9
  2.4 Cryptographical Tools ............................... 12
    2.4.1 Hash Function ................................. 12
    2.4.2 Random Oracle Model .......................... 12
    2.4.3 Public-Key Encryption ......................... 13
    2.4.4 Digital Signature ............................. 15
    2.4.5 Zero-Knowledge Proof ......................... 18
    2.4.6 Groth-Sahai Non-interactive Proof Systems for Bilinear Groups 19

3 Related Work 23
  3.1 Provable Data Procession (PDP) ................... 23
  3.2 Proof of Retrievability ............................ 25
  3.3 Third Party Auditing ............................... 26
List of Figures

2.1 Modules and maps [29] ....................................................... 20
2.2 Maps for pairing product equations. [29] ............................... 22
4.1 The third party auditing protocol by Wang et al. [44] .......................... 32
5.1 Indistinguishability game run between $A$ and $S$ .......................... 40
5.2 Indistinguishability analysis on Wang et al.’s RIC Protocol [47] ....... 43
5.3 The third party auditing protocol by Wang et al. [44] .................. 44
5.4 Indistinguishability analysis on Wang et al. RIC Protocol [44] ...... 45
1.1 Background

Cloud computing is a general term for anything that involves delivering hosted service over the Internet. Cloud computing has gained its popularity and has been widely applied to various computing fields because of its convenience. As long as users login on a cloud service with their user names and passwords, they can commit their computing task in the cloud. Cloud services can be broadly divided into three categories: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). Cloud computing has been well developed, with various services for a variety of applications such as storage services, virtual machines, database servers, etc. The major benefit from cloud computing has been well recognised in that it has showed great flexibility in computing, mobility of data access, cost reduction of managing data and services, etc.

As one of major service in cloud computing, cloud storage has been the most popular application adopted by almost all kinds of computing devices. In fact, it has become a trend that individuals and IT enterprises store data remotely to the cloud in a flexible on-demand manner, which has become a popular way of data outsourcing. This has reduced the burden for storage management and maintenances and costs on hardware and software, with great advancement of universal data access and convenience to users. In fact, cloud storage has become one of the major services in cloud computing where user data are stored and maintained by cloud servers. It allows users to access their data via computer networks at anytime and from anywhere.

Although the great advantage of cloud storage, security is still a major challenge. There are two major concerns on security which have been critical to the success of cloud storage systems. The first one is data security, which generally addresses
the issues of confidentiality and authenticity. It can usually be solved by using traditional cryptographic approaches. The second concern is about data integrity (e.g., [36 2 32]), which mainly concerns about whether the original files being kept well by the cloud server. Hashing and digital signatures can be adopted to measure to verify data integrity. However, the integrity requirement of cloud data require special treatments, because the following reasons. Since cloud server is generally regarded as untrusted, the adversary model should capture malicious cloud servers. It is also due to public verifiability, where means that the integrity proof from the cloud server is able to be checked by a third party, who has no knowledge about the stored data. One of the solutions to this issue aims to provide an efficient and secure method to allow a third party auditor (TPA) to check if the data item has been well maintained, i.e., the integrity of the data is ensured. As the data item is stored in the cloud, it is generally referred to as Remote Integrity Checking (RIC).

A major challenge to RIC is the data confidentiality and privacy of data against potential adversary who might be interested in the information of the stored data item. Depending on the security model, the adversary could be an eavesdropper, a malicious TPA or a malicious cloud server. In general, we should not allow the TPA to know the content of the stored data item, while an RIC procedure is being conducted. The difficulty is also due to the full control of the cloud server to the storage.

There are a number of proposed solutions which have been published in the literature. However, they are far from adequate to address the problem related to RIC. The work presented in this thesis provides a substantial contribution to RIC solutions and a critical foundation and security models to RIC systems.

1.2 Previous Solutions

The research of RIC is much boarder than RIC itself. Simple data integrity check in a remote data storage can be done by periodically examining the data files stored on the cloud server, but such an approach can be very expensive if the amount of data is huge. An interesting problem is to check data integrity remotely without the need of accessing the full copy of data stored on the cloud server. For example, the data owner possesses some verification token (e.g. a digest of the data file [18 23 5]), which is very small compared with the stored dataset. Several cryptographic notions can also be associated to RIC. The notion of Proof of Retrievability (POR)
is a primitive, which addresses data retrievability. However, RIC and POR are two different concepts, which serve to different purposes though they have some similarity. The other well-know notion is Proof of Data Ownership (PDP). Again, RIC and PDP are different concepts, while they have some similarity, especially when the cloud server acts as the prover.

POR is, loosely speaking, a kind of Proof of Knowledge (POK) where the knowledge is the data file. However, as pointed out in [31], in a POR, unlike a POK, neither the prover nor the verifier need actually have knowledge of the file $F$. Recently, there are a number of RIC research papers (e.g., [45, 47, 44]), which have misused the concept of POR and treated it as the same as RIC. This unfortunately caused the failure of many RIC schemes in the literature. One of errors made was due to wrongly applying the POR concept to RIC. One of examples is the RIC scheme recently proposed by Wang et al. To able to correctly adopt a POR scheme to RIC, we must correctly define the security model, which captures the goal of RIC (and POR). RIC requires that the data stored in the cloud server to be well maintained, which means that it will be regarded as a failure, if a file $F$ is changed to $F' = F + \Delta$, while the prover (cloud server) is still able to prove its integrity. However, POR has a different definition. The goal of POR is to prove the stored file is retrievable by the user. This means that as long as the file can be retrieved, the file can be stored in any form. For example, if the cloud server holds $F'$ and $\Delta$, then it should be able to prove the the file can be retrieved by the user. Another example is due to Juel and Kaliski Jr., where they encrypts file $F$ and randomly embeds a set of randomly-valued check blocks called sentinels. The use of encryption here renders the sentinels indistinguishable from other file blocks. The verifier challenges the prover by specifying the positions of a collection of sentinels and asking the prover to return the associated sentinel values. If the prover has modified or deleted a substantial portion of $F$, then with high probability it will also have suppressed a number of sentinels. This model obviously differs from the RIC model.

Provable Data Procession (PDP) has been seen as a close analog to the RIC. It allows a client that has stored data at an untrusted server to verify that the server possesses the original data without retrieving it. The model of PDP addresses probabilistic proofs of possession by sampling random sets of file blocks from the server. The client maintains a constant amount of metadata, which can be used to verify the proof. The PDP model for remote data checking usually supports large
data sets in distributed storage systems. If we regard the PDP prover is the cloud
data server, then it can be considered as an RIC procedure. However, RIC covers
wider aspects including public verifiability and privacy (we all elaborate more about
this later). Therefore, PDP is usually retreated as a separate topic.

An interesting problem is to check data integrity remotely without storing the
data locally. This approach is also referred to as client-side deduplication. This
makes data management more challenging, as clients do not have backups of the
data. A failure of the data server inevitably leads to data loss or illegal disclosure;
therefore, providing new and innovative solutions to these issues is critical to cloud
computing. In addition, the cloud storage operator would like to store a single copy
of each file, regardless of how many clients ask to store that file. This is referred to as
server-side deduplication, or cross-user deduplication. This technique is particularly
important given the increase of the data volume as well as the demand for online
storage services. Providing a remote integrity check whilst retaining client-side
deduplication and server-side deduplication will be desirable, as it greatly reduces
the risk of data loss. Unfortunately, to date, there exists no single unified model
that completely captures all the requirements of remote integrity checks in cloud
storage.

### 1.3 Motivation and Contributions

As mentioned earlier, the proofs of remote data retrievability have been wrongly
applied to RIC by several authors. However, the definitions of retrievability and
RIC are entirely different. In a retrievability proof, the verifier’s or user’s goal is
to prove that the stored file can be retrieved, whether or not the file is modified by
the adversary (the cloud server, for example). RIC defines an entirely different goal:
the verifier or user must be able to confirm that the file stored in the data storage
system has not been modified or altered.

Demonstrating this using Shacham and Waters POR scheme [42]. We found
that the linear property of the homomorphic authenticators is subject to a trivial
attack when it is applied to RIC, while the scheme is perfect for data retrieval. The
mechanism of the attack is as follows. Let us assume that $\sum_{i=0}^{n} a_i x_i$ is a (public)
linear homomorphic function, in which $a_i$ denotes coefficients and $x_i$ variables related
to the stored dataset. To launch an attack, the adversary appends a value $\alpha$ so that
it becomes $\sum_{i=0}^{n} a_i x_i + \alpha$, which is still a linear homomorphic function. Clearly, $\alpha$
1.3. Motivation and Contributions

stems from the change to the stored dataset \( \{x_i\} \). When it is applied to a data retrievability proof, the adversary (usually, the cloud server) manipulates the data flow sent to the user (verifier) so that the adversary can remove the components that were generated due to \( \alpha \) in response of a user query. Therefore, the user can still retrieve the correct dataset using an extractor defined in their algorithm. However, the integrity of the stored dataset has been broken. We found that there is no trivial solution to this problem that maintains data retrievability.

In this thesis, we will domesticate this type of flaws by using a typical example \[44\]. Our cryptanalysis provided in this thesis can also been applied to other similar RIC protocols which are based on the notion of POR.

It is also very important that an auditing process should not introduce new vulnerabilities of unauthorized information leakage towards their data security \[43\]. The previous efforts in RIC accommodate several security features including data integrity and confidentiality, which mainly ensure secure maintenance of data. However, they do not cover the issue of data privacy, which means that the communication flows (RIC proofs) from the cloud server should not reveal any useful information to the adversary. Intuitively, by “privacy”, we mean that an adversary should not be able to distinguish which file has been uploaded by the client and maintained by the cloud server. We refer it as Indistinguishability (or IND, for short). We believe that it is very important to consider such privacy issues adequately in protocol designs.

We show that the privacy property can be proved with formal proofs of indistinguishability (IND) that can be formalised by an IND game between a prover and a verifier, and we found that none of the RIC schemes in the literature capture the IND property. We refer this IND property as IND-privacy. Again, with the RIC scheme in \[44\] as an example, we demonstrate that it does not capture the IND-privacy.

The existing RIC protocols reveal some information on target file whose integrity is proved by the cloud server. Therefore, the IND adversary is able to distinguish two different files target in an RIC proof. We show that the IND-privacy can be achieved by using the Witness Indistinguishable (WI) proof to mask an RIC proof from the cloud server. That is, the cloud server proves by zero knowledge that it knows the RIC proof without revealing the actual proof to the adversary. We provide a concrete example by using the RIC scheme in \[44\]. We proved that the IND-privacy has indeed achieved when the WI proof is applied to the scheme.
1.4 Organisation of This Thesis

This thesis consists of six chapters. The highlight of this work is twofold: cryptanalysis of the RIC scheme in [44] and introduction of the new notion of IND-privacy. Apart from this chapter, we provide a brief outline of each chapter in following.

In Chapter 2, we describe some preliminaries, which will be applied to our schemes. It includes the number theoretical notions, complexity assumptions, digital signatures and encryption, and other cryptographic tools.

Chapter 3 provides a comprehensive review of related work. We start by describing the security requirements associated with RIC, POR and PDP, which are then revisited in terms of their security models.

In Chapter 4, we present a rigorous cryptanalysis on Wang et al.’s RIC protocol [44]. We found that their protocol is flawed and does not meet their security assumptions, which means that the adversary can modify any file stored in the cloud server and the cloud server can still prove the integrity of the file. We demonstrate our attacks based on two attacking scenarios and show the detail flaws under our attacks.

We found that RIC privacy was not adequately studied in the literature and believed that it should be captured in all RIC protocols. In Chapter 5, we introduce the new notion of IND-privacy, which provides a clear and sound definition of RIC privacy. Our privacy definition is essential for RIC protocols. We also provide a comprehensive cryptanalysis of the RIC scheme in [44]. We show that the RIC scheme in [44] does not capture IND-privacy. Therefore, we provide a novel method to repair these flawed protocols by using WI proofs. We prove that our new protocol indeed provide IND-privacy. Our solution can be also applied to other RIC protocols.

In Chapter 6, we conclude this thesis and discuss some future work.
In this chapter, we describe some related cryptographic definitions and tools we will use in this thesis.

2.1 Foundations of Algebra

In this section, we revisit some basic algebra definitions including group and field.

**Group**

A group consists of a set of elements and an operation which is executed between any two elements in the set. The formal definition of a group is described as follows:

**Definition 2.1 (Group)** A group \((G, \otimes)\) is a set \(G\) equipped with an operation \(\otimes\), and satisfies the following properties:

1. **Closure.** For all \(g, h \in G\), \(g \otimes h \in G\);
2. **Associativity.** For all \(g, h, \eta \in G\), \((g \otimes h) \otimes \eta = g \otimes (h \otimes \eta)\);
3. **Identity.** There exists \(1_G \in G\) called the identity of \((G, \otimes)\), such that \(1_G \otimes g = g \otimes 1_G = g\) for all \(g \in G\);
4. **Inverse.** For all \(g \in G\), there exists \(g^{-1} \in G\) called the inverse of \(g\) such that \(g \otimes g^{-1} = g^{-1} \otimes g = 1_G\).

For simplicity, a group \((G, \otimes)\) is often denoted as \(G\) when the operation \(\otimes\) is clear. The number of the elements in \(G\) is called the order of \(G\) and denoted as \(|G|\). A group \(G\) is a finite group if \(|G|\) is finite; otherwise, it is an infinite group. A group \(G\) is an Abelian group if for all \(g, h \in G\), \(g \otimes h = h \otimes g\).
Let $G(1^\ell)$ be a group generator which takes as input $1^\ell$ and outputs a group $G$ with order $p$, namely $G(1^\ell) \rightarrow (p,G)$. 

**Definition 2.2 (Order of Group Element)** Suppose that $g \in G$, the order of $g$ in $G$ is the least $i \in \mathbb{Z}^+$ such that $g^i = 1_G$. If for all $i \in \mathbb{Z}^+$, $g^i \neq 1_G$, the order of $g$ is infinite. The order of $g$ is denoted as $\text{ord}(g)$.

Especially, if any element in a group $G$ can be expressed by a specially element in $G$, $G$ is called as a cyclic group. The formal definition of a cyclic group is as follows:

**Definition 2.3 (Cyclic Group.)** A group $G$ is a cyclic group if there exists $g \in G$, for all $h \in G$, there exists $i \in \mathbb{Z}$ such that $h = g^i$. The element $g$ is called as a generator of the group $G$. $G$ is said to be generated by $g$ and denoted as $G = \langle g \rangle$.

**Field**

A field consists of a set of elements and two operations defined between any two elements in the set. The formal definition of a field is described as follows.

**Definition 2.4 (Field)** A field $(\mathbb{F}, \oplus, \otimes)$ consists of a set $\mathbb{F}$ and two operations: addition $\oplus$ and multiplication $\otimes$, and satisfies the following properties.

1. **Addition Group.** $(\mathbb{F}, \oplus)$ is an Abelian group. The identity of the group $(\mathbb{F}, \oplus)$ is denoted as $0_\mathbb{F}$ and called additive identity or zero-element;

2. **Multiplication Group.** Let $\mathbb{F}^* = \mathbb{F} - \{0_\mathbb{F}\}$. $(\mathbb{F}^*, \otimes)$ is an Abelian group. The identity of the group $(\mathbb{F}^*, \otimes)$ is denoted as $1_\mathbb{F}$ and called as multiplicative identity;

3. **Distributivity.** For all $g, h, \eta \in \mathbb{F}$, $(g \oplus h) \otimes \eta = (g \otimes \eta) \oplus (h \otimes \eta)$.

