1-2022

Authenticated data redaction with accountability and transparency

Jinhua MA

Xinyi HUANG

Yi MU

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 19, NO. 1, JANUARY/FEBRUARY 2022, 149-160.
DOI: 10.1109/TDSC.2020.2998135

Authenticated Data Redaction With Accountability and Transparency

Jinhua Ma, Xinyi Huang, *Member, IEEE*,
Yi Mu, *Senior Member, IEEE*, and Robert H. Deng, *Fellow, IEEE*

Abstract—A common practice in data redaction is removing sensitive information prior to data publication or release. In data-driven applications, one must be convinced that the redacted data is still trustworthy. Meanwhile, the data redactor must be held accountable for (malicious) redaction, which could change/hide the meaning of the original data. Motivated by these concerns, we present a novel solution for authenticated data redaction based on a new Redactable Signature Scheme with Implicit Accountability (RSS-IA). In the event of a dispute, not only the original data signer but also the redactor can generate an evidence tag to unequivocally identify the party who produced the data/signature pair. Without the evidence tag, the redaction operation is transparent. Furthermore, the redactor can independently prove the trustworthiness of the redacted data, without any interaction with the original data signer. Our design is built on a new approach which adds accountability to any transparent redactable signature schemes. We show that the proposed design satisfies all the security goals with affordable cost. As an extension, we show how to realize accountable, transparent and authenticated data redaction in the multi-redactor setting.

Index Terms—Data redaction, authenticity, transparency, accountability, redactable signature

1 Introduction

With the rapid development of information technologies, such as cloud computing, big data, Internet of Things, and blockchain, the exponential growth of the global data accelerates the process of enterprise innovation and social change on a global scale. Together with physical assets and human capitals, data have become important assets of enterprises and core strategic resources of countries.

The importance of data drives the development and transformation of various data-driven applications. In the meanwhile, data security has become critical and ensuring data authenticity has become essential during data processing and handling. As useful tools, digital signatures can effectively protect data authenticity and integrity. Traditionally, we require a signature to be existentially unforgeable against adaptive chosen-message attacks (EUF-CMA) [1], which ensures that no probabilistic polynomial-time (PPT) adversary can generate a valid signature for a new data without the private signing key. A valid signature with EUF-CMA security convinces the recipient that the received data has not been tampered with.

An EUF-CMA signature scheme provides strong guarantees of data authenticity and integrity, i.e., the signed data cannot be modified. However, there are also scenarios where data must be modified. To protect privacy, we can remove sensitive personal information, e.g., name and identification number, from the original data. Differential privacy achieves a higher level of privacy protection by adding carefully selected false data to the original one [2]. When data modification is a necessary, it would be desirable that the data modifier can prove the authenticity of legitimately modified data, without the help of the original data issuer. This process is called authenticated data modification. It is clear that authenticated data modification is a challenging issue: Any slight modification would lead the validity of modified data unverifiable, if the original data is protected by EUF-CMA signature schemes.

Digital signatures supporting reasonable data modification have become an active field of security research. This paper focuses on the "delete operation", i.e., removing sensitive information from the authenticated data. Redactable signatures allow a redactor to delete some portions of the signed data, and generate a valid signature for the remaining data without any help from the signer. The concept of redactable signature was respectively presented by Johnson *et al.* [3] and Steinfeld *et al.* [4]. Redactable signatures support authenticated deletions and preserve the origin/integrity verifiability of the redacted data, and hence serve as a remedy to the confliction between authenticated data redaction and traditional signatures with EUF-CMA.

Related Work. Featured with the functionality of redaction, redactable signatures have shown a wide applicability for authenticated data redaction. However, most of the existing redactable signature schemes (RSSs) suffer from the so called malicious redaction problem, where anyone can remove a portion of the signed data and generate a valid signature for the remaining data with public information.

- *J. Ma, X. Huang, and Y. Mu are with the Fujian Provincial Key Laboratory of Network Security and Cryptology, School of Mathematics and Informatics, Fujian Normal University, Fuzhou, Fujian 350000, China. E-mail: jinhuama55@hotmail.com, xyhuang81@yahoo.com, ymu.ieee@gmail.com.*
- *R. H. Deng is with the School of Information Systems, Singapore Management University, Singapore 188065. E-mail: robertdeng@smu.edu.sg.*

Dishonest redactors may abuse redaction, delete some data portions, and deliberately change the original information of the signed data.

There are two approaches in the literature to thwart malicious redaction. The first approach is specifying a redaction policy, i.e., the signer predefines a redaction policy for the signed data. Redaction control is ensured because one can only generate valid signatures for the data conforming with the redaction policy. At present, some RSSs provide coarse-grained redaction control [5], [6], [7], [8], [9], and others provide fine-grained redaction control [4], [10], [11], [12].

Another approach to prevent malicious redaction is making the redaction operation accountable and traceable. This would be more preferable in the situations where the redaction policy is unpredictable. For example, an EHR may be used for various purposes, such as clinic diagnosis, scientific research and driver license. It would be extremely difficult for the EHR issuer to precisely define one redaction policy meeting all situations. Although several works [13], [14], [15], [16] have discussed the design of RSSs with accountability, it is until 2015 that Pöhls and Samelin [17] presented the first formal study of RSS with accountability and named it accountable redactable signature (ARS). ARS allows anyone, using an evidence tag, to identify the party responsible for a valid data/signature pair, and the responsible party cannot deny it. The model and security requirements of ARS schemes were also formally defined in [17].

The design in [17] is a generic transformation that adds accountability to any RSSs by using sanitizable signature schemes (SSSs) [18], [19]. The signature σ of original data $\mathcal{M}$ includes two parts: σ_{RSS} and σ_{SSS}. Specifically, σ_{RSS} is the signature of $\mathcal{M}$, which is generated by invoking the signing algorithm $\mathsf{RSS.Sign}$. σ_{SSS} is the signature of $(\mathcal{M}, \sigma_{\mathsf{RSS}})$, which is generated by invoking the signing algorithm $\mathsf{SSS.Sign}$. The designated redactor can update σ_{RSS} to σ'_{RSS} by invoking the redacting algorithm $\mathsf{RSS.Redact}$, and update $(\mathcal{M}, \sigma_{\mathsf{SSS}})$ to $(\mathcal{M}', \sigma'_{\mathsf{SSS}})$ by invoking the sanitizing algorithm $\mathsf{SSS.Sanitize}$. The signer can generate an evidence tag π by invoking the proof algorithm $\mathsf{SSS.Proof}$. Using π, anyone can determine the responsible party for the controversial signature by invoking the judging algorithm $\mathsf{SSS.Judge}$. This ARS scheme inherits the accountability of the underlying SSS.

With reference to the accountability definition of the SSSs in [18], [19], the accountability of ARS scheme [17] is divided into three categories: signer-accountability, sanitizer-accountability and public-accountability. Signer-accountability requires that the signer cannot generate an evidence tag to blame the redactor for a data/signature pair not generated by her. Sanitizer-accountability requires that the redactor cannot generate a forged data/signature pair to blame the signer. Public-accountability requires that anyone can determine the generator of a data/signature pair using public information.

Transparency hides the redaction operation, which is a stronger privacy protection. It is mutually exclusive with public-accountability. Public-accountability introduced in [17] is used for the situations (such as EHR) when the transparency is not required. This paper aims at achieving accountability and transparency in RSSs. Hence, we introduce the notion named implicit-accountability, which ensures that no one is able to determine the generator of a data/signature pair using public information. As a result, implicit-accountability can co-exist with transparency.

Motivation. Traditional EUF-CMA signature schemes do not support authenticated data modification, but data modification is necessary for privacy protection. With RSSs, one can delete some parts of the data and prove the validity of the remaining data. In order to thwart against malicious "delete operation", there is a need to add accountability to the "delete operation". To the best of our knowledge, the design given in [17] is the only RSS with accountability (i.e., the generator of a data/signature pair can be identified with certain evidence tag) and transparency (i.e., data deletion operation is imperceptible without the evidence tag). This paper takes a step further and investigates the following issues.

1) The evidence tag for accountability in [17] can only be generated by the original signer. When disputes occur, if the signer is off-line or unwilling to generate the evidence tag, the accountability will be unattainable. Hence, it would be more desirable if the data redactor is also able to generate an evidence tag. This will significantly enhance the accountability of redactable signatures, in which evidence tags can be generated by multi-party, i.e., not only the signer but also the redactor can generate the evidence tag.
2) New security concerns arise if the signer and redactor can generate the evidence tag independently. Not only the signer but also the redactor may generate an evidence tag to blame others. In an extreme case, the collusion between the signer and the redactor may create two contradicting evidence tags. Therefore, there is a need to formally redefine the security requirements, particularly the accountability.
3) By invoking the algorithms of SSSs, the ARS scheme [17] inherits the accountability of the underlying SSS. However, sanitizable signatures, due to its design goal, is no longer suitable as a building block if the redactor is also able to generate the accountable evidence tag. This calls for the need of new design approaches.

Contributions. Our major contribution is a new approach of redactable signatures for authenticated data redaction. In our design, not only the signer but also the redactor can independently generate an evidence tag for a data/signature pair. This evidence tag is used to identify whether it is the signer or the redactor who generated the data/signature pair when a dispute occurs. Without this evidence tag, no one is able to determine the origin of a data/signature pair, i.e., the redaction is transparent.

1) *New signature notion:* Motivated by the need of authenticated data redaction, we revise the definitions of accountable and transparent redactable signatures in [17] and introduce a new notion named Redactable Signature Schemes with Implicit Accountability (RSS-IA). In the new notion, both the original signer and the redactor are able to generate an evidence tag to identify the accountable entity. The relevant security requirements are formally defined, including unforgeability, privacy, transparency, signer-accountability, redactor-accountability, and collusion-resistance.

2) *New generic design:* As a starting point, in the setting of single designated data redactor, we present a new generic design of adding accountability to any transparent RSSs. The transparency and accountability of redactor operations can be reduced to the Decisional Diffie-Hellman (DDH) problem, a carefully designed OR-proof, and other primitives. We prove that the new design is secure and satisfies all aforementioned security requirements.
3) *Extension to multi-redactor setting:* We then show how to improve the design by taking into account of multi-redactor. At the signing stage, the signer can choose a group of entities. Anyone in the group is able to redact the data, generate a valid signature for the redacted data and issue an accountable evidence. This is achieved by extending the OR-proof into the multi-party setting.
4) *Test and evaluations:* We test the performance of the proposed RSS-IA by computer simulation. The results validate that the presented designs are practical in achieving accountable and transparent authenticated data redaction.

Organization of This Paper. The rest of this paper is organized as follows. Section 2 reviews the main cryptographic primitives used in this paper. Section 3 provides the formal model and security definition of RSS-IA. Section 4 presents the first generic design of RSS-IA with a single designated redactor, and proves its security. Section 5 proposes the extended RSS-IA with multiple designated redactors, including its generic design and security analysis. Section 6 presents the applications and experiments of our schemes. Finally, we summarize this paper in Section 7.

2 PRELIMINARIES

This section reviews the main cryptographic primitives used in this paper.

2.1 Hardness Problems

Let $\mathbb{G}$ be an Abelian group of prime order $p = |\mathbb{G}|$, and g be a generator of $\mathbb{G}$. Here we describe three hardness problems to $\mathbb{G}$.

Definition 1 (Discrete Logarithm (DL) problem). *For uniform $x \in \mathbb{Z}_p^*$, given $(\mathbb{G}, p, g, g^x)$, compute x.*

Let $\mathcal{A}_{\mathsf{DL}}$ be any PPT attacker who attempts to solve the DL problem, i.e., computes x given $(\mathbb{G}, p, g, g^x)$. We denote the event that $\mathcal{A}_{\mathsf{DL}}$ successfully solves the DL problem (i.e., $\mathcal{A}_{\mathsf{DL}}(\mathbb{G}, p, g, g^x) = x$) by Event 1. We say that the DL problem is hard if the probability of Event 1 is a negligible function negl of the security parameter λ, i.e., $\Pr[\text{Event 1}] \leq \mathsf{negl}(\lambda)$.

Definition 2 (Computational Diffie-Hellman (CDH) problem). *For uniform $x_1, x_2 \in \mathbb{Z}_p^*$, given $(\mathbb{G}, p, g, g^{x_1}, g^{x_2})$, compute $g^{x_1 \cdot x_2}$.*

Similarly, let $\mathcal{A}_{\mathsf{CDH}}$ be any PPT attacker who attempts to solve the CDH problem, i.e., computes $g^{x_1 \cdot x_2}$ given $(\mathbb{G}, p, g, g^{x_1}, g^{x_2})$. The event that $\mathcal{A}_{\mathsf{CDH}}$ successfully solves the CDH problem (i.e., $\mathcal{A}_{\mathsf{CDH}}(\mathbb{G}, p, g, g^{x_1}, g^{x_2}) = g^{x_1 \cdot x_2}$) is denoted by Event 2. We say that the CDH problem is hard if the probability of Event 2 is a negligible function negl of λ, i.e., $\Pr[\text{Event 2}] \leq \mathsf{negl}(\lambda)$.