### 2.2 Bilinear Groups

In this section, we review the knowledge related to bilinear group.

**Definition 2.5 (Bilinear Map [10])** Suppose that $G_1$, $G_2$ and $G_T$ are three cyclic groups with the same order $p$. Let $g$ and $h$ be the generators of $G_1$ and $G_2$, respectively. A bilinear map (pairing) is a map $e : G_1 \times G_2 \rightarrow G_T$ satisfying the following properties :
2.3 Complexity Assumptions

1. **Bilinearity.** For all \( x \in G_1, y \in G_2 \) and \( a, b \in \mathbb{Z}_p \), \( e(x^a, y^b) = e(x, y)^{ab} \).

2. **Non-degeneracy.** \( e(g, h) \neq 1_{G_T} \) where \( 1_{G_T} \) is the identity of the group \( G_T \).

3. **Computability.** For all \( x \in G_1 \) and \( y \in G_2 \), there exists an efficient algorithm to compute \( e(x, y) \).

**Definition 2.6** (Bilinear Groups [25]) \( G_1, G_2, \) and \( G_T \) constitute a bilinear group if there exists a bilinear map \( e : G_1 \times G_2 \to G_T \), where \( |G_1| = |G_2| = |G_T| = p \).

Galbraith, Paterson and Smart [25] divided pairing operations used in cryptography into three types:

1. \( G_1 = G_2 \);

2. \( G_1 \neq G_2 \), there exists an efficiently computable homomorphism map \( \psi : G_1 \to G_2 \);

3. \( G_1 \neq G_2 \), there are no efficiently computable homomorphism maps between groups \( G_1 \) and \( G_2 \).

We say that a pairing is symmetric if \( G_1 = G_2 \) and denote the symmetric bilinear group as \( (e, p, G_1, G_2, G_T) \). Pairing is often constructed on suitable elliptic curves, so its efficiency is determined by the selected elliptic curves. When selecting elliptic curves for a pairing, two factors must be considered: the group size \( l \) of the elliptic curves and the embedding degree \( d \). Generally, to achieve the security of 1,024-bit RSA, the two parameters \( l \) and \( d \) should satisfy \( l \times d \geq 1,024 \) [33, 13]. The security parameter \( \ell \) must be large enough (say, at least \( 2^{160} \)) so that the discrete logarithm problem in groups \( G_1 \) and \( G_2 \) is hard. The embedding degree \( d \) should not be too large (less than or equal to 50) so that the bilinear maps can be computable.

In the rest of this thesis, we denote \( \mathcal{GG}(1^\ell) \to (e, p, G_1, G_2, G_T) \) as a bilinear group generator which takes as input \( 1^\ell \) and outputs bilinear groups \( (e, p, G_1, G_2, G_T) \) with order \( p \) and a bilinear map \( e : G_1 \times G_2 \to G_T \).

### 2.3 Complexity Assumptions

In this section, we review the complexity assumptions used throughout this thesis.

**Discrete Logarithm Assumption**
2.3. Complexity Assumptions

The discrete logarithm (DL) assumption \cite{38} in a finite field is one of the basic assumptions in cryptography research. The DL assumption is defined as follows.

**Definition 2.7 (Discrete Logarithm (DL) Assumption \cite{38}.)** Let \( G(1^\ell) \rightarrow (p, G) \) and \( G = \langle g \rangle \). Given \( (g, y) \in G^2 \), we say that the discrete logarithm assumption holds on \( G \) if no PPT adversary \( A \) can compute a \( x \in \mathbb{Z}_p \) such that \( y = g^x \) with the advantage

\[
Adv^\text{DL}_A = \Pr [ y = g^x | A(p, g, y, G) \rightarrow x ] \geq \epsilon(\ell)
\]

where the probability is taken over the random choice of \( y \in G \) and the bits consumed by the adversary \( A \).

**Computational Diffie-Hellman Assumption**

Diffie and Hellman \cite{19} proposed this assumption and constructed a key exchange scheme based on it. This assumption is defined as follows.

**Definition 2.8 (Computational Diffie-Hellman (CDH) Assumption \cite{19}.)** Let \( x, y \xleftarrow{\$} \mathbb{Z}_p \), \( G(1^\ell) \rightarrow (p, G) \) and \( G = \langle g \rangle \). Given \( (g, g^x, g^y) \), we say that the computational Diffie-Hellman assumption holds on \( G \) if no PPT adversary \( A \) can compute \( g^{xy} \) with the advantage

\[
Adv^\text{CDH}_A = \Pr [ A(g, g^x, g^y) \rightarrow g^{xy} ] \geq \epsilon(\ell)
\]

where the probability is taken over the random choices of \( x, y \xleftarrow{\$} \mathbb{Z}_p \) and the bits consumed by the adversary \( A \).

Maurer \cite{35} discussed the relationships between DL assumption and CDH assumption.

**Decisional Diffie-Hellman Assumption**

Boneh \cite{9} surveyed the various applications of decisional Diffie-Hellman assumption and demonstrated some results regarding it security.

**Definition 2.9 (Decisional Diffie-Hellman (DDH) Assumption \cite{9}.)** Let \( x, y, z \xleftarrow{\$} \mathbb{Z}_p \), \( G(1^\ell) \rightarrow (p, G) \) and \( G = \langle g \rangle \). Given \( (g, g^x, g^y) \), we say that the decisional Diffie-Hellman assumption holds on \( G \) if no PPT adversary \( A \) can distinguish \( (X, Y, Z) = (g^x, g^y, g^z) \) from \( (X, Y, Z) = (g^x, g^y, g^z) \) with the advantage

\[
Adv^\text{DDH}_A = | \Pr [ A(X, Y, g^{xy}) = 1 ] - \Pr [ A(X, Y, g^z) = 1 ] | \geq \epsilon(\ell)
\]

where the probability is taken over the random choices \( x, y, z \xleftarrow{\$} \mathbb{Z}_p \) and the bits consumed by the adversary \( A \).
Computational Bilinear Diffie-Hellman

Boneh and Franklin [10] introduced this assumption. This assumption is as follows.

Definition 2.10 (Computational Bilinear Diffie-Hellman (CBDH) Assumption [10])
Let $\mathcal{G}(1^\ell) \rightarrow (e, p, G, G_T)$ and $G = \langle g \rangle$. We say that the computational bilinear Diffie-Hellman assumption holds on $(e, p, G, G_T)$ if no PPT adversaries $A$ can compute $e(g, g)^{abc}$ from $(A, B, C) = (g^a, g^b, g^c)$ with the advantage

$$\text{Adv}^{CBDH}_A = \Pr[A(A, B, C) \rightarrow e(g, g)^{abc}] \geq \epsilon(\ell)$$

where the probability is taken over the random choices of $a, b, c \xleftarrow{\$} \mathbb{Z}_p$ and the bits consumed by $A$.

Decisional Bilinear Diffie-Hellman Assumption

Boneh and Franklin [10] introduced this assumption and used it to construct an identity-based encryption (IBE) scheme. This assumption is defined as follows.

Definition 2.11 (Decisional Bilinear Diffie-Hellman (DBDH) Assumption [10])
Let $a, b, c, z \xleftarrow{\$} \mathbb{Z}_p$, $\mathcal{G}(1^\ell) \rightarrow (e, p, G, G_T)$ and $G = \langle g \rangle$. We say that the decisional bilinear Diffie-Hellman assumption holds on $(p, e, G, G_T)$ if no PPT adversary $A$ can distinguish $(A, B, C, Z) = (g^a, g^b, g^c, e(g, g)^{abc})$ from $(A, B, C, Z) = (g^a, g^b, g^c, e(g, g)^z)$ with the advantage

$$\text{Adv}^{DBDH}_A = |\Pr[A(A, B, C, e(g, g)^{abc}) = 1] - \Pr[A(A, B, C, e(g, g)^z) = 1]| \geq \epsilon(\ell)$$

where the probability is taken over the random choices of $a, b, c, z \xleftarrow{\$} \mathbb{Z}_p$ and the bits consumed by the adversary $A$.

Symmetric External Diffie-Hellman Assumption

The Symmetric External Diffie-Hellman (SXDH) assumption [29] is defined as follows.

Definition 2.12 (Symmetric External Diffie-Hellman Assumption [29])
Let $x, y, z \xleftarrow{\$} \mathbb{Z}_p$, $\mathcal{G}(1^\ell) \rightarrow (e, p, G_1, G_2, G_T)$ and $G_b = \langle g_b \rangle$ for any $b \in \{1, 2\}$. We say that the Symmetric External Diffie-Hellman Assumption holds on $(p, e, G_1, G_2, G_T)$ if no PPT adversary $A$ can distinguish $(g_b, g_b^x, g_b^y, g_b^{xy})$ from $(g_b, g_b^x, g_b^y, g_b^z)$ with the advantage

$$\text{Adv}^{SXDH}_A = |\Pr[A(g_b, g_b^x, g_b^y, g_b^{xy}) = 1] - \Pr[A(g_b, g_b^x, g_b^y, g_b^z) = 1]| \geq \epsilon(\ell)$$
where the probability is taken over the random choices of \( x, y, z \overset{R}{\leftarrow} \mathbb{Z}_p \) and the bits consumed by the adversary \( A \).

2.4 Cryptographical Tools

In this section, we introduce some useful cryptographical tools, including hash function, random oracle model, public-key encryption, digital signature and zero-knowledge proof.

2.4.1 Hash Function

Carter and Wegman [15] introduced the universal classes of hash functions and divided them into tree types. Roughly speaking, a hash function \( H : \{0,1\}^* \rightarrow \{0,1\}^\lambda \) is a deterministic function which can map a bit string with any length to a bit string with fixed length \( \lambda \). A hash function should provide the following properties [34]:

1. **Mixing Transformation.** The output of \( H \) should be computationally indistinguishable from a uniform binary string in \([0,2^\lambda]\);

2. **Pre-image Resistance.** Given a value \( y \), it is computationally infeasible to find a value \( x \) such that \( y = H(x) \);

3. **Collusion Resistance.** It is computationally infeasible to find \( x \neq y \) such that \( H(x) = H(y) \).

Hash function is an important cryptographical primitive and has been used as a building block to design encryption scheme [24], digital signature scheme [8], message authentication code (MAC) scheme [6], etc.

2.4.2 Random Oracle Model

A hash function should satisfy the mixing transformation property, namely the output of a hash function is computationally indistinguishable from the uniform distribution over its output’s space. If the output of a hash function is uniform distribution over its output’s space, it is a very powerful and ideal hash function called *random oracle* [34]. A random oracle is a powerful hash function as it combines
the properties: deterministic, efficient and uniform output. Furthermore, a random oracle is an ideal hash function as there are no so powerful computing mechanism or machinery in current computing models.

Bellare and Rogaway [8] introduced the notion of random oracle model. In this model, a special entity called Simulator can simulate every party’s behavior. So, whenever a party wants to obtain the output of a random oracle \( H \) on a value \( x \), he must make a random oracle query on the value \( x \) to the Simulator. Simulator maintains a \( H \)-table consisting of pairs \((z, H(z))\). For a query on the value \( x \), Simulator checks whether \( x \) is listed in the table. If it has been in the table, Simulator responds with the value \( H(x) \) (deterministic); otherwise, Simulator creates a new value \( H(x) \) uniformly at random from the output’s space of \( H \), adds the pair \((x, H(x))\) to the table and responds with \( H(x) \) (uniform).

Random oracle model is a very efficient tool to prove the security of cryptographic protocols. Generally, protocols designed in this model are more efficient than those designed in standard model. Whereas, a scheme which is proven to be secure in the random oracle model does not necessarily imply that it is secure in the standard model [14].

Unless otherwise specified, by saying a scheme is secure, we mean that it is secure in the standard model in this thesis.

2.4.3 Public-Key Encryption

Diffie and Hellman [19] introduced new research directions in cryptography called public-key cryptography (PKC) where two parties can communicate over public channels without compromising the security of the system.

A public-key (asymmetric) encryption (PKE) scheme is a public-key cryptographic scheme used to protect the confidentiality of the transferred massages. In a PKE scheme, a secret-public key pair is generated. Notably, it is computationally infeasible to obtain the secret key from the public key. This is in contrast with a symmetric encryption scheme where both the decryption key and the encryption key are same or it is easy to compute one from the other.

The formal definition of a PKE scheme is as follows [19]. A PKE scheme consists of the following four algorithm.

\textbf{Setup}(1^t) \rightarrow \text{params}. The setup algorithm takes as input \(1^t\) and outputs the public parameters \(\text{params}\).
2.4. Cryptographical Tools

KeyGen$(1^ℓ) → (SK, PK)$. The key generation algorithm takes as input $1^ℓ$ and outputs a secret-public pair $KG(1^ℓ) → (SK, PK)$.

Enc$(params, PK, M) → CT$. The encryption algorithm takes as input the public parameters $params$, the public key $PK$ and a message $M$, and outputs a ciphertext $CT$.

Dec$(params, SK, CT) → M$. The decryption algorithm takes as input the public parameters $params$, the secret key $SK$ and the ciphertext $CT$, and outputs the message $M$.

Definition 2.13 Correctness. We say that a public-key encryption scheme is correct if

$$\Pr \begin{bmatrix} \text{Dec}(params, SK, CT) \rightarrow M & \text{Setup}(1^ℓ) \rightarrow params; \\ \text{KeyGen}(1^ℓ) \rightarrow (SK, PK); \\ \text{Enc}(params, PK, M) \rightarrow CT \end{bmatrix} = 1$$

where the probability is taken over the random coins consumed by all algorithms in the scheme.

Security Model. The standard notion of the security for a PKE scheme is called indistinguishability against adaptive chosen ciphertext attacks (IND-CCA2) [39]. This model is defined by the following game executed between a challenger $C$ and an adversary $A$.

Setup. $C$ runs $\text{Setup}(1^ℓ)$ to generate the public parameters $params$ and sends them to $A$.

KeyGen. $C$ runs $\text{KeyGen}(1^ℓ)$ to generate the secret-public key pair $(SK, PK)$ and sends the public key $PK$ to $A$.

Phase 1. $A$ can adaptively query the decryption oracle. $A$ submits a ciphertext $CT$ to $C$, where $CT = Enc(param, PK, M)$. $C$ runs $\text{Dec}(params, SK, CT)$ and responds $A$ with $M$. This query can be made multiple times.

Challenger. $A$ submits two messages $M_0$ and $M_1$ with equal length. $C$ randomly selects $M_b$ and computes $CT^* = Enc(params, PK, M_b)$, where $b \in \{0, 1\}$. $C$ responds $A$ with $CT^*$.
Phase 2. $\mathcal{A}$ can adaptively query the decryption oracle. $\mathcal{A}$ submits a ciphertext $CT$ to $\mathcal{C}$, where the only restrict is $CT \neq CT^*$. Phase 1 is repeated. This query can be made multiple times.

**Guess.** $\mathcal{A}$ outputs his guess $b'$ on $b$. $\mathcal{A}$ wins the game if $b' = b$.

**Definition 2.14 IND-CCA2.** We say that a public-key encryption scheme is $(T, q, \epsilon(\ell))$-indistinguishable against adaptive chosen ciphertext attacks (IND-CCA2) if no PPT adversary $\mathcal{A}$ making $q$ decryption queries can win the game with the advantage

$$Adv_{\mathcal{A}}^{IND-CCA2} = \left| \Pr[b' = b] - \frac{1}{2} \right| \geq \epsilon(\ell)$$

in the above model.