Definition 3 (Decisional Diffie-Hellman (DDH) problem). *For uniform $x_1, x_2, x_3 \in \mathbb{Z}_p^*$, given $(\mathbb{G}, p, g, g^{x_1}, g^{x_2}, g^{x_3})$, distinguish $g^{x_3} \stackrel{?}{=} g^{x_1 \cdot x_2}$.*

Generally speaking, the DDH problem is to distinguish a CDH problem from a uniform group element. Let $\mathcal{A}_{\mathsf{DDH}}$ be any PPT attacker who attempts to solve the DDH problem, i.e., given uniform g^{x_1}, g^{x_2}, and a third group element g^{x_3}, decides whether $g^{x_3} = g^{x_1 \cdot x_2}$ or whether g^{x_3} was chosen uniformly from $\mathbb{G}$. We denote the event that $\mathcal{A}_{\mathsf{DDH}}$ successfully decides whether $g^{x_3} = g^{x_1 \cdot x_2}$ (i.e., $\mathcal{A}_{\mathsf{DDH}}(\mathbb{G}, p, g, g^{x_1}, g^{x_2}, g^{x_1 \cdot x_2}) = 1$) by Event 3, and its probability is $\Pr[\text{Event 3}]$. Meanwhile, we denote the event that $\mathcal{A}_{\mathsf{DDH}}$ successfully decides whether g^{x_3} was chosen uniformly from $\mathbb{G}$ (i.e., $\mathcal{A}_{\mathsf{DDH}}(\mathbb{G}, p, g, g^{x_1}, g^{x_2}, g^{x_3}) = 1$) by Event 4, and its probability is $\Pr[\text{Event 4}]$. We say that the DDH problem is hard if the advantage probability of $\mathcal{A}_{\mathsf{DDH}}$ solves the DDH problem is a negligible function negl of λ, i.e., $|\Pr[\text{Event 4}] - \Pr[\text{Event 3}]| \leq \mathsf{negl}(\lambda)$.

More details about the above hardness problems can be found in [20].

2.2 Redactable Signatures

An ordinary RSS consists of four polynomial-time algorithms (RSS.KGen, RSS.Sign, RSS.Redact, RSS.Verify) [21]. The probabilistic algorithm RSS.KGen takes as input a security parameter λ, and outputs a public/secret key pair $(\mathsf{pk}_{\mathsf{RSS}}, \mathsf{sk}_{\mathsf{RSS}})$ for the signer, written as $(\mathsf{pk}_{\mathsf{RSS}}, \mathsf{sk}_{\mathsf{RSS}}) \leftarrow \mathsf{RSS}.KGen(1^\lambda)$. The algorithm RSS.Sign takes as input a data $\mathcal{M}$ and $\mathsf{sk}_{\mathsf{RSS}}$, and returns a data/signature pair $(\mathcal{M}, \sigma)$, written as $(\mathcal{M}, \sigma) \leftarrow \mathsf{RSS.Sign}(\mathsf{sk}_{\mathsf{RSS}}, \mathcal{M})$. The algorithm RSS.Redact takes as input $(\mathsf{pk}_{\mathsf{RSS}}, \mathcal{M}, \sigma)$ and a redaction subset $\mathcal{X} \subseteq \mathcal{M}$, and returns a redacted data/signature pair $(\mathcal{M}', \sigma')$, written as $(\mathcal{M}', \sigma') \leftarrow \mathsf{RSS.Redact}(\mathsf{pk}_{\mathsf{RSS}}, \mathcal{M}, \sigma, \mathcal{X})$, where $\mathcal{M}' \leftarrow \mathcal{M} \setminus \mathcal{X}$. The algorithm RSS.Verify takes as input $(\mathsf{pk}_{\mathsf{RSS}}, \mathcal{M}, \sigma)$, and outputs a decision $b \in \{1, 0\}$, written as $b \leftarrow \mathsf{RSS.Verify}(\mathsf{pk}_{\mathsf{RSS}}, \mathcal{M}, \sigma)$, where $b = 1$ means $(\mathcal{M}, \sigma)$ is valid; otherwise, it is invalid.

We require the usual correctness properties of RSSs to hold. Secure RSSs should satisfy unforgeability and privacy. In some scenarios, it should also provide transparency and accountability. Formal security definitions about unforgeability, privacy and transparency can be found in [21], and the definition about accountability is given in this paper.

3 DEFINITIONS OF RSS-IA

In this section, we first present the syntax definition of Redactable Signature Schemes with Implicit Accountability (RSS-IA). Subsequently, we formally define the security properties that RSS-IA should possess.

3.1 Syntax Definition of RSS-IA

In RSS-IA, an evidence tag is necessary to disclose who is the generator of a data/signature pair. In our design, this evidence tag can be generated not only by the original signer but also by the redactor, independently. This is the major difference from the definition in [17], where only the

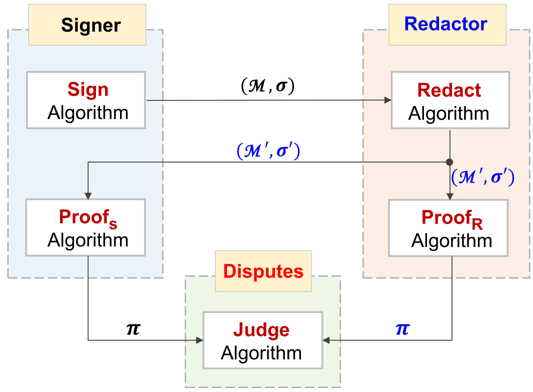

Fig. 1. The operational flow chart of our RSS-IA.

original signer is able to produce the evidence tag. Furthermore, as in [17], the redactor is also designated by the original signer in our RSS-IA. The operational flow chart of our RSS-IA is shown in Fig. 1.

Definition 4. *An* RSS-IA *consists of seven algorithms* $(\mathsf{KGen}, \mathsf{Sign}, \mathsf{Redact}, \mathsf{Verify}, \mathsf{Proof_S}, \mathsf{Proof_R}, \mathsf{Judge})$*:*

$\mathsf{KGen}(1^\lambda)$*: This probabilistic algorithm takes as input a security parameter* λ*. It outputs a key pair* $(\mathsf{PK_S}, \mathsf{SK_S})$ *for the signer and a key pair* $(\mathsf{PK_R}, \mathsf{SK_R})$ *for the redactor:*

$$\{(\mathsf{PK_S}, \mathsf{SK_S}), (\mathsf{PK_R}, \mathsf{SK_R})\} \leftarrow \mathsf{KGen}(1^\lambda).$$

$\mathsf{Sign}(\mathsf{SK_S}, \mathsf{PK_R}, \mathcal{M})$*: This algorithm takes as input the secret key* $\mathsf{SK_S}$*, the public key* $\mathsf{PK_R}$ *and a data* $\mathcal{M}$*. It outputs a data/signature pair* $(\mathcal{M}, \sigma)$*:*

$$(\mathcal{M}, \sigma) \leftarrow \mathsf{Sign}(\mathsf{SK_S}, \mathsf{PK_R}, \mathcal{M}).$$

$\mathsf{Redact}(\mathsf{SK_R}, \mathsf{PK_S}, \mathcal{M}, \sigma, \mathcal{X})$*: This algorithm takes as input the secret key* $\mathsf{SK_R}$*, the public key* $\mathsf{PK_S}$*, a valid data/signature pair* $(\mathcal{M}, \sigma)$ *and a redaction subset* $\mathcal{X} \subseteq \mathcal{M}$*. It outputs a redacted data/signature pair* $(\mathcal{M}', \sigma')$*, where* $\mathcal{M}' = \mathcal{M} \backslash \mathcal{X}$*:*

$$(\mathcal{M}', \sigma') \leftarrow \mathsf{Redact}(\mathsf{SK_R}, \mathsf{PK_S}, \mathcal{M}, \sigma, \mathcal{X}).$$

$\mathsf{Verify}(\mathsf{PK_S}, \mathsf{PK_R}, \mathcal{M}, \sigma)$*: This algorithm takes as input the public keys* $\{\mathsf{PK_S}, \mathsf{PK_R}\}$ *and a data/signature pair* $(\mathcal{M}, \sigma)$*. It outputs a decision* $b \in \{1, 0\}$*, with* $b = 1$ *meaning* $(\mathcal{M}, \sigma)$ *is* valid *and* $b = 0$ *meaning* invalid*:*

$$b \leftarrow \mathsf{Verify}(\mathsf{PK_S}, \mathsf{PK_R}, \mathcal{M}, \sigma).$$

$\mathsf{Proof_S}(\mathsf{SK_S}, \mathsf{PK_R}, \mathcal{M}, \sigma)$*: This algorithm, run by the original signer, takes as input the secret key* $\mathsf{SK_S}$*, the public key* $\mathsf{PK_R}$ *and a valid data/signature pair* $(\mathcal{M}, \sigma)$*. It outputs an evidence tag* π*:*

$$\pi \leftarrow \mathsf{Proof_S}(\mathsf{SK_S}, \mathsf{PK_R}, \mathcal{M}, \sigma).$$

$\mathsf{Proof_R}(\mathsf{SK_R}, \mathsf{PK_S}, \mathcal{M}, \sigma)$*: This algorithm, run by the redactor, takes as input the secret key* $\mathsf{SK_R}$*, the public key* $\mathsf{PK_S}$ *and a valid data/signature pair* $(\mathcal{M}, \sigma)$*. It outputs an evidence tag* π*:*

$$\pi \leftarrow \mathsf{Proof_R}(\mathsf{SK_R}, \mathsf{PK_S}, \mathcal{M}, \sigma).$$

$\mathsf{Judge}(\mathsf{PK_S}, \mathsf{PK_R}, \mathcal{M}, \sigma, \pi)$*: This algorithm takes as input the public keys* $\{\mathsf{PK_S}, \mathsf{PK_R}\}$*, a valid data/signature pair* $(\mathcal{M}, \sigma)$*, and an evidence tag* π*. It outputs a decision* $d \in \{\mathsf{Signer}, \mathsf{Redactor}, \perp\}$*.* $d = \mathsf{Signer}$ *means* $(\mathcal{M}, \sigma)$ *is generated by the signer,* $d = \mathsf{Redactor}$ *means* $(\mathcal{M}, \sigma)$ *is generated by the redactor, and* $d = \perp$ *means* π *is a invalid evidence tag for* $(\mathcal{M}, \sigma)$*:*

$$d \leftarrow \mathsf{Judge}(\mathsf{PK_S}, \mathsf{PK_R}, \mathcal{M}, \sigma, \pi).$$

The correctness of RSS-IA requires that: (a) if $\{(\mathsf{PK_S}, \mathsf{SK_S}), (\mathsf{PK_R}, \mathsf{SK_R})\}$ is correctly generated by KGen and $(\mathcal{M}, \sigma)$ is correctly generated by Sign, $\mathsf{Verify}(\mathsf{PK_S}, \mathsf{PK_R}, \mathcal{M}, \sigma)$ must return 1; and (b) if $\{(\mathsf{PK_S}, \mathsf{SK_S}), (\mathsf{PK_R}, \mathsf{SK_R})\}$ is correctly generated by KGen, $(\mathcal{M}, \sigma)$ is correctly generated by Sign, and $(\mathcal{M}', \sigma')$ is correctly generated by Redact for any redaction subset $\mathcal{X} \subseteq \mathcal{M}$, $\mathsf{Verify}(\mathsf{PK_S}, \mathsf{PK_R}, \mathcal{M}', \sigma')$ must return 1.

3.2 Security Definitions of RSS-IA

A secure accountable RSS should satisfy unforgeability, privacy, transparency, and accountability properties.

- Unforgeability: It ensures that the redactor cannot generate a valid signature for any new data. The redactor can only delete some portions of the signed data in an authentic way.
- Privacy: It ensures that the signature of the redacted data reveals no information on the removed parts.
- Transparency: It hides the redaction operation, which is a stronger privacy protection.
- Accountability: It ensures that the generator of a data/signature pair can be identified by the evidence tag.