Another security notion for public-key encryption is called indistinguishability against adaptive chosen plaintext attacks (IND-CPA). In this model, the adversary $\mathcal{A}$ is not allowed to query the decryption oracle. The formal definition for this model is as follows.

**Definition 2.15 IND-CPA.** We say that a public-key encryption scheme is $(T, \epsilon(\ell))$-indistinguishable against adaptive chosen plaintext attacks (IND-CPA) if no PPT adversary $\mathcal{A}$ who is restricted to query the decryption oracle can win the game with the advantage

$$Adv_{\mathcal{A}}^{IND-CPA} = \left| \Pr[b' = b] - \frac{1}{2} \right| \geq \epsilon(\ell)$$

in the above model.

Some well known PKE schemes include the ElGamal encryption scheme [21], RSA encryption scheme [40], Cramer-Shoup encryption scheme [17] and RSA-OAEP encryption scheme [24].

**2.4.4 Digital Signature**

Digital signature was proposed by Diffie and Hellman [19]. It is the electronic version of a handwritten signature. A valid digital signature can convince a verifier that it was generated by a known party for a public message. Especially, a digital signature can provide non-repudiation property, namely a signer cannot deny he has generated the signature.

A digital signature scheme is formally defined as follows [28]. It consists of the following four algorithms.
Setup\((1^\ell) \rightarrow \text{params.}\) The setup algorithm takes as input \(1^\ell\) and outputs the public parameters \text{params.}\)

KeyGen\((1^\ell) \rightarrow (SK, PK)\). The key generation algorithm takes as input \(1^\ell\) and outputs a secret-public key pair \((SK, PK)\).

Sign\((\text{params}, SK, M) \rightarrow \sigma\). The signature algorithm takes as input the public parameters \text{params}, the secret key \(SK\) and a message \(M\), and outputs a signature \(\sigma\) on \(M\).

Verify\((\text{params}, M, PK, \sigma) \rightarrow \text{True/False}\). The verification algorithm takes as input the public parameters \text{params}, the message \(M\), the public key \(PK\) and the signature \(\sigma\), and outputs True if \(\text{Sign}(\text{params}, M, SK) \rightarrow \sigma\); otherwise, it outputs False.

**Definition 2.16 Correctness.** We say that a digital signature is correct if

\[
\Pr \left[ \text{Verify}(\text{params}, M, PK, \sigma) \rightarrow \text{True} \bigg| \text{Setup}(1^\ell) \rightarrow \text{params}; \text{KeyGen}(1^\ell) \rightarrow (SK, PK); \text{Sign}(\text{params}, SK, M) \rightarrow \sigma. \right] \geq 1 - \epsilon(\ell)
\]

and

\[
\Pr \left[ \text{Verify}(\text{params}, M, PK, \sigma) \rightarrow \text{False} \bigg| \text{Setup}(1^\ell) \rightarrow \text{params}; \text{KeyGen}(1^\ell) \rightarrow (SK, PK); \text{Sign}(\text{params}, SK, M) \rightarrow \sigma. \right] < \epsilon(\ell)
\]

where the probability is taken over the random coins consumed by all algorithms in the scheme.

**Security Model.** A digital signature scheme should achieve the traditional security called existential unforgeability under adaptive chosen message attacks (EU-CMA)\(^{28}\). This model is formally defined by the following game executed between a challenger \(C\) and an adversary \(A\).

**Setup.** \(C\) runs \(\text{Setup}(1^\ell)\) to generate the public parameters \text{params} and sends them to \(A\).

**KeyGen.** \(C\) runs \(\text{KeyGen}(1^\ell)\) to generate a secret-public pair \((SK, PK)\) and sends \(PK\) to \(A\).
Query. \( \mathcal{A} \) can adaptively query the signature oracle. \( \mathcal{A} \) sends a message \( M \) to \( \mathcal{C} \). \( \mathcal{C} \) runs \( \text{Sign}(\text{params}, SK, M) \) to generate a signature \( \sigma \) on \( M \) and responds \( \mathcal{A} \) with \( \sigma \). This query can be made multiple times.

Output. \( \mathcal{A} \) outputs a message-signature pair \((M^*, \sigma^*)\). \( \mathcal{A} \) wins the game if \( M^* \) has not been used to query the signature oracle and \( \text{Verify}(\text{params}, M^*, PK, \sigma^*) \rightarrow \text{True} \).

**Definition 2.17 EU-CMA.** We say that a digital signature scheme is \((T, q, \epsilon(\ell))\)-existentially unforgeable against adaptive chosen message attacks (EU-CMA) if no PPT adversary \( \mathcal{A} \) can win the game with the advantage

\[
\text{Adv}^{\text{EU-CMA}}_{\mathcal{A}} = \Pr[\text{Verify}(\text{params}, M^*, PK, \sigma^*) \rightarrow \text{True}] \geq \epsilon(\ell)
\]

in the above model.

An, Dodis and Rabin [1] proposed a stronger definition for the security of digital signature schemes called strongly existential unforgeability under an adaptive chosen message attack (SEU-CMA). This model is defined by the following game executed between a challenger \( \mathcal{C} \) and an adversary \( \mathcal{A} \).

**Setup.** \( \mathcal{C} \) runs \( \text{Setup}(1^\ell) \) to generate the public parameters \( \text{params} \) and sends them to \( \mathcal{A} \).

**KeyGen.** \( \mathcal{C} \) runs \( \text{KeyGen}(1^\ell) \) to generate a secret-public pair \((SK, PK)\) and sends \( PK \) to \( \mathcal{A} \).

**Query.** \( \mathcal{A} \) can adaptively query the signature oracle. \( \mathcal{A} \) adaptively sends messages \( \{M_1, M_2, \cdots, M_q\} \) to \( \mathcal{C} \). \( \mathcal{C} \) runs \( \text{Sign}(\text{params}, SK, M_i) \) to generate a signature \( \sigma_i \) on \( M_i \) and responds \( \mathcal{A} \) with \( \sigma_i \), for \( i = 1, 2, \cdots, q \).

**Output.** \( \mathcal{A} \) outputs a message-signature pair \((M^*, \sigma^*)\). \( \mathcal{A} \) wins the game if \((M^*, \sigma^*) \notin \{(M_1, \sigma_1), (M_2, \sigma_2), \cdots, (M_q, \sigma_q)\}\) and \( \text{Verify}(\text{params}, M^*, PK, \sigma^*) \rightarrow \text{True} \).

**Definition 2.18 SEU-CMA.** We say that a digital signature scheme is \((T, q, \epsilon(\ell))\)-strongly existentially unforgeable against adaptive chosen message attacks (SEU-CMA) if no PPT adversary \( \mathcal{A} \) can win the game with the advantage

\[
\text{Adv}^{\text{SEU-CMA}}_{\mathcal{A}} = \Pr[\text{Verify}(\text{params}, M^*, PK, \sigma^*) \rightarrow \text{True}] \geq \epsilon(\ell)
\]

in the above model.
2.4.5 Zero-Knowledge Proof

Introduced by Goldwasser, Micali and Rackoff \cite{27}, a zero-knowledge proof is an interactive protocol which can be used by a prover to convince a verifier that a statement is true without releasing any more information than the validity of the statement. A zero-knowledge proof of knowledge (ZK-PoK) is a protocol which can be used by a prover to convince a verifier that he knows a secret value without the verifier knowing anything about the value. There are two parties in a zero-knowledge proof: a prover $P$ which has unlimited computation ability and a verifier $V$ which is computationally bound. By $(P \leftrightarrow V)[x]$, we denote that $P$ proves to $V$ that the statement $x$ is correct. The formal definition for a perfect zero-knowledge proof \cite{26} is as follows.

**Definition 2.19** A pair $(P \leftrightarrow V)$ is an interactive proof system for a language $L$ if the following properties can be satisfied:

1. **Completeness.** For all $x \in L$, $\Pr[V(x, s) = 1|(P \leftrightarrow V)[x] \rightarrow s] = 1 - \frac{1}{n^\kappa}$ for each $\kappa$ and the sufficient large input length $n$.

2. **Soundness.** For all $x \notin L$ and $P'$, $\Pr[V(x, s) = 1|(P' \leftrightarrow V)[x] \rightarrow s] \leq \frac{1}{n^\kappa}$. In other words, if $P'$ can convince $V$ that $x \notin L$ is correct with the advantage $\epsilon$, there exists a knowledge extractor, given rewindable black-box access \footnote{In a rewindable black-box access, the extractor can send any values which its selects to the prover and obtain the corresponding outputs of the prover, without knowing how the prover computes the outputs. It allows the extractor to literally rewind a run of the prover to a previous state.} to $P'$, can output the witness of the statement $x$ with the advantage $\epsilon - \frac{1}{n^\kappa}$.

3. **Zero-Knowledge.** For all $x \in L$ and $V$, there exists an simulator $S$ such that the two outputs $S_V(x)$ and $V(x)$ are indistinguishable, where $S_V(x)$ denotes the distribution generated by the simulator $S$ on input $x$ and $V(x)$ denotes the distribution generated by the verifier $V$ who interacts with the prover $P$ on inputs $x$.

All languages in $\text{NP}$ have zero-knowledge proofs if there exist one-way functions \cite{26}.
2.4.6 Groth-Sahai Non-interactive Proof Systems for Bilinear Groups

Groth and Sahai [29] introduced efficient non-interactive witness-indistinguishable proofs and non-interactive zero-knowledge proofs. Their goal is to spread the use of non-interactive zero-knowledge proofs from mainly theoretical purposes to the large class of practical cryptographic protocols based on bilinear groups. In this thesis, we employ the Groth-Sahai proof technique to our provably secure RIC protocol, which captures IND-privacy. In this section, we revisit the main component of their scheme, which is directly related to this thesis.

Let $\mathcal{R}, +, \cdot, 0, 1$ be a finite commutative ring. An $\mathcal{R}$-module $A$ is an abelian group $(A,+)$ where the ring acts on the group such that $\forall r, s \in \mathcal{R}, \forall x, y \in A$:

$$(r + s)x = rx + sx \land r(x + y) = rx + ry \land r(sx) = (rs)x \land 1x = x.$$  

A cyclic group $G$ of order $n$ can naturally be viewed as a $\mathbb{Z}_n$-module. Let $A_1, A_2, A_T$ be finite $\mathcal{R}$-modules with a bilinear map: $f : A_1 \times A_2 \rightarrow A_T$. Consider quadratic equations over variables $x_1, ..., x_m \in A_1, y_1, ..., y_n \in A_2$ of the form

$$\sum_{j=1}^{n} f(a_j, y_j) + \sum_{i=1}^{m} f(x_i, b_i) + \sum_{i=1}^{m} \sum_{j=1}^{n} \gamma_{ij} f(x_i, y_j) = t. \tag{2.1}$$

In order to simplify notation, let us define

$$\vec{x} \cdot \vec{y} = \sum_{j=1}^{n} f(x_i, y_i). \tag{2.2}$$

Then equation (2.1) can be written as

$$\vec{a} \cdot \vec{y} + \vec{x} \cdot \vec{b} + \vec{x} \cdot \Gamma \vec{y} = t. \tag{2.3}$$

For pairing product equations, define $\mathcal{R} = \mathbb{Z}_n$, $A_1 = G_1$, $A_2 = G_2$, $A_T = G_T$, $f(x, y) = e(x, y)$, and Equation (2.1) can be written as

$$\langle \vec{A} \cdot \vec{Y} \rangle \langle \vec{X} \cdot \vec{B} \rangle (\vec{X} \cdot \Gamma \vec{Y}) = t_T. \tag{2.4}$$

Commitment from Modules

In the Groth-Sahai proofs, we will commit to the variables $x_1, ..., x_m \in A_1, y_1, ..., y_n \in A_2$. It is done by mapping them into other $\mathcal{R}$-modules $B_1, B_2$ and making the commitments in those modules.
To commit to elements from one \( \mathcal{R} \)-modules \( A \), the public key for the commitment scheme will describe another \( \mathcal{R} \)-module \( B \) and \( \mathcal{R} \)-linear maps \( \iota : A \to B \) and \( p : B \to A \). Operations in the modules and computation of the map \( \iota \) will be efficiently computable, but \( p \) is had to compute. The public key will also contain elements \( u_1, \ldots, u_\hat{m} \in B \). To commit to \( x \in A \), pick \( r_1, \ldots, r_\hat{m} \leftarrow \mathcal{R} \) at random and compute the commitment

\[
c := \iota(x) + \sum_{i=1}^{\hat{m}} r_i u_i.
\] (2.5)

The commitment scheme will have two types of commitment keys.

- Binding key: It defines \( (B, \iota, p, u_1, \ldots, u_\hat{m}) \) where \( \forall i : p(u_i) = 0 \) and \( p \circ \iota \) is non-trivial. The commitment (2.5) therefore contains the nontrivial information \( p(c) = p(\iota(c)) \) about \( x \).

- Hiding key: It defines \( (B, \iota, p, u_1, \ldots, u_\hat{m}) \) where \( \iota(A) \subseteq \langle u_1, \ldots, u_\hat{m} \rangle \). The commitment (2.5) therefore perfectly hides the element \( x \) when \( r_1, \ldots, r_\hat{m} \) are chosen at random from \( \mathcal{R} \).

For multiple elements \( x_1, \ldots, x_m \in A \), we write \( \vec{c} := \iota(\vec{x}) + R\vec{u} \) with \( R \in \text{Mat}_{m \times \hat{m}}(\mathcal{R}) \) for making commitments \( c_1, \ldots, c_m \) where \( c_i := \iota(x_i) + \sum_{j=1}^{\hat{m}} r_{ij} u_j \).

\[
\begin{align*}
A_1 & \times A_2 & \xrightarrow{f} & A_T \\
\iota_1 & \leftarrow p_1 & \iota_2 & \leftarrow p_2 & \iota_T & \leftarrow p_T \\
B_1 & \times B_2 & \xrightarrow{F} & B_T \\
\forall x \in A_1, \forall y \in A_2 : & F(\iota_1(x), \iota_2(y)) = \iota_T(f(x, y)), & \\
\forall x \in B_1, \forall y \in B_2 : & f(p_1(x), p_2(y)) = p_T(F(x, y))
\end{align*}
\]

Figure 2.1: Modules and maps [29]

The Groth-Sahai proof system is in the common reference string (CRS) model. Part of the CRS specifies \( B_1, \iota_1, p_1, u_1, \ldots, u_\hat{m} \) and \( B_2, \iota_2, p_2, v_1, \ldots, v_\hat{m} \), which are commitment keys for \( A_1 \) and \( A_2 \). Another part of the CRS specifies a third \( \mathcal{R} \)-module \( B_T \) together with \( \mathcal{R} \)-linear maps \( \iota_T : A_T \to B_T \) and \( p_T : B_T \to A_T \) and a bilinear
2.4. Cryptographical Tools

map \( F : B_1 \times B_2 \to B_T \). It is required that the maps are commutative as described in Figure 2.1 and with the exception of \( p_1, p_2, \) and \( p_T \), they are efficiently computable.

**Instantiation Based on the SXDH Assumption**

**Setup.** The setup algorithm \( G_{SXD} \) returns a prime order bilinear group \( gk = (p, G_1, G_2, G_T, e, P_1, P_2) \).