The evidence tag for accountability in [17] can only be generated by the original signer, which is a kind of weak accountability. In our RSS-IA, not only the original signer but also the data redactor can generate the evidence tag of accountable entity. This brings new security concerns to RSS-IA. Specifically, the signer or the redactor may generate an evidence tag to blame others. In an extreme case, the collusion between the signer and the redactor may create two contradicting evidence tags. Therefore, the accountability should be further divided into three types:

- Signer-Accountability: It ensures that the signer cannot blame a redactor for a data/signature pair not generated by her.
- Redactor-Accountability: It ensures that the redactor cannot blame the signer for a data/signature pair not generated by her.
- Collusion-Resistance: It ensures that even the signer and the redactor collude with each other by sharing all secret information, they cannot both claim (or deny) themselves as the generator of a valid data/signature pair.

On the basis of the security definitions in [17], in the following sections we define these security goals of RSS-IA. Specifically, the unforgeability model of RSS-IA is similar with the sanitizer-unforgeability model in [17]. The attacker in our privacy and transparency models is able to query the oracle $\mathsf{Proof_R}$, which is the major differences from [17].

3.2.1 Unforgeability of RSS-IA

The unforgeability of RSS-IA is formalized that without access to the signer's secret key $\mathsf{SK_S}$, even a dishonest PPT redactor $\mathcal{A}$ cannot generate a valid signature for a new data.

Definition 5 (**Unforgeability**). *An* RSS-IA *is unforgeable, if for any PPT adversary $\mathcal{A}$, the success probability of $\mathcal{A}$ in winning* **Game 1**, *i.e., $Succ_{\mathcal{A}}^{\mathsf{UNF}}(\lambda) = \Pr[\textbf{Game 1} = 1]$, is a negligible function of the security parameter λ.*

Game 1: Unforgeability$_{\mathcal{A}}^{\mathsf{RSS-IA}}(\lambda)$
$\{(\mathsf{PK_S}, \mathsf{SK_S}), (\mathsf{PK_R}, \mathsf{SK_R})\} \leftarrow \mathsf{KGen}(1^\lambda)$
$\mathcal{Q} \leftarrow \emptyset$
$(\mathcal{M}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{SK_S},\cdot,\cdot)}_{\mathsf{Proof_S}(\mathsf{SK_S},\cdot,\cdot,\cdot)}(1^\lambda, \mathsf{PK_S}, \mathsf{PK_R}, \mathsf{SK_R})$
for $i = 1, 2, \ldots, q$, let $\mathcal{M}_i$ be the ith query to Sign oracle, $\mathcal{Q} \leftarrow \mathcal{Q} \cup \mathsf{span}(\mathcal{M}_i)$, and $\mathsf{span}(\mathcal{M}_i)$ denote the power set of $\mathcal{M}_i$
return 1, if $\mathcal{M}^* \notin \mathcal{Q} \wedge \mathsf{Verify}(\mathsf{PK_S}, \mathsf{PK_R}, \mathcal{M}^*, \sigma^*) = 1$
else, return 0

3.2.2 Privacy of RSS-IA

The privacy of RSS-IA is formalized that without access to the secret keys $\mathsf{SK_S}$ and $\mathsf{SK_R}$, even if a PPT attacker $\mathcal{A}$ can query all oracles adaptively and choose two candidate data/redaction-subset pairs, it should be infeasible to tell from which one the redacted data stems.

Definition 6 (**Privacy**). *An* RSS-IA *is private, if for any PPT adversary $\mathcal{A}$, the success advantage of $\mathcal{A}$ in winning* **Game 2**, *i.e., $Adv_{\mathcal{A}}^{\mathsf{Pri}}(\lambda) = |\Pr[\textbf{Game 2} = 1] - \frac{1}{2}|$, is a negligible function of λ.*

Game 2: **Privacy**$_{\mathcal{A}}^{\mathsf{RSS-IA}}(\lambda)$
$\{(\mathsf{PK_S}, \mathsf{SK_S}), (\mathsf{PK_R}, \mathsf{SK_R})\} \leftarrow \mathsf{KGen}(1^\lambda)$
$\mathcal{Q} \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{SK_S},\cdot,\cdot),\mathsf{Redact}(\mathsf{SK_R},\cdot,\cdot,\cdot,\cdot)}_{\mathsf{Proof_S}(\mathsf{SK_S},\cdot,\cdot,\cdot),\mathsf{Proof_R}(\mathsf{SK_R},\cdot,\cdot,\cdot)}(1^\lambda, \mathsf{PK_S}, \mathsf{PK_R})$
(where $\mathcal{Q} = (\mathcal{M}_0, \mathcal{X}_0, \mathcal{M}_1, \mathcal{X}_1)$)
if $\mathcal{M}_0 = \mathcal{M}_1$ or $\mathcal{M}_0 \setminus \mathcal{X}_0 \neq \mathcal{M}_1 \setminus \mathcal{X}_1$, return 0
$b' \leftarrow \mathcal{A}^{\mathsf{LoRRedact}(\mathsf{SK_S},\mathsf{SK_R},b,\cdot)}(1^\lambda, \mathsf{PK_S}, \mathsf{PK_R}, \mathcal{Q})$
oracle LoRRedact does: $b \leftarrow \{0, 1\}$
$(\mathcal{M}_b, \sigma) \leftarrow \mathsf{Sign}(\mathsf{SK_S}, \mathsf{PK_R}, \mathcal{M}_b)$
return $(\mathcal{M}', \sigma') \leftarrow \mathsf{Redact}(\mathsf{SK_R}, \mathsf{PK_S}, \mathcal{M}_b, \sigma, \mathcal{X}_b)$
return 1, if $b' = b$
else, return 0

3.2.3 Transparency of RSS-IA

The transparency of RSS-IA is formalized that without access to the secret keys $\mathsf{SK_S}$ and $\mathsf{SK_R}$, even if a PPT attacker $\mathcal{A}$ can choose the challenging data/redaction-subset pair after querying all oracles adaptively, it should be infeasible to determine the accountable party for an output data/signature pair with public information, i.e., judge whether a signature is generated by the signer or the redactor.

Definition 7: (**Transparency**). *An* RSS-IA *is transparent, if for any PPT adversary $\mathcal{A}$, the success advantage of $\mathcal{A}$ in winning* **Game 3**, *i.e., $Adv_{\mathcal{A}}^{\mathsf{Trans}}(\lambda) = |\Pr[\textbf{Game 3} = 1] - \frac{1}{2}|$, is a negligible function of λ.*

Game 3: **Transparency**$_{\mathcal{A}}^{\mathsf{RSS-IA}}(\lambda)$
$\{(\mathsf{PK_S}, \mathsf{SK_S}), (\mathsf{PK_R}, \mathsf{SK_R})\} \leftarrow \mathsf{KGen}(1^\lambda)$
$(\mathcal{M}, \mathcal{X}) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{SK_S},\cdot,\cdot),\mathsf{Redact}(\mathsf{SK_R},\cdot,\cdot,\cdot,\cdot)}_{\mathsf{Proof_S}(\mathsf{SK_S},\cdot,\cdot,\cdot),\mathsf{Proof_R}(\mathsf{SK_R},\cdot,\cdot,\cdot)}(1^\lambda, \mathsf{PK_S}, \mathsf{PK_R})$
if $\mathcal{X} \not\subset \mathcal{M}$, return 0
$\mathcal{M}' \leftarrow \mathcal{M} \setminus \mathcal{X}$
if $\mathcal{M}'$ has been queried to Sign oracle, return 0
$b' \leftarrow \mathcal{A}^{\mathsf{Redact/Sign}(\mathsf{SK_S},\mathsf{SK_R},b,\cdot,\cdot)}(1^\lambda, \mathsf{PK_S}, \mathsf{PK_R}, \mathcal{M}, \mathcal{X})$
oracle Redact/Sign does: $b \leftarrow \{0, 1\}$
if $b = 0$,
$(\mathcal{M}, \sigma) \leftarrow \mathsf{Sign}(\mathsf{SK_S}, \mathsf{PK_R}, \mathcal{M})$
if $(\mathcal{M}, \sigma, \mathcal{X})$ has been queried to Redact oracle, return 0
else, $(\mathcal{M}', \sigma'_0) \leftarrow \mathsf{Redact}(\mathsf{SK_R}, \mathsf{PK_S}, \mathcal{M}, \sigma, \mathcal{X})$
else, $(\mathcal{M}', \sigma'_1) \leftarrow \mathsf{Sign}(\mathsf{SK_S}, \mathsf{PK_R}, \mathcal{M}')$
if $(\mathcal{M}', \sigma'_b)$ has been queried to $\mathsf{Proof_S}$ or $\mathsf{Proof_R}$ oracle, return 0
else, return $(\mathcal{M}', \sigma'_b)$
return 1, if $b' = b$
else, return 0

3.2.4 Signer-Accountability of RSS-IA

The signer-accountability of RSS-IA is formalized that without access to the secret key $\mathsf{SK_R}$, even if a dishonest PPT signer $\mathcal{A}$ can query all oracles related to the redactor adaptively, it should be infeasible to generate an evidence tag π with which the Judge algorithm outputs Redactor.

Definition 8 (**Signer−Accountability**). *An* RSS-IA *is signer-accountable, if for any PPT signer $\mathcal{A}$, the success probability of $\mathcal{A}$ in winning* **Game 4**, *i.e., $Succ_{\mathcal{A}}^{\mathsf{S-IA}}(\lambda) = \Pr[\textbf{Game 4} = 1]$, is a negligible function of λ.*

Game 4: Signer−Accountability$_{\mathcal{A}}^{\mathsf{RSS-IA}}(\lambda)$
$\{(\mathsf{PK_S}, \mathsf{SK_S}), (\mathsf{PK_R}, \mathsf{SK_R})\} \leftarrow \mathsf{KGen}(1^\lambda)$
$(\mathcal{M}, \sigma, \pi) \leftarrow \mathcal{A}^{\mathsf{Redact}(\mathsf{SK_R},\mathsf{PK_S},\cdot,\cdot,\cdot)}_{\mathsf{Proof_R}(\mathsf{SK_R},\mathsf{PK_S},\cdot,\cdot)}(1^\lambda, \mathsf{PK_S}, \mathsf{SK_S}, \mathsf{PK_R})$
if $\mathcal{M}$ is an output of Redact oracle, return 0
return 1, if $\mathsf{Verify}(\mathsf{PK_S}, \mathsf{PK_R}, \mathcal{M}, \sigma) = 1 \wedge$
$\mathsf{Judge}(\mathsf{PK_S}, \mathsf{PK_R}, \mathcal{M}, \sigma, \pi) = \mathsf{Redactor}$
else, return 0

3.2.5 Redactor-Accountability of RSS-IA

The redactor-accountability of RSS-IA is formalized that without access to the secret key $\mathsf{SK_S}$, even if a dishonest PPT redactor $\mathcal{A}$ can query all oracles related to the signer adaptively, it should be infeasible to generate an evidence tag π with which the Judge algorithm outputs Signer.

Definition 9 (**Redactor−Accountability**). *An* RSS-IA *is redactor-accountable, if for any PPT redactor $\mathcal{A}$, the success probability of $\mathcal{A}$ in winning* **Game 5**, *i.e., $Succ_{\mathcal{A}}^{\mathsf{R-IA}}(\lambda) = \Pr[\textbf{Game 5} = 1]$, is a negligible function of λ.*

Game 5: Redactor−Accountability$_{\mathcal{A}}^{\mathsf{RSS-IA}}(\lambda)$
$\{(\mathsf{PK_S}, \mathsf{SK_S}), (\mathsf{PK_R}, \mathsf{SK_R})\} \leftarrow \mathsf{KGen}(1^\lambda)$
$(\mathcal{M}, \sigma, \pi) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{SK_S},\mathsf{PK_R},\cdot)}_{\mathsf{Proof_S}(\mathsf{SK_S},\mathsf{PK_R},\cdot,\cdot)}(1^\lambda, \mathsf{PK_S}, \mathsf{PK_R}, \mathsf{SK_R})$
if $\mathcal{M}$ has been queried to Sign oracle, return 0
else if $\mathsf{Verify}(\mathsf{PK_S}, \mathsf{PK_R}, \mathcal{M}, \sigma) = 1 \wedge$
$\mathsf{Judge}(\mathsf{PK_S}, \mathsf{PK_R}, \mathcal{M}, \sigma, \pi) = \mathsf{Signer}$
return 1
otherwise, return 0

3.2.6 Collusion-Resistance of RSS-IA

The collusion-resistance of RSS-IA is formalized that even if with the secret keys $\mathsf{SK_S}$ and $\mathsf{SK_R}$, it should be infeasible for a PPT attacker $\mathcal{A}$ to generate two evidence tags for a data/signature pair, such that the Judge algorithm outputs Signer on input one evidence tag, and outputs Redactor on input another evidence tag.