**Commitment.** Consider a group \( G \) of prime order \( p \). We will use the a commitment key of the form

\[
\begin{align*}
    u_1 &= (\mathcal{P}, \mathcal{Q}) := (\mathcal{P}, \alpha \mathcal{P}), \\
    u_2 &= (\mathcal{U}, \mathcal{V}),
\end{align*}
\]

where \( \alpha \leftarrow \mathbb{Z}_p^* \) is chosen at random. We can choose \( u_2 \) in two different ways: \( u_2 := tu_1 \) or \( u_2 := tu_1 - (O, \mathcal{P}) \) for a random \( t \in \mathbb{Z}_p^* \). The former choice of \( u_2 \) gives a perfectly binding commitment key, whereas the latter choice of \( u_2 \) gives a perfectly hiding commitment key. These two types of commitment keys are computationally indistinguishable under the decision Diffie-Hellman assumption in \( G \).

In order to commit a value \( \mathcal{X} \in G_1 \) using random \( r_1, r_2 \in \mathbb{Z}_p^* \):

\[
\begin{align*}
    \mathcal{U}(\mathcal{Z}) &= (O, \mathcal{Z}), \\
    p(\mathcal{Z}_1, \mathcal{Z}_2) &= \mathcal{Z}_2 - \alpha \mathcal{Z}_1, \\
    c &= \mathcal{U}(\mathcal{X}) + r_1 u_1 + r_2 u_2.
\end{align*}
\]

We can see that on a binding key where \( u_2 = tu_1 \), we have that \( p \circ \mathcal{U} \) is the identity map on \( G \) and \( p(u_1) = p(u_2) = O \). The commitment \( c = ((r_1 + r_2 t) \mathcal{P}, (r_1 + r_2 t) \mathcal{Q} + \mathcal{X}) \) corresponds to an ElGamal encryption of \( \mathcal{X} \).

Commitment to a scalar \( x \in \mathbb{Z}_p \) using randomness \( r \in \mathbb{Z}_p \) works as follows.

\[
\begin{align*}
    u &= u_2 + (O, \mathcal{P}), \\
    \mathcal{U}'(z) &= z u, \\
    p'(z_1, \mathcal{P}; z_2, \mathcal{P}) &= z_2 - \alpha z_1, \\
    c &= \mathcal{U}'(x) + ru_1.
\end{align*}
\]

On a binding key \( p' \circ \mathcal{U}' \) is the identity map and \( p'(u_1) = 0 \), so the commitment scheme is perfectly binding and the commitment \( c = ((r + xt) \mathcal{P}, (r + xt) \mathcal{Q} + x \mathcal{P}) \) is an ElGamal encryption of \( x \mathcal{P} \).

The CRS contains the commitment keys \( (u_1, u_2, v_1, v_2) \), where \( (u_1, u_2) \) is a commitment key for group \( G_1 \) implicitly defining maps \( \mathcal{U}_1, p_1, \mathcal{U}_1', p_1' \) as described above and \( (v_1, v_2) \) is a commitment key for \( G_2 \) implicitly defining maps \( \mathcal{U}_2, p_2, \mathcal{U}_2', p_2' \) as described above.

Let \( B_1 = G^2_1, B_2 = G^2_2 \) and \( B_T := G^4_T \). The map \( F \) is defined as follows.

\[
\begin{align*}
    F : G^2_1 \times G^2_2 &\to G^4_T, \\
    \left( \begin{array}{c}
    \mathcal{X}_1 \\
    \mathcal{X}_2
    \end{array} \right), \left( \begin{array}{c}
    \mathcal{Y}_1 \\
    \mathcal{Y}_2
    \end{array} \right) &\to \left( \begin{array}{c}
    e(\mathcal{X}_1, \mathcal{Y}_1) \\
    e(\mathcal{X}_1, \mathcal{Y}_2) \\
    e(\mathcal{X}_2, \mathcal{Y}_1) \\
    e(\mathcal{X}_2, \mathcal{Y}_2)
    \end{array} \right)
\end{align*}
\]
For pairing product equations, the CRS will describe $R = \mathbb{Z}_p, A_1 = G_1, A_2 = G_2, A_T = G_T; B_1 = G_1^2, B_2 = G_2^2, B_T = G_T^4$, and the following linear and bilinear maps.

![Diagram](image)

**Figure 2.2: Maps for pairing product equations.** [29]

For the multiscalar multiplication in $G_1$ or $G_2$, and quadratic equations in $\mathbb{Z}_p$, readers may refer to [29] for the details.
In this chapter, we revisit some related work and provide a comprehensive review on recent work related to Remote Integrity Check (RIC), Proof of Data Procession (PDP), Proof of Retrievability (POR) and Proof of Ownership (POW). Although the focus of this thesis is on RIC, it is important to also highlight PDP, POR and POW, as they are closely related to RIC. In fact, some well-known schemes are based on the techniques developed in PDP, POR and POW.

### 3.1 Provable Data Procession (PDP)

Ateniese et al. [3] introduced a model of PDP that can be used for RIC; namely a client that has stored data at an untrusted server can verify that the server possesses the original data without retrieving it. They presented two provably-secure PDP schemes that are more efficient than previous schemes. The overhead at the server is low, as opposed to linear in the size of the data. They also proposed a generic transformation that adds robustness to any remote data checking scheme.

Ateniese et al.’s scheme improves the response length of the simple MAC-based scheme using homomorphic authenticators or homomorphic verifiable tags. Tags computed for multiple file blocks can be combined into a single value. The client precomputes tags for each block of a file and then stores the file and its tags in a server. The client can verify that the server possesses the file by generating a random challenge for a randomly selected set of file blocks. The server retrieves the queried blocks and their corresponding tags, and uses them to generate a proof of possession. The client is thus convinced of data possession, without actually having to retrieve file blocks.

In their scheme, Ateniese et al. introduced Homomorphic Verifiable Tags, which lead to the notion of homomorphic authenticators. The authenticator $\sigma_i$ on a file
block $m_i$ is constructed in such a way that a verifier can be convinced that a linear combination of blocks $\sum_i \nu_i m_i$ (with arbitrary weights $\{\nu_i\}$) was correctly generated using an authenticator computed from $\{\sigma_i\}$. Therefore, their scheme offers two elegant features: Blockless Verification, which means that using the homomorphic verifiable tags the server can construct a proof that allow the client to verify if the server possesses certain file blocks, even if the client does not have access to the actual file blocks; and Homomorphic Tags, which means that given two tags, anyone can combine them into a tag corresponding to the sum of the messages. These features make their scheme unique. Homomorphic authenticators have also been adopted in RIC and POR, which will be described later.

Ateniese et al. also proposed a strong adversary model, which states that although the data server must answer challenges from the client, it is not trusted to store the file and may try to convince the client it possesses the file, even if the file is totally or partially corrupted. Their motivation stems from the potential issue that the data server might be financially motivated to sell the same storage resource to multiple clients.

![Figure 3.1: Protocol for provable data possession][3]

The PDP protocol of Ateniese et al. is illustrated in Figure 3.1. The client $C$ is the data owner of the file $F$, which consists of $f$ blocks. $C$ generates a small piece of metadata stored locally and sends the file $F$ to the server $S$. The local copy might be deleted by the client. Therefore, their protocol achieves client side deduplication. The server stores the file and responds to challenges sent by the client.
3.2 Proof of Retrievability

The notion of Proof of Retrievability (POR) is proposed by Juels and Kaliski [31]. The concept of POR is based on that of Proof of Knowledge (POK) [7], that is, if the prover can successfully pass the data integrity check, the verifier can retrieve the unmodified data file from the prover. Juels and Kaliski’s POR scheme uses spot-checking and error-correcting codes to ensure both “possession” and “retrievability” of data files on remote data storage systems. However, one limitation of their approach is that the integrity checking must be done by the data owner. In other words, it is not publicly verifiable.

Juels and Kaliski’s POR scheme addresses retrievability of encrypted files. They introduced a new method to allow randomly embedding a set of checking blocks called sentinels in the ciphertext of $F$. The encryption adopted in their protocol enables the sentinels to be indistinguishable from the other file blocks. During a run of POR, the verifier challenges the prover by specifying the position of a collection of sentinels and asking the prover to return the associated sentinel values. If the prover has deleted a substantial portion of $F$, then the verifier will be able to detect the change with high probability. In order to protect against small corruption by the prover, they also used error-correcting codes in their POR scheme.

Although Juels and Kaliski also described a straightforward Merkle-tree construction for public POR, this approach only works with encrypted data. Later, several improved frameworks for POR protocols were proposed by Bowers et al. [12], and Dodis et al. [20]. However, these works mainly focus on private audibility (i.e. the integrity checking is done by the data owner).

Shacham and Waters [42] later provided two new POR schemes which are efficient and provably secure. With a complete different idea from the Juels and Kaliski POR, they presented two compact POR schemes, one for private auditing and the other for public auditing. Both schemes are very efficient and proven secure. Their first scheme is based on the BLS signature scheme [11] and can be applied to public verification, but the full security was proved in the random oracle model. Their second scheme is constructed based on a pseudo-random function allows only private auditing, but its security is proved in the standard model. Shacham and Waters’ public auditing scheme serves as the basis of several Third Party Auditing RIC in [44] and other related work.
3.3. Third Party Auditing

Shacham and Waters’ first scheme is not publicly verifiable. They require breaking an erasure encoded file into \( n \) blocks \( m_1, \ldots, m_n \in \mathbb{Z}_p \) for some large \( p \). The user authenticates each block as follows. He selects a random \( \alpha \in \mathbb{Z}_p \) and PRF key \( k \) for function \( f \). These values serve as his secret key. He then calculates an authentication value for each block \( i \) as

\[
\sigma_i = f_k(i) + \alpha m_i \in \mathbb{Z}_p.
\]

\( \{m_i\} \) and the authenticator \( \{\sigma_i\} \) are stored on the server. To prove retrievability, the verifier chooses a random challenge set \( I \) of \( l \) indices along with \( l \) random \( \nu_i \) and sends them to the prover. The prover then calculates the response, a pair \( (\sigma, \mu) \), as

\[
\begin{align*}
\sigma &\leftarrow \sum_{(i,\nu_i)\in Q} \nu_i \sigma_i \quad \text{and} \quad \mu &\leftarrow \sum_{(i,\nu_i)\in Q} \nu_i m_i.
\end{align*}
\]

The verifier can check that the response was correctly formed by checking that

\[
\sigma \overset{?}{=} \alpha \mu + \sum_{(i,\nu_i)\in Q} \nu_i f_k(i).
\]

Their second scheme is constructed based on BLS signature. The structure of the BLS signatures permits them to be aggregated into linear combinations. The security proof on this scheme is base on Computational Diffie-Hellman assumption. The public auditing scheme of Shacham and Waters is presented below.

Assume that a user’s private key is \( x \in \mathbb{Z}_p \) and his public key is \( v = g^x \in G \) along with another generator \( u \in G \). The signature on block \( i \) is \( \sigma_i = [H(i)u^{m_i}]^x \). On receiving query \( Q = \{(i,\nu_i)\} \), the prover computes and sends back \( \sigma \leftarrow \prod_{(i,\nu_i)\in Q} \nu_i \sigma_i \) and \( \mu \leftarrow \prod_{(i,\nu_i)\in Q} \nu_i m_i \). The verification equation is

\[
e(\sigma, g) \overset{?}{=} e \left( \prod_{(i,\nu_i)\in Q} H(i)^{\nu_i} u^\mu, v \right).
\]

3.3 Third Party Auditing

The concept of public auditing in remote data integrity checking is first proposed by Ateniese et al. [4]. They introduced the notion of public auditability in their “provable data possession” (PDP) model for ensuring possession of data files on untrusted storages. Ateniese et al. utilized the RSA-based homomorphic linear authenticators for auditing outsourced data and suggested the approach of randomly sampling.
However, as pointed out in [44], Ateniese et al.’s public auditing scheme exposes the linear combination of sampled blocks to external auditor. When used directly, the protocol is not privacy preserving, and thus may leak user data information to the external auditor.

Shah et al. [43] has proposed to use a Third Party Auditor (TPA) to keep online storage honest by first encrypting the data, and then sending a number of pre-computed symmetric-keyed hashes over the encrypted data to the auditor. The auditor then verifies the integrity of the data file and the server’s possession of a previously committed decryption key. However, Shah et al.’s scheme only works for encrypted files, and requires the auditor to maintain state. The scheme also suffers from bounded usage since the keyed hashes may be used up.

Dynamic data storage has also been considered in some recent literatures of RIC schemes. In [47], Wang et al. proposed to combine BLS-based Homomorphic Linear Authenticator with Merkle Hash Tree to support fully data dynamics. In an independent work, Erway et al. [22] also developed a skip list based scheme to enable provable data possession with full dynamics support. However, similar to Ateniese et al.’s public auditing scheme [4], neither of these two schemes is privacy preserving, since the verification in both protocols requires the linear combination of sampled data blocks as an input.

3.4 Privacy-preserving Third Party Auditing

The first privacy-preserving third party auditing scheme was proposed by Wang et al. in [44]. In their paper, privacy-preserving means that the TPA should not be able to recover the data file from the RIC protocol messages (i.e. integrity proofs) sent by the cloud server. Wang et al.’s privacy preserving RIC scheme is still based on BLS-based Homomorphic Linear Authenticator. However, they additional employed a Schnorr Signature scheme [41] to “blind” the linear combination of sampled data blocks. Another attractive feature of Wang et al.’s privacy preserving RIC scheme is that it can support batch auditing, which allows the TPA to efficiently perform multiple auditing tasks, perhaps from different users, in a batch manner.

The protocol involves three entities: the cloud server, the cloud user, and the third party auditor (TPA). The cloud user relies on the cloud server to store and maintain his/her data. Since the user no longer keeps the data locally, it is of critical importance for the user to ensure that the data are correctly stored and maintained
by the cloud server. In order to avoid periodically data integrity verification, the user will resort to a TPA for checking the integrity of his/her outsourced data.

Nevertheless, as we will show later, Wang et al.’s scheme is not secure against their own security model and does not provide the IND-Privacy. One major contribution of this research is to find an efficient way to improve Wang et al’s scheme so that IND-Privacy can also be achieved.

### 3.5 Proof of Data Ownership

Proof of data ownership (POW) is a mechanism that allows users to prove their ownership of the files. If the files are stored in a cloud, the proof can be carried out remotely. The obvious difference between POW and RIC is that in a POW, the prover could be any one who owns or stores the file or who has the knowledge of the entire file. Roughly speaking, when we consider a POW in an RIC scenario, a POW could be similar to an RIC proof if the prover is a data server.

A technique applied at cloud storage is sever side deduplication. The cloud server only stores one single copy of each file regardless how many users asked to store this file, which could save bandwidth and hardware cost. When users trying to access their files, they prove their identification. In a typical cloud storage system with deduplication technique, users prove their ownership by hash values or some relating information. However, this kind of information tend to be obtained easily or could be obtained by other users.