Definition 10 (**Collusion−Resistance**). *An* RSS-IA *satisfies collusion-resistance, if for any PPT adversary $\mathcal{A}$, the success probability of $\mathcal{A}$ in winning* **Game 8**, *i.e.,* $Succ_{\mathcal{A}}^{\mathsf{CR-IA}}(\lambda) = \Pr[\textbf{Game 8} = 1]$, *is a negligible function of* λ.

Game 8: Collusion−Resistance$_{\mathcal{A}}^{\mathsf{RSS-IA}}(\lambda)$
$\{(\mathsf{PK_S}, \mathsf{SK_S}), (\mathsf{PK_R}, \mathsf{SK_R})\} \leftarrow \mathsf{KGen}(1^\lambda)$
$(\mathcal{M}, \sigma, \pi, \pi') \leftarrow \mathcal{A}(1^\lambda, \mathsf{PK_S}, \mathsf{PK_R}, \mathsf{SK_S}, \mathsf{SK_R})$
return 1, if
$\mathsf{Verify}(\mathsf{PK_S}, \mathsf{PK_R}, \mathcal{M}, \sigma) = 1 \wedge$
$\mathsf{Judge}(\mathsf{PK_S}, \mathsf{PK_R}, \mathcal{M}, \sigma, \pi) = \mathsf{Signer} \wedge$
$\mathsf{Judge}(\mathsf{PK_S}, \mathsf{PK_R}, \mathcal{M}, \sigma, \pi') = \mathsf{Redactor}$
else, return 0

4 Generic RSS-IA Construction

In this section, we first provide a generic RSS-IA construction with a single designated redactor. Then, we prove its security.

4.1 Generic RSS-IA

Our design is a generic transformation that adds implicit accountability to any transparent RSSs. Let (RSS.KGen, RSS.Sign, RSS.Redact, RSS.Verify) be the relevant algorithms in a transparent RSS without accountability. Our design also needs two cryptographic hash functions: $H_1 : \{0,1\}^* \rightarrow \mathbb{G}$ and $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_p$. $\mathbb{G}$ is an Abelian group of prime order p and generator g. Denote the identities of the signer and the redactor by Signer and Redactor, respectively.

Our RSS-IA consists of seven algorithms: KGen, Sign, Redact, Verify, Proof$_\mathsf{S}$, Proof$_\mathsf{R}$ and Judge.

$\mathsf{KGen}(1^\lambda)$. Given a security parameter λ, this algorithm generates the key pairs of the signer and the redactor.

1) To generate the key pair $(\mathsf{PK_S}, \mathsf{SK_S})$ of the signer:
   a) Runs the key generation algorithm $\mathsf{RSS.KGen}(1^\lambda)$, and gets a key pair $(\mathsf{pk_{RSS}}, \mathsf{sk_{RSS}})$.
   b) Chooses a random number $\mathsf{x_S} \xleftarrow{R} \mathbb{Z}_p^*$ uniformly and randomly, and calculates $\mathsf{y_S} = g^{\mathsf{x_S}}$.
   c) The secret key $\mathsf{SK_S}$ of the signer is $(\mathsf{sk_{RSS}}, \mathsf{x_S})$, and public key $\mathsf{PK_S}$ is $(\mathsf{pk_{RSS}}, \mathsf{y_S})$.
2) To generate the key pair $(\mathsf{PK_R}, \mathsf{SK_R})$ of the redactor, the redactor chooses a random number $\mathsf{x_R} \xleftarrow{R} \mathbb{Z}_p^*$ as her secret key $\mathsf{SK_R}$, and calculates $\mathsf{y_R} = g^{\mathsf{x_R}}$ as her public key $\mathsf{PK_R}$.

$\mathsf{Sign}(\mathsf{SK_S}, \mathsf{PK_R}, \mathcal{M})$. The signature σ of a data $\mathcal{M}$ consists of three components: $(\sigma_1, \sigma_2, \sigma_3)$.

1) σ_1 is the output of the RSS.Sign: The signer runs the signing algorithm $\mathsf{RSS.Sign}(\mathsf{sk_{RSS}}, \mathcal{M})$, and gets the redactable signature σ_1 for the original data $\mathcal{M}$.
2) σ_2 is an auxiliary tag for accountability:
   a) Calculates $h = H_1(\mathcal{M}, \sigma_1)$.
   b) Calculates $\sigma_2 = h^{\mathsf{x_S}}$.
3) σ_3 is a proof of the statement $\sigma_2 = h^{\mathsf{x_S}}$ or $\sigma_2 = h^{\mathsf{x_R}}$:
   a) Chooses $\{\nu, \beta_3, \beta_4\} \xleftarrow{R} \mathbb{Z}_p^*$, and calculates $\beta_2 = H_2(g^\nu, h^\nu, g^{\beta_3}{\mathsf{y_R}}^{\beta_4}, h^{\beta_3}{\sigma_2}^{\beta_4}) - \beta_4 \bmod p$, and $\beta_1 = \nu - \beta_2 \mathsf{x_S} \bmod p$.
   b) Denotes $\sigma_3 = \{\beta_1, \beta_2, \beta_3, \beta_4\}$.
4) Returns the signature σ for the data $\mathcal{M}$, where $\sigma = \{\sigma_1, \sigma_2, \sigma_3\}$.

σ_1 enables the redactor to generate a valid signature after redaction, σ_2 ensures the accountability, and σ_3 is an Or-Proof (which serves for the transparency). The Or-Proof is motivated by the techniques in [22].

$\mathsf{Redact}(\mathsf{SK_R}, \mathsf{PK_S}, \mathcal{M}, \sigma, \mathcal{X})$. $(\mathcal{M}, \sigma)$ is a valid data/signature pair, where $\sigma = \{\sigma_1, \sigma_2, \sigma_3\}$ and $\sigma_3 = \{\beta_1, \beta_2, \beta_3, \beta_4\}$. The signature σ' of a redacted data $\mathcal{M}'$ consists of three components: $(\sigma_1', \sigma_2', \sigma_3')$. $\mathcal{X} \subseteq \mathcal{M}$ is a redaction subset.