In [30], some elegant solutions to POW were presented. Their solution is similar to a POR. It works by encoding files using erasure code and building a Merkle-tree over the encoded file. In this solution, they used the collision-resistant hash function $H$ and the $\alpha$-erasure-code $E$. On an input file $F$, the verifier generates the encoding $X = E(F)$ and the Merkle tree $MT_{H,b}$, then possess the only the root of the tree as verification information. At auditing phase, user receives the challenge with leaf indexes as random selected number, then proofs with sibling-paths of all the leaves. Finally, Verifier outputs its argument. Another solution in this paper is more efficient at a cost of weaker form of security. It hashes the potentially large file using universal hash function into a reduction buffer of size at most 64 MBytes, then the Merkle-tree protocol implements on this hashed values. The difference from the solution before is using a hash function with public randomness instead of encoding files using an erasure code.
Data security and privacy is an important but challenging problem in cloud computing. One of the security concerns from cloud users is how to efficiently verify the integrity of their data stored on the cloud server. Third Party Auditing (TPA) is a new technique proposed in recent years to achieve this goal. In a recent paper [44], Wang et al. proposed a highly efficient and scalable TPA protocol and also a Zero Knowledge Public Auditing protocol which can prevent offline guessing attacks. However, in this chapter, we point out several security weaknesses in Wang et al.’s protocols: first, we show that an attacker can arbitrarily modify the cloud data without being detected by the auditor in the integrity checking process, and the attacker can achieve this goal even without knowing the content of the cloud data or any verification metadata maintained by the cloud server; secondly, we show that the Zero Knowledge Public Auditing protocol cannot achieve its design goal, that is to prevent offline guessing attacks.

4.1 Introduction

In INFOCOM’10, Wang et al. [46] proposed a privacy-preserving public auditing protocol with high efficiency and scalability. In particular, the proposed protocol supports batch auditing, which means the third party auditor can concurrently handle simultaneous auditing of multiple tasks. In [44], Wang et al. further extended their TPA protocol and proposed a new Zero Knowledge Public Auditing (ZKPA) protocol. The main security goal of the ZKPA protocol is to prevent offline guessing attack (or offline dictionary attack). It is worth noting that the early version of Wang et al.’s TPA protocol published in INFOCOM’10 is insecure: Xu et al. showed [16] that the cloud server can modify the user data without being caught by the auditor in the auditing process. However, Xu et al.’s attack cannot be applied to the TPA
4.2 Review of Wang et al.’s Threat Model and Protocols

4.2.1 The Threat Model

We briefly review the threat model presented in [44]. The cloud data storage service involves three entities: the cloud server, the cloud user, and the third party auditor (TPA). The cloud user relies on the cloud server to store and maintain his/her huge amount of data. Since the user no longer keeps the data locally, it is of critical importance for the user to ensure that the data are correctly stored and maintained.
by the cloud server. In order avoid periodically data integrity verification, the user may resort to a TPA for checking the integrity of his/her outsourced data. However, the data must be kept secret from the TPA during the integrity checking process.

In [44], it is assumed that data integrity threats can come from both internal and external attacks to the cloud server, such as malicious software bugs, hackers, network bugs, etc. Besides, the cloud server may also try to hide data corruption incidents to users for the sake of reputation. However, it is assumed that the TPA is reliable and independent, and would not collude with the cloud server.

Five security goals are listed in [44]: public auditability, storage correctness, privacy preserving, batch auditing, and lightweight. Among these goals, storage correctness and privacy preserving (i.e. the auditor cannot learn the content of the user data in the auditing process) are the most important security goals that must be achieved by a privacy-preserving third party auditing protocol. For Zero Knowledge Public Auditing, there is an extra security goal, that is the protocol must be secure against offline guessing attacks.

### 4.2.2 The Third Party Auditing Protocol

Let \((p, G_1, G_2, G_T, e, g)\) denote the system parameters of a bilinear group. Wang et al.’s privacy-preserving public auditing scheme works as follows:

**Setup Phase:**

**KeyGen:** The cloud user runs KeyGen to generate the public and secret keys. Specifically, the user generates a random verification and signing key pair \((spk, ssk)\) of a digital signature scheme, a random \(x \leftarrow \mathbb{Z}_p\), a random element \(u \leftarrow G_1\), and computes \(v \leftarrow g^x\). The user secret key is \(sk = (x, ssk)\) and the user public key is \(pk = (spk, v, g, u, e(u, v))\).

**SigGen:** Given a data file \(F = (m_1, ..., m_n)\), the user first chooses uniformly at random from \(\mathbb{Z}_p\) a unique identifier \(name\) for \(F\). The user then computes authenticator \(\sigma_i\) for each data block \(m_i\) as \(\sigma_i \leftarrow (H(W_i) \cdot u^{m_i})^x \in G_1\) where \(W_i = name||i\). Denote the set of authenticators by \(\phi = \{\sigma_i\}_{1 \leq i \leq n}\). Then the user computes \(t = name||SSig_{ssk}(name)\) as the file tag for \(F\), where \(SSig_{ssk}(name)\) is the user’s signature on \(name\) under the signing key \(ssk\). It was assumed that the TPA knows the number of blocks \(n\). The user then sends \(F\) along with the verification metadata \((\phi, t)\) to the cloud server and deletes them from local storage.
Audit Phase (Figure 4.1):

VerifySig: The TPA first retrieves the file tag \( t \) and verifies the signature \( SSig_{ssk}(name) \) by using \( spk \). The TPA quits by emitting FALSE if the verification fails. Otherwise, the TPA recovers \( name \).

<table>
<thead>
<tr>
<th>TPA</th>
<th>Cloud Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Retrieve file tag ( t ) and verify its signature. Quit if fail.</td>
<td>Compute ( \mu' = \sum_{i \in I} \nu_i \mu_i ) and ( \sigma = \prod_{i \in I} \sigma_i^{\mu_i} ). Randomly pick ( r \leftarrow \mathbb{Z}_p ) and compute ( R = e(u,v)^r ) and ( \gamma = h(R) ).</td>
</tr>
<tr>
<td>2. Generate a random challenge ( chal = {i, \nu_i}_{i \in I} )</td>
<td>Compute ( \gamma = h(R) ) and then compute ( \mu = r + \gamma \mu' \mod p );</td>
</tr>
<tr>
<td>Compute ( \gamma = h(R) ) and then verify ( (\mu, \sigma, R) ).</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.1: The third party auditing protocol by Wang et al. [44].

Challenge: The TPA generates a challenge \( chal \) for the cloud server as follows: first pick a random \( c \)-element subset \( I = \{s_1, ..., s_c\} \) of set \([1, n]\), and then for each element \( i \in I \), choose a random value \( \nu_i \). The TPA sends \( chal = \{(i, \nu_i)\}_{i \in I} \) to the cloud server.

GenProof: Upon receiving the challenge \( chal \), the server generates a response to prove the data storage correctness. Specifically, the server chooses a random element \( r \leftarrow \mathbb{Z}_p \), and calculates \( R = e(u,v)^r \in G_T \). Let \( \mu' \) denote the linear combination of sampled blocks specified in \( chal \): \( \mu' = \sum_{i \in I} \nu_i \mu_i \). To blind \( \mu' \) with \( r \), the server computes \( \mu = r + \gamma \mu' \mod p \), where \( \gamma = h(R) \in \mathbb{Z}_p \). Meanwhile, the server also calculates an aggregated authenticator \( \sigma = \prod_{i \in I} \sigma_i^{\nu_i} \). It then sends \( (\mu, \sigma, R) \) as the response to the TPA.

VerifyProof: Upon receiving the response \( (\mu, \sigma, R) \) from the cloud server, the TPA validates the response by first computing \( \gamma = h(R) \) and then checking the following verification equation

\[
R \cdot e(\sigma^\gamma, g) = e((\prod_{i=s_1}^{s_c} H(W_i)^{\nu_i})^\gamma \cdot u^\mu, v). \tag{4.1}
\]
4.3. Security Weaknesses in Wang et al.’s TPA and ZKPA Protocols

The verification is successful if the equation holds.

4.2.3 The Zero Knowledge Public Auditing Protocol

As pointed out by Wang et al. in [44], the TPA protocol presented above is vulnerable to offline guessing attack. In order to prevent the attack, Wang et al. proposed the Zero Knowledge Public Auditing (ZKPA) protocol which is an extension of their TPA protocol.

The Setup phase is almost the same as in the TPA protocol, except that an additional generator \( g_1 \in G_1 \) is introduced in the user public key. In the Audit phase, upon receiving the challenge \( chal = \{(i, \nu_i)\}_{i \in I} \), the cloud server selects three random blind elements \( r_m, r_\sigma, \rho \in \mathbb{Z}_p \), and calculates \( R = e(g_1, g)^{r_\sigma}e(u, v)^{r_m} \in G_T \) and \( \gamma = h(R) \in \mathbb{Z}_p \). The cloud server then calculates \( \mu', \sigma \) according to the TPA protocol (Fig. 5.3), blinds both \( \mu' \) and \( \sigma \) by computing \( \mu = r_m + \gamma \mu' \mod p \), \( \varsigma = r_\sigma + \gamma \rho \mod p \) and \( \Sigma = \sigma g_1^\rho \). The cloud server then sends \( (\varsigma, \mu, \Sigma, R) \) as the response to the TPA. To verify it, the TPA computes \( \gamma = h(R) \) and then checks

\[
R \cdot e(\Sigma^\gamma, g) \overset{?}{=} e(\left(\prod_{i=s_1}^{s_2} H(W_i)^{\nu_i}\right)^\gamma \cdot u^\mu, v) \cdot e(g_1, g)^\varsigma. \tag{4.2}
\]

4.3 Security Weaknesses in Wang et al.’s TPA and ZKPA Protocols

4.3.1 Storage Correctness

It is originally believed that Wang et al. TPA protocol can achieve all the five design goals given in Sec. 4.2.1. However, below we show that the protocol cannot achieve the important goal of Storage Correctness: an attacker can arbitrarily modify the data but at the same time fool the auditor to believe that the data are well maintained by the cloud server. We describe the attack in two different scenarios: in the first scenario, the attacker (e.g. a hacker or internal employee of the cloud server) can learn the content of the user data file \( F \) and modify it; while in the second scenario, the attacker can modify the file \( F \) but does not know its content (e.g. a malicious programmer plants a bug in the software running on the cloud server).

**Scenario 1:** In this scenario, we assume the attacker (e.g. an employee of the cloud server) can access the user data file \( F \). The attacker first makes a copy of the
original file, and then modifies file blocks $m_i$ to $m_i^* = m_i + \beta_i$ for $1 \leq i \leq n$.

In the audit phase, after verifying the file tag $t = name \| SSig_{ssk}(name)$, the TPA sends a challenge $\{(i, \nu_i)\}_{i \in I}$ to the cloud server. Upon receiving the challenge, the cloud server would honestly compute $R = e(u, v)^r$ for a randomly chosen $r$ and $\sigma = \prod_{i \in I} \sigma_i^{\nu_i}$. However, as the data file has been modified, the cloud server would calculate

$$\mu^* = r + \gamma \mu'^* = r + \gamma \sum_{i = s_1}^{s_c} \nu_i m_i^*$$

$$= r + \gamma \sum_{i = s_1}^{s_c} \nu_i (m_i + \beta_i)$$

$$= \mu + \gamma \sum_{i = s_1}^{s_c} \nu_i \beta_i.$$

When the cloud server sends the response $(\mu^*, \sigma, R)$ to the TPA, the attacker intercepts the message and generates a new response as follows:

1. compute $\gamma = h(R)$, $\alpha = \gamma \sum_{i = s_1}^{s_c} \nu_i \beta_i$, $\hat{R} = R \cdot e(u^\alpha, v)$ and $\hat{\gamma} = h(\hat{R})$;

2. compute $\mu' = \sum_{i = s_1}^{s_c} \nu_i m_i$ and $\hat{\mu} = \mu'(\hat{\gamma} - \gamma) + \mu^*$.

The attacker then sends $(\hat{\mu}, \sigma, \hat{R})$ to the TPA who will perform the verification according to Equation (1). The verification will be successful as shown below.

$$\hat{R} \cdot e(\sigma^\hat{\gamma}, g) = e(u, v)^r e(u^\alpha, v) e\left((\prod_{i = s_1}^{s_c} \sigma_i^{\nu_i})^{\hat{\gamma}}, g\right)$$

$$= e(u^r, v) e(u^\alpha, v) e\left((\prod_{i = s_1}^{s_c} (H(W_i)u^{m_i})^{\nu_i})^{\hat{\gamma}}, g\right)$$

$$= e(u^r, v) e(u^\alpha, v) e\left((\prod_{i = s_1}^{s_c} H(W_i)^{\nu_i})^{\nu_i \hat{\gamma}}, g^x\right)$$

$$= e(u^r, v) e(u^\alpha, v) e\left((\prod_{i = s_1}^{s_c} H(W_i)^{\nu_i})^{\nu_i \hat{\gamma}}, u^\nu \mu'\right)$$

$$= e(u^r, v) e(u^\alpha, v) e\left((\prod_{i = s_1}^{s_c} H(W_i)^{\nu_i})^{\nu_i \hat{\gamma}} u^\nu \mu'\right)$$

$$= e\left((\prod_{i = s_1}^{s_c} H(W_i)^{\nu_i})^{\hat{\gamma}} u^\nu \mu' + \alpha + r, v\right)$$
4.3. Security Weaknesses in Wang et al.’s TPA and ZKPA Protocols

\[ e((\prod_{i=s_1}^{s_c} H(W_i)^{\nu_i})^{\tilde{\gamma}} u^\tilde{\mu'} + \mu^* - \gamma \mu', v) \]

\[ = e((\prod_{i=s_1}^{s_c} H(W_i)^{\nu_i})^{\tilde{\gamma}} u^{\tilde{\mu}}, v). \]

**Scenario 2:** In the second scenario, we assume the attacker (e.g., a malicious programmer who has planted a software bug on the cloud server) modifies the file block \( m_i \) to \( m_i^* = m_i + \beta_i \) for \( 1 \leq i \leq n \). However, the attacker only knows \( \beta_i \) (i.e., how the user data are modified) but not \( m_i \) or \( m_i^* \).

In the audit phase, the TPA and the cloud server honestly execute the auditing protocol. That is, TPA sends a challenge \( \{(i, \nu_i)\}_{i \in I} \) to the cloud server, and the cloud server sends back a response \( (\mu^*, R, \sigma) \) where \( R = e(u, v)^r \) for a randomly chosen \( r \), \( \sigma = \prod_{i \in I} \sigma_i^{\nu_i} \), and

\[ \mu^* = r + \gamma \mu^* = r + \gamma \sum_{i=s_1}^{s_c} \nu_i m_i^* = r + \gamma \sum_{i=s_1}^{s_c} \nu_i (m_i + \beta_i) \]

\[ = \mu + \gamma \sum_{i=s_1}^{s_c} \nu_i \beta_i. \]

The attacker intercepts the response \( (\mu^*, R, \sigma) \) from the cloud server to the TPA, and modifies \( \mu^* \) to \( \mu = \mu^* - \alpha \) where \( \alpha = \gamma \sum_{i=s_1}^{s_c} \nu_i \beta_i = h(R) \sum_{i=s_1}^{s_c} \nu_i \beta_i \). It is easy to see that by doing such a simple modification, the attacker derives a correct response with respect to the original message blocks \( \{m_i\}_{i \in I} \). In this way, the attacker can successfully fool the auditor to believe that the data file \( F \) is well preserved, while the real file on the cloud server has been modified.

### 4.3.2 Offline Guessing Attack

The TPA protocol presented in Fig. 5.3 is vulnerable to offline guessing attack [44], since the TPA can always guess whether \( \mu' = \tilde{\mu}' \), by checking

\[ e(\sigma, g) \equiv e((\prod_{i=s_1}^{s_c} H(W_i)^{\nu_i}) \cdot u^\tilde{\mu'}, v) \quad (4.3) \]

where \( \tilde{\mu}' \) is constructed from random coefficients chosen by the TPA in the challenge and the guessed message \( \{\tilde{m}_i\}_{s_1 \leq i \leq s_c} \).

In order to prevent the offline guessing attack, in the ZKPA protocol, two additional blind elements \( r_\sigma \) and \( \rho \) are introduced. It was believed that the ZKPA
4.4 Conclusion

The ZKPA protocol can effectively prevent the offline guessing attack. However, below we show that the ZKPA protocol is still vulnerable to offline guessing attack. Given $\text{chal} = \{(i, \nu_i)\}_{i \in I}$ and the response $(\varsigma, \mu, \Sigma, R)$, our attack works as follows:

1. for the guessed message $\{\tilde{m}_i\}_{s_1 \leq i \leq s_c}$, compute $\tilde{\mu}' = \Sigma_{i \in I} \nu_i \tilde{m}_i$ and $\tilde{r}_m = \mu - \gamma \tilde{\mu}' \mod p$;
2. compute $e(g_1, g)^{\tilde{r}_m} = R/e(u, v)^{\tilde{r}_m}$;
3. compute $e(g_1, g)^{\tilde{\rho}} = (e(g, g)^{\varsigma}/e(g_1, g)^{\tilde{r}_m})^{\gamma^{-1}}$ and $e(\tilde{\sigma}, g) = e(\Sigma, g)/e(g_1, g)^{\tilde{\rho}}$;
4. check the equation

$$e(\tilde{\sigma}, g) = e(\prod_{i = s_1}^{s_c} H(W_i)^{\nu_i}) \cdot u^{\tilde{\mu}'}, v).$$

If the equation holds, then output the guessed message $\{\tilde{m}_i\}_{s_1 \leq i \leq s_c}$; otherwise, goto step 1 for another guess.

The above attack essentially shows that the attacker can successfully remove the additional blind elements introduced in the ZKPA protocol and use equation (4.3) to locate the message $\{m_i\}_{s_1 \leq i \leq s_c}$.

4.4 Conclusion

In this chapter, we revisited a privacy-preserving third party auditing (TPA) cloud storage integrity checking protocol and its extended version for zero knowledge public auditing (ZKPA). We showed several security weaknesses in these protocols. It is still an open problem to design a ZKPA protocol that can prevent offline guessing attacks, and we leave it as our future work.
As we have seen in the previous chapters, with a rapid growth of data storages in the cloud, the data integrity check in a remote data storage system has become an important issue. A number of works have been done in the literature to address this issue. However, data privacy issues for data storage systems in the cloud have not been formally defined and investigated. We believe that these issues are equally important that the communication flows of integrity checking proofs from the cloud server should not reveal any useful information of the stored data. In this chapter, we introduce a novel definition of data privacy for the cloud by an Indistinguishability game. We found that the previous remote integrity checking schemes have not captured this feature. We also found that by using witness indistinguishable proofs to the communication flows from the cloud server, the data privacy is achievable. We provide a comprehensive study on data privacy in RIC proofs and a concrete scheme that guarantees both data integrity and privacy.

5.1 Introduction

Although data integrity check is a crucial and necessary step in cloud storage security, it is important that such a process should not introduce new vulnerabilities of unauthorized information leakage. The previous efforts in Remote Integrity Checking (RIC) accommodate several security features including data integrity and confidentiality, which mainly ensure secure maintenance of data. However, they do not cover the issue of data privacy, which means that the communication flows (RIC proofs) from the cloud server should not reveal any useful information to the adversary. Intuitively, by “privacy”, we mean that an adversary should not be able to distinguish which file has been uploaded by the client to the cloud server. We refer it as Indistinguishability (IND). We believe that it is very important to consider such
privacy issues adequately in protocol designs. Taking some existing TPA based RIC proofs [46, 44, 47] as an example, the proof sent by the cloud server to the auditor does not allow the auditor to recover the file, but the auditor can still distinguish which file (among a set of possible files) is involved in the RIC proof, which is clearly undesirable.

In this chapter, we propose an indistinguishability-based definition of data privacy (IND-Privacy, for short) for TPA based RIC protocols. We show that two recently published RIC schemes [44, 47] are insecure under our new definition, which means some information about the user file is leaked in the RIC proof. We then provide an new construction to demonstrate how IND-privacy can be achieved. We show that by applying the Witness Indistinguishability proof technique [29], we are able to achieve IND-privacy in RIC protocols. To the best of our knowledge, our construction is the first scheme that can achieve IND-privacy.

The rest of this chapter is organised as follows. In Section 2, we describe the security model and definition of data privacy for RIC proofs. In Section 3, we analyse the RIC protocols by Wang et al. and show why their RIC protocols fail to capture data privacy. In Section 4, we demonstrate how data privacy can be achieved with a witness indistinguishability proof. We also provide the definition of soundness for RIC proofs and show the soundness of our protocol based on witness indistinguishability proof. We conclude the paper in Section 6.

5.2 Definitions and Security Model

We will focus on TPA based Remote Integrity Checking (RIC) protocols in this chapter. The protocol involves three entities: the cloud server, the cloud user, and the third party auditor (TPA). The cloud user relies on the cloud server to store and maintain his/her data. Since the user no longer keeps the data locally, it is of critical importance for the user to ensure that the data are correctly stored and maintained by the cloud server. In order to avoid periodically data integrity verification, the user will resort to a TPA for checking the integrity of his/her outsourced data. We define a TPA protocol for cloud storage as a tuple of five algorithms:

- **KeyGen**: Taking as input a security parameter $\lambda$, the algorithm $\text{KeyGen}$ generates the public and private key pair $(pk, sk)$ of a cloud user (or data owner).
5.2. Definitions and Security Model

- **TokenGen**: Taking as input a file $F$ and the user private key $sk$, this algorithm generates a file tag $t$ (which includes a file name $name$) and an authenticator $\sigma$ for $F$. The file and file tag, as well as the authenticator are then stored in the cloud server.

- **Challenge**: Given the user public key $pk$ and a file tag $t$, this algorithm is run by the auditor to generate a random challenge $chal$ for the cloud server.

- **Respond**: Taking as input $(F, t, \sigma, chal)$, this algorithm outputs a proof $\mathcal{P}$, which is used to prove the integrity of the file.

- **Verify**: Taking as input $(pk, t, chal, \mathcal{P})$, the algorithm outputs either True or False.

**RIC Privacy.** We define the data privacy for RIC proofs via an indistinguishability game between a simulator $S$ (i.e. the cloud server or prover) and an adversary $A$ (i.e. the auditor or verifier).

**Setup**: The simulator runs $\text{KeyGen}$ to generate $(sk, pk)$ and passes $pk$ to the adversary $A$.

**Phase 1**: $A$ is allowed to make Token Generation queries. To make such a query, $A$ selects a file $F$ and sends it to $S$. $S$ generates a file tag $t$, an authenticator $\sigma$, and then returns $(t, \sigma)$ to $A$.

**Phase 2**: $A$ chooses two different files $F_0, F_1$ that has not appeared in Phase 1, and sends them to $S$. $S$ calculates $(t_0, \sigma_0)$ and $(t_1, \sigma_1)$ by running the $\text{TokenGen}$ algorithm. $S$ then tosses a coin $b \in \{0, 1\}$, and sends $t_b$ back to $A$. $A$ generates a challenge $chal$ and sends it to $S$. $S$ generates a proof $\mathcal{P}$ based on $(F_b, t_b, \sigma_b)$ and $A$’s challenge $chal$ and then sends $\mathcal{P}$ to $A$. Finally, $A$ outputs a bit $b'$ as the guess of $b$. The process is illustrated in Figure 5.1.

Define the advantage of the adversary $A$ as

$$Adv_A(\lambda) = |\Pr[b' = b] - 1/2|.$$

**Definition 5.1** We say an RIC proof has indistinguishability if for any polynomial-time algorithm $A$, $Adv_A(\lambda)$ is a negligible function of the security parameter $\lambda$.

---

1In the RIC protocols presented in this paper, the file $F$ is divided into multiple data blocks, and the authenticator $\sigma$ for $F$ is in fact a set of authenticators for individual data blocks.
5.3 Indistinguishability Analysis of Existing RIC Protocols

In this section, we will show that several RIC protocols proposed in the literature cannot satisfy our definition of IND-Privacy.

5.3.1 An RIC Protocol by Wang et al. [47]

In [47], Wang et al. presented an RIC protocol based on Merkle Hash Tree (MHT) [37]. Their protocol works as follows.

Setup Phase: The cloud user generates the keys and authentication tokens for the files as follows.

KeyGen: The cloud user runs KeyGen to generate the public and private key pair. Specifically, the user generates a random verification and signing key pair \((spk, ssk)\) of a digital signature scheme, and set the public key \(pk = (v, spk)\) and \(sk = (x, ssk)\) where \(x\) is randomly chosen from \(\mathbb{Z}_p\) and \(v = g^x\).

TokenGen: Given a file \(F = (m_1, m_2, \cdots, m_n)\), the client chooses a file name \(name\), a random element \(u \in G_1\) and calculates the file tag

\[
t = name||n||u||SSig_{ssk}(name||n||u),
\]

and authenticators \(\sigma_i = (H(m_i) \cdot u^{m_i})^x\) where \(H\) is a cryptographic hash function.
modeled as a random oracle. The client then generates a root $R$ based on the construction of Merkle Hash Tree (MHT) where the leave nodes of the tree are an ordered set of hash values $H(m_i)(i = 1, 2, \cdots, n)$. The client then signs the root $R$ under the private key $x$: $\sigma_{sk}(H(R)) = (H(R))^x$ and sends $\{F, t, \{\sigma_i\}, \sigma_{sk}(H(R))\}$ to the cloud server.

**Audit Phase**: The TPA first obtains the file tag $t$ and verifies the signature $SSig_{ssk}(name||n||u)$ by using $spk$. If the verification is successful, the TPA obtains $name$ and $u$.

**Challenge**: To generate $chal$, TPA picks a random subset $I = \{s_1, s_2, s_3, \ldots, s_c\}$ of set $[1, n]$, where $s_1 \leq \cdots \leq s_c$. Then, the TPA sends a challenge $chal = \{i, \nu_i\}_{i \in I}$ to the cloud server where $\nu_i$ is randomly selected from $\mathbf{Z}_p$.

**Response**: Upon receiving the challenge $chal = \{i, \nu_i\}_{i \in I}$, the cloud server computes $\mu = \sum_{i \in I} \nu_i m_i$ and $\sigma = \prod_{i \in I} \sigma_i^{\nu_i}$. The cloud server will also provide the verifier with a small amount of auxiliary information $\{\Omega_i\}_{i \in I}$, which are the node siblings on the path from the leaves $H(m_i)_{i \in I}$ to the root $R$ of the MHT. The server sends the proof $\mathcal{P} = \{\mu, \sigma, \{H(m_i), \Omega_i\}_{i \in I}, \sigma_{sk}(H(R))\}$ to the TPA.

**Verify**: Upon receiving the responses from the cloud server, the TPA generates the root $R$ using $\{H(m_i), \Omega_i\}_{i \in I}$, and authenticates it by checking

$$e(\sigma_{sk}(H(R)), g) = e(H(R), v).$$

If the authentication fails, the verifier rejects by emitting FALSE. Otherwise, the verifier checks

$$e(\sigma, g) = e((\prod_{i=s_1}^{s_c} H(m_i)\nu_i)u^\mu, v).$$

If the equation holds, output True; otherwise, output False.

**Indistinguishability Analysis**

It is easy to see that the above RIC protocol does not provide IND-Privacy. Let $\mathcal{A}$ denote an IND adversary which works as follows (also see Fig. 5.2).

- $\mathcal{A}$ chooses distinct files $F_0 = (m_1^{(0)}, \cdots, m_n^{(0)})$ and $F_1 = (m_1^{(1)}, \cdots, m_n^{(1)})$ where $m_i^{(0)} \neq m_i^{(1)}$. 
5.3 Indistinguishability Analysis of Existing RIC Protocols

• $S$ generates $(t_0, \{\sigma_i^{(0)}\}, \text{sig}_{sk}(H(R^{(0)})))$ and $(t_1, \{\sigma_i^{(1)}\}, \text{sig}_{sk}(H(R^{(1)})))$ for $F_0$ and $F_1$ respectively. $S$ then chooses a random $b \in \{0, 1\}$ and sends $t_b$ back to $A$.

• $A$ chooses a random challenge $\text{chal} = \{i, \nu_i\}_{i \in I}$.

• $S$ computes and sends to $A$ the response

$$P = (\mu^{(b)}, \sigma^{(b)}, \{H(m_i^{(b)}), \Omega^{(b)}_i\}_{i \in I}, \text{sig}_{sk}(H(R^{(b)}))).$$

• $A$ chooses an index $i \in I$ and calculates $H(m_i^{(0)})$ and compare it with the received $H(m_i^{(b)})$. If they are equal, output 0; otherwise, output 1.

**Probability Analysis.** It is easy to see that $A$ has an overwhelming probability to guess the value of $b$ correctly since the probability that

$$m_i^{(0)} \neq m_i^{(1)} \land H(m_i^{(0)}) = H(m_i^{(1)})$$

is negligible since the hash function is assumed to be a random oracle in [47].

5.3.2 Another Privacy Preserving RIC Protocol by Wang et al. [44]

In [44], Wang et al. introduced a new RIC protocol. Compared with the RIC protocol presented above, this new protocol aims to achieve the additional property of privacy preserving (i.e. the TPA cannot learn the content of the file in the auditing process).

Let $(p, G_1, G_2, G_T, e, g_1, g, H, h)$ be the system parameters as introduced above. Wang et al.’s privacy-preserving public auditing scheme works as follows (also see Fig. 5.3):

**Setup Phase:**

**KeyGen:** The cloud user runs KeyGen to generate the public and private key pair. Specifically, the user generates a random verification and signing key pair $(spk, ssk)$ of a digital signature scheme, a random $x \leftarrow \mathbb{Z}_p$, a random element $u \leftarrow G_1$, and computes $v \leftarrow g^x$. The user secret key is $sk = (x, ssk)$ and the user public key is $pk = (spk, v, u)$. 
5.3. Indistinguishability Analysis of Existing RIC Protocols

\[ A \]

Generate two distinct files:
\[ F_0 = (m_1^{(0)}, \ldots, m_n^{(0)}) \]
\[ F_1 = (m_1^{(1)}, \ldots, m_n^{(1)}) \]
\[ F_0, F_1 \rightarrow \text{Compute} (t_0, \{\sigma_i^{(0)}\}, \text{sig}_{sk}(H(R^{(0)}))) \]
\[ \text{and} (t_1, \{\sigma_i^{(1)}\}, \text{sig}_{sk}(H(R^{(1)}))) \]
Randomly select \( b \in \{0, 1\} \)
\[ t_b \leftarrow \text{chal} = \{i, \nu_i\}_{i \in I} \]
\[ \text{chal} \rightarrow \text{Compute} \mu^{(b)} = \sum_{i \in I} \nu_i m_i^{(b)} \]
\[ \sigma^{(b)} = \prod_{i \in I} (\sigma_i^{(b)})^{\nu_i} \]
Prepare \( \{\Omega_i\}_{i \in I} \)
Set the proof
\[ P = \mu^{(b)}, \sigma^{(b)}, \{H(m_i^{(b)}), \Omega_i^{(b)}\}_{i \in I}, \text{sig}_{sk}(H(R^{(b)})) \]

\[ S \]

Choose \( i \in I \)
Calculate \( H(m_i^{(0)}) \)
If \( H(m_i^{(0)}) = H(m_i^{(b)}) \)
return 0
Otherwise, return 1

Figure 5.2: Indistinguishability analysis on Wang et al.’s RIC Protocol [47].

**TokenGen**: Given a data file \( F = (m_1, \ldots, m_n) \), the user first chooses uniformly at random from \( \mathbb{Z}_p \) a unique identifier \( name \) for \( F \). The user then computes authenticator \( \sigma_i \) for each data block \( m_i \) as \( \sigma_i \leftarrow (H(W_i) \cdot u^{m_i})^x \in G_1 \) where \( W_i = name \| i \).
Denote the set of authenticators by \( \phi = \{\sigma_i\}_{1 \leq i \leq n} \). Then the user computes \( t = name \| SSig_{ssk}(name) \) as the file tag for \( F \), where \( SSig_{ssk}(name) \) is the user’s signature on \( name \) under the signing key \( ssk \). It was assumed that the TPA knows the number of blocks \( n \). The user then sends \( F \) along with the verification metadata \( (\phi, t) \) to the cloud server and deletes them from local storage.

**Audit Phase**: The TPA first obtains the file tag \( t \) and verifies the signature \( SSig_{ssk}(name) \) by using \( spk \). The TPA quits by emitting \( \bot \) if the verification fails. Otherwise, the TPA recovers \( name \).

**Challenge**: The TPA generates a challenge \( chal \) for the cloud server as follows: first picks a random \( c \)-element subset \( I = \{s_1, \ldots, s_c\} \) of set \([1, n]\), and then for each
5.3. Indistinguishability Analysis of Existing RIC Protocols

TPA | Cloud Server
---|---
1. Retrieve the file tag $t$ and verify its signature. Quit if fail. | 3. Compute $\mu' = \sum_{i \in I} \nu_i m_i$ and $\sigma = \prod_{i \in I} \sigma_i^\nu_i$;
2. Generate a random challenge $\text{chal} = \{i, \nu_i\}_{i \in I}$ | 4. Randomly pick $r \leftarrow \mathbb{Z}_p$ and compute $R = e(u, v)^r$ and $\gamma = h(R)$;
6. Compute $\gamma = h(R)$ and then verify $(\mu, \sigma, R)$. | 5. Compute $\mu = r + \gamma \mu' \mod p$

Figure 5.3: The third party auditing protocol by Wang et al. [44].

element $i \in I$, chooses a random value $\nu_i \in \mathbb{Z}_p$. The TPA sends $\text{chal} = \{(i, \nu_i)\}_{i \in I}$ to the cloud server.

**Response:** Upon receiving the challenge $\text{chal}$, the server generates a response to prove the data storage correctness. Specifically, the server chooses a random element $r \leftarrow \mathbb{Z}_p$, and calculates $R = e(u, v)^r \in G_T$. Let $\mu'$ denote the linear combination of sampled blocks specified in $\text{chal}$: $\mu' = \sum_{i \in I} \nu_i m_i$. To blind $\mu'$ with $r$, the server computes $\mu = r + \gamma \mu' \mod p$, where $\gamma = h(R) \in \mathbb{Z}_p$. Meanwhile, the server also calculates an aggregated authenticator $\sigma = \prod_{i \in I} \sigma_i^\nu_i$. It then sends $(\mu, \sigma, R)$ as the response to the TPA.

**Verify:** Upon receiving the response $(\mu, \sigma, R)$ from the cloud server, the TPA validates the response by first computing $\gamma = h(R)$ and then checking the following verification equation

$$R \cdot e(\sigma^\gamma, g) \overset{?}{=} e((\prod_{i = i_1}^{i_2} H(W_i)^{\nu_i})^\gamma \cdot u^\mu, v). \quad (5.1)$$

The verification is successful if the equation holds.

**Indistinguishability Analysis**

In [44], it has been shown that the RIC proof is privacy preserving. That is, the TPA cannot recover the file $F$ from the proof. This is done by concealing the value
5.3. Indistinguishability Analysis of Existing RIC Protocols

Generate two distinct files:
\[ F_0 = (m_1^{(0)}, \ldots, m_n^{(0)}) \]
\[ F_1 = (m_1^{(1)}, \ldots, m_n^{(1)}) \]

\[ \text{chal} = \{i, \nu_i\}_{i \in I} \]

\[ \text{chal} \xrightarrow{t_b} \]

\[ \text{chal} \xrightarrow{t_b} \]

\[ \mu_0' = \sum_{i \in I} (\nu_i m_i^{(0)}) \]

Check if
\[ e(\prod_{i \in I} H(W_i)^{\nu_i} \mu_0', v) = e(\sigma^{(b)}, g). \]

If true, output 0; otherwise, output 1.

Figure 5.4: Indistinguishability analysis on Wang et al. RIC Protocol [44].

of \( \mu' \). However, we found that such a treatment could not guarantee that there is no information leakage during the auditing process. Below we show that Wang et al.’s scheme cannot achieve indistinguishability. Let \( \mathcal{A} \) denote an IND adversary which works as follows (also see Fig. 5.4).

- \( \mathcal{A} \) chooses two distinct files \( F_0 = (m_1^{(0)}, \ldots, m_n^{(0)}) \) and \( F_1 = (m_1^{(1)}, \ldots, m_n^{(1)}) \) such that \( m_i^{(0)} \neq m_i^{(1)} \) for \( 1 \leq i \leq n \).

- \( \mathcal{S} \) generates \( (t_0, \{\sigma_i^{(0)}\}) \) and \( (t_1, \{\sigma_i^{(1)}\}) \) for \( F_0 \) and \( F_1 \) respectively. \( \mathcal{S} \) then chooses a random \( b \in \{0, 1\} \) and sends \( t_b \) back to \( \mathcal{A} \).

- After receiving the tag \( t_b \), \( \mathcal{A} \) chooses a random challenge \( \text{chal} = \{i, \nu_i\}_{i \in I} \).

- \( \mathcal{S} \) computes and sends to \( \mathcal{A} \) the response \( \mathcal{P} = (\mu, \sigma^{(b)}, R) \).

- \( \mathcal{A} \) computes \( \mu_0' = \sum_{i \in I} (\nu_i m_i^{(0)}) \) and checks if
\[ e(\prod_{i \in I} (H(W_i))^{\nu_i} \mu_0', v) = e(\sigma^{(b)}, g). \]

If it is true, return 0; otherwise, return 1.
5.3. Indistinguishability Analysis of Existing RIC Protocols

Probability Analysis. If \( b = 0 \), then \( \sigma^{(b)} = \sigma^{(0)} \) and the equation
\[
e(\prod_{i \in I}(H(W_i))^{\nu_i}w^\mu_i, v) = e(\sigma^{(0)}, g)
\]
always holds. On the other hand, if \( b = 1 \), then \( \sigma^{(b)} = \sigma^{(1)} \) and
\[
e(\prod_{i \in I}(H(W_i))^{\nu_i}w^\mu_i, v) = e(\sigma^{(1)}, g)
\]
holds only when
\[
\mu'_0(= \sum_{i \in I}(\nu_i m^{(0)}_i)) = \mu'_1(= \sum_{i \in I}(\nu_i m^{(1)}_i)),
\]
which happens only with probability \( 1/p \) for randomly selected \( \{\nu_i\}_{i \in I} \) since \( m^{(0)}_i \neq m^{(1)}_i \) for all \( i \in I \). Therefore, \( A \) has an overwhelming probability to guess the value of \( b \) correctly.

Indistinguishability of the Zero-Knowledge RIC Protocol in [44].

In [44], Wang et al. also extended their privacy-preserving RIC protocol presented above to a Zero-Knowledge (ZK) RIC Protocol.

The Setup phase is almost the same as in the original RIC protocol, except that an additional generator \( g_1 \in G_1 \) is introduced in the user public key. In the Audit phase, upon receiving the challenge \( chal = \{(i, \nu_i)\}_{i \in I} \), the cloud server selects three random blind elements \( r_m, r_\sigma, \rho \in \mathbb{Z}_p \), and calculates \( R = e(g_1, g)^{r_\sigma}e(u, v)^{r_m} \in G_T \) and \( \gamma = h(R) \in \mathbb{Z}_p \). The cloud server then computes \( \mu', \sigma \) according to the original RIC protocol, blinds both \( \mu' \) and \( \sigma \) by computing \( \mu = r_m + \gamma \mu' \mod p, \varsigma = r_\sigma + \gamma \rho \mod p \), and \( \Sigma = \sigma g_1^\rho \). The cloud server then sends \( (\varsigma, \mu, \Sigma, R) \) as the response to the auditor. To verify it, the auditor computes \( \gamma = h(R) \) and then checks
\[
R \cdot e(\Sigma^\gamma, g) = e(\prod_{i=s_1}^{s_\varsigma} H(W_i)^{\nu_i})^\gamma \cdot w^\mu, v) \cdot e(g_1, g)^\varsigma. \tag{5.2}
\]

However, the above Zero-Knowledge RIC protocol does not achieve IND-Privacy either. Consider the following attack performed by an IND adversary \( A \).

1. \( A \) chooses two distinct files \( F_0 = (m^{(0)}_1, \cdots, m^{(0)}_n) \) and \( F_1 = (m^{(1)}_1, \cdots, m^{(1)}_n) \) such that \( m^{(0)}_i \neq m^{(1)}_i \) for \( 1 \leq i \leq n \).

2. \( S \) generates \( (t_0, \{\sigma^{(0)}_i\}) \) and \( (t_1, \{\sigma^{(1)}_i\}) \) for \( F_0 \) and \( F_1 \) respectively. \( S \) then chooses a random \( b \in \{0, 1\} \) and sends \( t_b \) back to \( A \).
3. After receiving the tag $t_b$, $A$ chooses a random challenge $chal = \{i, \nu_i\}_{i \in I}$.

4. $S$ computes and sends to $A$ the response $P = (\varsigma, \mu, \Sigma, R)$.

5. $A$ computes $\mu'_0 = \sum_{i \in I} \nu_i m^{(0)}_i$ and $\tilde{r}_m = \mu - \gamma \mu'_0 \mod p$ where $\gamma = h(R)$. Then $A$ computes $e(g_1, g)^{\tilde{r}_m} = R/e(u, v)^{\tilde{r}_m}$, $e(g_1, g)^{\tilde{r}} = (e(g_1, g)^{\varsigma} / e(g_1, g)^{\tilde{r}_m})^{\gamma^{-1}}$, and $e(\tilde{\sigma}, g) = e(\Sigma, g) / e(g_1, g)^{\tilde{r}}$.

6. $A$ then checks the equation

$$e(\tilde{\sigma}, g)^2 = e((\prod_{i=S_1}^{S_c} H(W_i)^{\nu_i}) \cdot u^{\mu'_0}, v).$$

(5.3)

If the equation holds, $A$ outputs 0, otherwise, outputs 1.

If $b = 0$, then $e(\tilde{\sigma}, g) = e(\sigma^{(0)}, g)$, and hence Equation 5.3 always holds. On the other hand, if $b = 1$, then Equation 5.3 holds only when $\mu'_0(= \sum_{i \in I}(\nu_i m^{(0)}_i)) = \mu'_1(= \sum_{i \in I}(\nu_i m^{(1)}_i))$, which happens only with probability $1/p$.

5.4 A New RIC Protocol with IND-Privacy

In order to achieve the IND-privacy, we adopt the Witness Indistinguishable Proof of Knowledge technique proposed by Groth and Sahai [29]. Their method can be applied to pairing groups. Our goal is to protect both the file and the corresponding authenticator so that the adversary cannot learn any information about the file.

Similar to Wang et al.’s scheme [44] reviewed in Section 5.3.2, our scheme is still based on the “aggregate authenticator” introduced by Shacham and Waters [42]. That is, the cloud server will prove that the equation

$$e(\sigma, g) = e((\prod_{i=S_1}^{S_c} H(W_i)^{\nu_i}) u^{\mu'}, v)$$

holds, where $\mu' = \sum_{i \in I} \nu_i m_i$ and $\sigma = \prod_{i \in I} \sigma_i^{\nu_i}$. We will treat $(u^{\mu'}, \sigma)$ as the witness when applying the Groth-Sahai proof system, and rewrite Equation 5.4 as follows

$$e(\sigma, g) e(u^{\mu'}, v^{-1}) = e((\prod_{i=S_1}^{S_c} H(W_i)^{\nu_i}), v).$$

(5.5)

In order to protect the privacy of $\mu'$ (or $u^{\mu'}$) and $\sigma$, the user computes an additional commitment key $\vec{u} = (u_1, u_2)$ of the form

$$u_1 = (u, u^\alpha), \quad u_2 = (u^\tau, u^{\tau \alpha}),$$
where $\alpha, \tau$ are selected from $\mathbb{Z}_p$ at random and $u$ is the same generator of $G_1$ used in Wang et al.’s scheme. This additional commitment key $\vec{u}$ is now part of the user public key. To hide $u^{\mu'}$ and $\sigma$, the Cloud Server computes the commitments $\vec{c} = (c_1, c_2)$ as

$$c_1 = (c_{11}, c_{12}) = (u^{r_{11}+r_{12}\tau}, u^{\alpha(r_{11}+r_{12}\tau)}\sigma),$$
$$c_2 = (c_{21}, c_{22}) = (u^{r_{21}+r_{22}\tau}, u^{\alpha(r_{21}+r_{22}\tau)}u^{\mu'}).$$

where $r_{i,j}$ ($i, j \in \{1, 2\}$) are randomly selected from $\mathbb{Z}_p$. The Cloud Server also computes

$$\vec{\pi} = (\pi_1, \pi_2) = ((1, g^{r_{11}v-r_{21}}), (1, g^{r_{12}v-r_{22}})).$$

and sends $(\vec{c}, \vec{\pi})$ as the response to the TPA.

TPA then verifies the response sent by the Cloud Server by checking the equality of

$$\vec{c} \bullet \begin{pmatrix} 1 & g & v^{-1} \\ 1 & v^{-1} \end{pmatrix} = \nu_T(t_T)(\vec{u} \bullet \vec{\pi}) \tag{5.6}$$

where $t_T$ represents the right hand side of Equation (5.5) and $\nu_T$ denotes the following transformation:

$$t_T \rightarrow \begin{pmatrix} 1 & 1 \\ 1 & t_T \end{pmatrix}.$$

The “•” operation is defined as follows: define a function

$$F((x_1, x_2), (y_1, y_2)) = \begin{pmatrix} e(x_1, y_1) & e(x_1, y_2) \\ e(x_2, y_1) & e(x_2, y_2) \end{pmatrix}$$

for $(x_1, x_2) \in G_1^2$ and $(y_1, y_2) \in G_2^2$, and the “•” operation is defined as

$$\vec{x} \bullet \vec{y} = F(x_1, y_1)F(x_2, y_2).$$

**Correctness.** To verify Equation (5.6),

Left $= \vec{c} \bullet \begin{pmatrix} 1 & g & v^{-1} \\ 1 & v^{-1} \end{pmatrix} = \begin{pmatrix} e(c_{11}, 1) & e(c_{11}, g) \\ e(c_{12}, 1) & e(c_{12}, g) \end{pmatrix} \begin{pmatrix} e(c_{21}, 1) & e(c_{21}, v^{-1}) \\ e(c_{22}, 1) & e(c_{22}, v^{-1}) \end{pmatrix}$

Right $= \nu_T(t_T)F(u_1, \pi_1)F(u_2, \pi_2)$

$$= \begin{pmatrix} 1 & 1 \\ 1 & t_T \end{pmatrix} \begin{pmatrix} 1 & e(u, g^{r_{11}v-r_{21}}) \\ 1 & e(u^{\alpha}, g^{r_{11}v-r_{21}}) \end{pmatrix} \begin{pmatrix} 1 & e(u^{\tau}, g^{r_{12}v-r_{22}}) \\ 1 & e(u^{\tau \alpha}, g^{r_{12}v-r_{22}}) \end{pmatrix}$$

and we have

$$e(c_{11}, 1)e(c_{21}, 1) = 1 \cdot 1 \cdot 1$$
\[ e(c_{12}, 1)e(c_{22}, 1) = 1 = 1 \cdot 1 \]

\[ e(c_{11}, g)e(c_{21}, v^{-1}) = e(u^{r_{11}+r_{12}\tau}, g) \cdot e(u^{r_{21}+r_{22}\tau}, v^{-1}) \]
\[ = e(u, g^{r_{11}})e(u^{r_{21}}, v^{-1})e(u^{\tau}, g^{r_{12}})e(u^{\tau}, v^{-r_{22}}) \]
\[ = e(u^{r_{11}+r_{12}\tau}, g)e(u^{r_{21}+r_{22}\tau}, v^{-1}) \]

\[ e(c_{12}, g)e(c_{22}, v^{-1}) = e(u^{\alpha(r_{11}+r_{12}\tau)}, g)e(u^{\alpha(r_{21}+r_{22}\tau)}, v^{-1})e(\sigma, g)e(u^{\mu'}, v^{-1}) \]
\[ = t_T e(u^{\alpha r_{11}}, g)e(u^{\alpha r_{21}}, v^{-1})e(u^{\alpha r_{22}}, v^{-1}) \]
\[ = t_T e(u^{\alpha}, g^{r_{11}r_{21}})e(u^{\alpha r_{22}}, g^{r_{12}r_{22}}) \]

### 5.4.1 IND-Privacy of Our New Scheme

Below we show that our new RIC protocol has the IND-Privacy under the symmetric external Diffie-Hellman (SXDH) assumption \[29\]. Let \( g_k = (\lambda, p, G_1, G_2, G_T, e, g_1, g_2) \) define a bilinear map \( e : G_1 \times G_2 \to G_T \) where \( g_b \) is a generator of \( G_b \) for \( b = \{0, 1\} \). The SXDH assumption holds if for any polynomial time algorithm \( A \) and any \( b \in \{1, 2\} \) we have

\[ | \Pr[x, y \leftarrow Z_p^* : A(g_k, g_b^x, g_b^y, g_b^{xy}) = 1] - \Pr[x, y, r \leftarrow Z_p^* : A(g_k, g_b^x, g_b^y, g_b^r) = 1]| \leq \epsilon \]

where \( \epsilon \) is negligible in the security parameter \( \lambda \).

**Theorem 5.1** *Our new RIC protocol has IND-Privacy if the SXDH problem is hard.*

**Proof:** Let \( A \) denote an adversary who has a non-negligible advantage \( \epsilon \) in winning the IND game, we construct another algorithm \( B \) which can solve the SXDH problem also with a non-negligible probability.

\( B \) receives a challenge \( g_k, A = u^x, B = u^y, C = u^z \) where \( g_k = (p, G_1, G_2, G_T, e, u, g) \) and \( z \) is either \( xy \) or a random element \( \xi \) in \( Z_p \). \( B \) sets up the IND game for \( A \) as follows

1. \( B \) uses the information in \( g_k \) to generate all the systems parameters and public/private keys as described in Wang et al.’s TPA scheme (Sec. 5.3.2).

2. \( B \) also sets the values of the commitment key \( \bar{u} = (u_1, u_2) \) in our scheme as \( u_1 = (u, A) \) and \( u_2 = (B, C) \).
Upon receiving the two files $F_0$ and $F_1$ from $A$, $B$ simulates the game as follows. $B$ generates a random file identifier $name$ and the file tag $t = name\|SSig_{ssk}(name)$, and uses $name$ and the secret key $x$ to compute the authenticators $\{\sigma_i^{(0)}\}$ (for $F_0$) and $\{\sigma_i^{(1)}\}$ (for $F_1$) honestly. After that, $B$ tosses a random coin $b \leftarrow \{0,1\}$, and sends the file tag $t$ back to $A$. Upon receiving the challenge $chal$ from $A$, $B$ computes $\mu'_0, \mu'_1$, and the corresponding aggregated authenticators $\sigma^{(0)}$ and $\sigma^{(1)}$ honestly. $B$ then generates the response to $A$ as follows.

1. Randomly choose $r_{11}, r_{12}, r_{21}, r_{22}$ from $\mathbb{Z}_p$.
2. Compute $c_{11} = u^{r_{11}}B^{r_{12}}, c_{12} = A^{r_{11}}C^{r_{12}}\sigma^{(b)}, c_{21} = u^{r_{21}}B^{r_{22}}, c_{22} = A^{r_{21}}C^{r_{22}}u^{\mu'_b}$.
3. Compute $\vec{\pi} = (\pi_1, \pi_2) = ((1, g^{r_{11}}u^{r_{21}}), (1, g^{r_{12}}u^{r_{22}}))$.

$B$ then sends the response $(\vec{c}, \vec{\pi})$ to $A$. If $A$ outputs $b'$ such that $b' = b$, then $B$ outputs 1; otherwise $B$ outputs 0.

Case 1: $z = xy$. In this case, the distribution of the response $(\vec{c}, \vec{\pi})$ is identically to that of a real response, and hence we have

$$\Pr[b' = b] = 1/2 + \epsilon.$$ 

Case 2: $z = \xi$. In this case, the commitment scheme is perfectly hiding. That is, for a valid proof $(\vec{c}, \vec{\pi})$ satisfying equation 5.6, it can be expressed as a proof for $(u^{\mu'_0}, \sigma_0)$ (with randomness $(r_{11}^0, r_{12}^0, r_{21}^0, r_{22}^0)$), or a proof for $(u^{\mu'_1}, \sigma_1)$ (with randomness $(r_{11}^1, r_{12}^1, r_{21}^1, r_{22}^1)$). Therefore, we have

$$\Pr[b' = b] = 1/2.$$

Combining both cases, we have

$$\Pr[B(gk, u^x, u^y, u^{xy}) = 1]) - \Pr[B(gk, u^x, u^y, u^{\xi}) = 1])
= \Pr[b' = b | z = xy] - \Pr[b' = b | z = \xi]
= \epsilon.$$ 

5.4.2 Soundness of the Protocol

Having shown the IND feature of the protocol, we have seen that adversary $A$ cannot distinguish the file that has been used by the cloud server in an RIC proof. The
remaining task is to prove the “soundness” of the protocol. We say a protocol is sound if it is infeasible for the cloud server to change a file without being caught by the TPA in an auditing process. We formally define the soundness games between a simulator $\mathcal{B}$ and an adversary $\mathcal{A}$ (i.e. the cloud server) as follows.

- **Key Generation.** $\mathcal{B}$ generates a user key pair $(sk, pk)$ by running $\text{KeyGen}$, and then provides $pk$ to $\mathcal{A}$.

- **Phase 1.** $\mathcal{A}$ can now interact with $\mathcal{B}$ and make at most $\ell$ Token Generation queries. In each query, $\mathcal{A}$ sends a file $F_i = \{m_{i1}, m_{i2}, \ldots, m_{in}\} (1 \leq i \leq \ell)$ to $\mathcal{B}$, which responds with the corresponding file tag $t_i$ and authentication tokens $\phi_i = \{\sigma_{ij}\} (1 \leq j \leq n)$.

- **Phase 2.** $\mathcal{A}$ outputs a file $F^*$ and a file tag $t^*$ such that $t^* = t_i$ but $F^* \neq F_i$ for an $i \in [1, \ell]$ (i.e. at least one message block of $F_i$ has been modified by $\mathcal{A}$). $\mathcal{B}$ then plays the role as the verifier and executes the RIC protocol with $\mathcal{A}$ by sending a challenge $\text{chal}^* = \{j, \nu_j\}$ which contains at least one index $j$ such that $F^*$ differs from $F_i$ in the $j$-th message block.

- **Decision.** Based on the proof $\mathcal{P}^*$ computed by $\mathcal{A}$, $\mathcal{B}$ makes a decision which is either True or False.

**Definition 5.2** We say a witness indistinguishable RIC protocol is $\epsilon$-sound if

$$\Pr[\mathcal{B} \text{ outputs True}] \leq \epsilon.$$ 

Below we prove that our RIC protocol is sound under the co-CDH assumption. Let $(p, G_1, G_2, G_T, e, g_1, g)$ be the system parameters defined as above where $e: G_1 \times G_2 \rightarrow G_T$ is a bilinear map. Let $\psi: G_2 \rightarrow G_1$ denote an efficiently computable isomorphism such that $\psi(g) = g_1$.

**Computational co-Diffie-Hellman (co-CDH) Problem** on $(G_1, G_2)$: Given $g_1, u \in G_1$ and $g, g^a \in G_2$ as input where $g_1$ and $g$ are generators of $G_1$ and $G_2$ respectively, $a$ is randomly chosen from $\mathbb{Z}_p$, and $u$ is randomly chosen from $G_1$, compute $u^a \in G_1$.

**Theorem 5.2** The proposed witness indistinguishable RIC protocol is $\text{negl}(\lambda)$-sound, where $\text{negl}(\lambda)$ is a negligible function of the security parameter $\lambda$, if the co-CDH problem is hard.
Proof: Our proof is by contradiction. We show that if there exists an adversary $A$ that can win the soundness game with a non-negligible probability, then we can construct another adversary $B$ which can solve the co-CDH problem also with a non-negligible probability.

According to the soundness game, $F^* = \{m_1^*, m_2^*, \ldots, m_n^*\}$ must be different from the original file $F_i = \{m_1, m_2, \ldots, m_n\}$ associated with $t^*$ (or $t_i$). That means there must exist an $i \in [1, n]$ such that $m_i^* \neq m_i$. Below we show that if $A$ can pass the verification for $\mu^*$ where $\mu^* = \sum_{i \in I} \nu_i m_i^*$ and at least one of $\{m_i^*\}_{i \in I}$ is modified by $A$, then $B$ can solve the co-CDH problem.

$B$ is given an instance of the co-CDH problem $(g_1, u, g, g^x)$ where $g_1$ and $g$ are generators of $G_1$ and $G_2$ respectively such that $\psi(g) = g_1$, and $u$ is a random element in $G_1$. $B$’s goal is to compute $u^x \in G_1$. $B$ honestly generates the signing key pair $(spk, ssk)$, $\alpha, \tau \in \mathbb{Z}_p$ and the commitments key $\vec{u}_1 = (u, u^{\alpha})$, $\vec{u}_2 = (u^\tau, u^{\tau\alpha})$ according to the protocol specification. $B$ also sets $g^x$ as value of $\nu$ in the user public key, but the value of $x$ is unknown to $B$. $B$ then simulates the game as follows.

**Phase 1:** $B$ answers $A$’s queries in Phase 1 as follows. To generate a file tag $t_i$ for a file $F_i$, $B$ first chooses $\text{name}$ at random and generates the file tag $t_i = \text{name} \parallel \text{SSig}_{ssk}(\text{name})$. For each block $m_j(1 \leq j \leq n)$ in $F_i$, $B$ chooses at random $r_j \in R \mathbb{Z}_p$ and programs the random oracle $H(W_j) = g_1^{r_j} / u^{m_j}$.

$B$ then computes

$$\sigma_j = (H(W_j)u^{m_j})^x = (g_1^{r_j})^x = (\psi(v))^r.$$ 

It is easy to verify that $\sigma_j$ is a valid authenticator with regards to $m_j$.

**Phase 2:** Suppose $A$ outputs a response $P^* = (\vec{c}, \vec{\pi})$ for $t^*$, $\{m_i^*\}_{i \in I}$ and challenges $\{\nu_i\}_{i \in I}$ where at least one $m_i^*$ has been modified by the adversary. Denote $\mu^* = \sum_{i \in I} \nu_i m_i^*$.

Let $\mu = \sum_{i \in I} \nu_i m_i$ and $\sigma = \prod_{i \in I} \sigma_i^{\nu_i}$ denote the original file and authenticator that satisfy

$$e(\sigma, g) = e((\prod_{i \in I} H(W_i)^{\nu_i})u^{\mu}, v).$$  

(5.7)

$B$ then uses the value of $\tau$, which is used to generate the commitment key $\vec{u}$, to obtain $\sigma^* = c_{12}/c_{11}^\alpha$ and $u^{\mu^*} = c_{22}/c_{21}^\alpha$ from the commitment $\vec{c} = (c_1, c_2)$. Since $P^*$
can pass the verification, from Equation 5.6 we have
\[ e(\sigma^*, g) = e((\prod_{i \in I} H(W_i)_{\nu_i}) u^{\mu^*}, v). \] (5.8)

From Equation 5 and Equation 6, we can obtain
\[ e(\sigma^*/\sigma, g) = e(u^{\mu^* - \mu}, v). \]

Since \( B \) chooses the challenges \( \nu_i \) randomly, with overwhelming probability \( 1 - 1/p \),
\( \mu^* = \sum_{i \in I} \nu_i m_i^* \neq \sum_{i \in I} \nu_i m_i = \mu \), and hence \( B \) can obtain
\[ u^x = (\sigma^*/\sigma)^{1/\mu^* - \mu}. \]

## 5.5 Conclusion

In this chapter, we studied a new desirable security notion called IND-Privacy for remote data integrity checking protocols for cloud storage. We showed that several well-known RIC protocols cannot provide this property, which could render the privacy of user data exposed in an auditing process. We then proposed a new RIC protocol which can provide IND-Privacy. Our construction is based on an efficient Witness Indistinguishable Proof of Knowledge system. In addition, we also proved the soundness of the newly proposed protocol, which means the cloud server cannot modify the user data without being caught by the third party auditor in an auditing process.
Chapter 6

Conclusion

6.1 What Have We Done

In this thesis, we have presented a comprehensive study on RIC. We started by giving some preliminary knowledge and tools which are utilised in this thesis, and provided a brief overview on the related work such as PDP, POR and POW along with a revisit of current RIC schemes. The major contributions of this work lie in the new discoveries including a comprehensive analysis on a well-known RIC scheme [44] and introduction of the new notion of IND-privacy. In the former one, as described in Chapter 4 we described two attack scenarios to show that the RIC schemes proposed in [44] are flawed. This discovery is significant as it implies that all existing RIC schemes based on the Shacham-Waters POR scheme [42] will have the same problems.

In the latter one, as presented in Chapter 5 we defined the formal model of Instinguishability and IND-privacy. This is for the first time that IND-privacy is formally studied in RIC protocols. We found that the existing RIC schemes cannot hold the IND-Privacy, which means that the IND-distinguisher is able to differentiate two different files from the RIC proof of the cloud server. Unfortunately, this problem has never been found in the previous study of RIC. We found that IND-privacy can be achieved with Witness Indistinguishable (WI) proofs. The method to adopt WI proofs is to mask the RIC proof with the commitment scheme introduced by Groth and Sahai [29]. Their WI proofs for pairing groups were utilised in our RIC schemes. We show that Wang et al.’s RIC protocol [44] indeed captures the IND-privacy when our technique has been applied.

Our finding of IND-privacy provides a new tool to cloud security. We predict that this tool will be widely adopted in analysis of cloud security protocols, beside RIC protocols.
6.2 Future Work

We believe that our new notion of RIC privacy can also be applied to other related protocols such as PDP, POW and POR. It is worth exploring.

As mentioned in Chapter 4, it is still an open problem of how to design a zero knowledge public auditing protocol that can present offline guessing attacks. We will leave it as our future work.