1) σ_1' is the output of the RSS.Redact: The redactor runs the redacting algorithm $\mathsf{RSS.Redact}(\mathsf{pk_{RSS}}, \mathcal{M}, \sigma_1, \mathcal{X})$, and gets the redacted signature σ_1' for the redacted data $\mathcal{M}' \leftarrow \mathcal{M} \setminus \mathcal{X}$.
2) σ_2' is an auxiliary tag for accountability:
   a) Calculates $h = H_1(\mathcal{M}', \sigma_1')$.
   b) Calculates $\sigma_2' = h^{\mathsf{x_R}}$.
3) σ_3' is a proof of the statement $\sigma_2' = h^{\mathsf{x_S}}$ or $\sigma_2' = h^{\mathsf{x_R}}$:
   a) Chooses $\{\nu', \beta_1', \beta_2'\} \xleftarrow{R} \mathbb{Z}_p^*$, and calculate $\beta_4' = H_2(g^{\beta_1'}{\mathsf{y_S}}^{\beta_2'}, h^{\beta_1'}{\sigma_2'}^{\beta_2'}, g^{\nu'}, h^{\nu'}) - \beta_2' \bmod p$, and $\beta_3' = \nu' - \beta_4' \mathsf{x_R} \bmod p$.
   b) Denotes $\sigma_3' = \{\beta_1', \beta_2', \beta_3', \beta_4'\}$.
4) Returns the signature σ' for the data $\mathcal{M}'$, where $\sigma' = \{\sigma_1', \sigma_2', \sigma_3'\}$.

$\mathsf{Verify}(\mathsf{PK_S}, \mathsf{PK_R}, \mathcal{M}, \sigma)$: Given a data/signature pair $(\mathcal{M}, \sigma)$, where $\sigma = \{\sigma_1, \sigma_2, \sigma_3\}$ and $\sigma_3 = \{\beta_1, \beta_2, \beta_3, \beta_4\}$, the verifier does the following.

1) Verifying the validity of σ_1: The verifier runs the verify algorithm RSS.Verify, and gets a decision $b \leftarrow \mathsf{RSS.Verify}(\mathsf{pk_{RSS}}, \mathcal{M}, \sigma_1)$. If $b = 0$, return 0.
2) Verifying the validity of σ_2: The verifier
   a) Calculates $h = H_1(\mathcal{M}, \sigma_1)$.
   b) If $\beta_2 + \beta_4 = H_2(g^{\beta_1}{\mathsf{y_S}}^{\beta_2}, h^{\beta_1}{\sigma_2}^{\beta_2}, g^{\beta_3}{\mathsf{y_R}}^{\beta_4}, h^{\beta_3}{\sigma_2}^{\beta_4})$, it indicates σ_2 is valid, returns 1.
3) Otherwise, returns 0.

$\mathsf{Proof_S}(\mathsf{SK_S}, \mathsf{PK_R}, \mathcal{M}, \sigma)$. $(\mathcal{M}, \sigma)$ is a valid data/signature pair, where $\sigma = \{\sigma_1, \sigma_2, \sigma_3\}$ and $\sigma_3 = \{\beta_1, \beta_2, \beta_3, \beta_4\}$. To generate an evidence tag π to reveal the generator of $(\mathcal{M}, \sigma)$, the signer:

1) Calculates $h = H_1(\mathcal{M}, \sigma_1)$.
2) If $\sigma_2 = h^{\mathsf{x_S}}$, the signer proves "$\mathsf{x_S} = \log_g \mathsf{y_S} = \log_h \sigma_2$":
   a) Chooses $\omega \xleftarrow{R} \mathbb{Z}_p^*$.
   b) Calculates $\pi_1 = H_2(g^\omega, h^\omega)$ and $\pi_2 = \omega - \pi_1 \mathsf{x_S} \bmod p$.
   c) Returns the evidence tag $\pi = \{\pi_1, \pi_2\}$.
3) Otherwise, i.e., $\sigma_2 \neq h^{\mathsf{x_S}}$, the signer proves "$\mathsf{x_S} = \log_g \mathsf{y_S} \neq \log_h \sigma_2$":
   a) Chooses $\omega_1, \omega_2, \omega_3 \xleftarrow{R} \mathbb{Z}_p^*$.
   b) Calculates $\pi_1 = (h^{\mathsf{x_S}}/\sigma_2)^{\omega_1}$, $\pi_2 = H_2(\pi_1, g^{\omega_2}/{\mathsf{y_S}}^{\omega_3}, h^{\omega_2}/{\sigma_2}^{\omega_3})$, $\pi_3 = \omega_3 + \omega_1 \pi_2 \bmod p$ and $\pi_4 = \omega_2 + \omega_1 \pi_2 \mathsf{x_S} \bmod p$.
   c) Returns the evidence tag $\pi = \{\pi_1, \pi_2, \pi_3, \pi_4\}$.

As we can see, σ_2 is an undeniable signature. The generation of the evidence tag π is based on the non-interactive zero-knowledge proof for Chaum's scheme [23], which ensures that one is not able to prove "$\sigma_2 = h^{\mathsf{x_S}}$" if "$\sigma_2 = h^{\mathsf{x_R}}$" (or, "$\sigma_2 = h^{\mathsf{x_R}}$" if "$\sigma_2 = h^{\mathsf{x_S}}$").

$\mathsf{Proof_R}(\mathsf{SK_R}, \mathsf{PK_S}, \mathcal{M}, \sigma)$. $(\mathcal{M}, \sigma)$ is a valid data/signature pair, where $\sigma = \{\sigma_1, \sigma_2, \sigma_3\}$ and $\sigma_3 = \{\beta_1, \beta_2, \beta_3, \beta_4\}$. To generate an evidence tag π to reveal the generator of $(\mathcal{M}, \sigma)$, the redactor:

1) Calculates $h = H_1(\mathcal{M}, \sigma_1)$.
2) If $\sigma_2 = h^{\mathsf{x_R}}$, the redactor proves "$\mathsf{x_R} = \log_g \mathsf{y_R} = \log_h \sigma_2$":
   a) Chooses $\omega \xleftarrow{R} \mathbb{Z}_p^*$.
   b) Calculates $\pi_1 = H_2(g^\omega, h^\omega)$ and $\pi_2 = \omega - \pi_1 \mathsf{x_R} \bmod p$.
   c) Returns the evidence tag $\pi = \{\pi_1, \pi_2\}$.
3) Otherwise, i.e., $\sigma_2 \neq h^{\mathsf{x_R}}$, the redactor proves "$\mathsf{x_R} = \log_g \mathsf{y_R} \neq \log_h \sigma_2$":
   a) Chooses $\omega_1, \omega_2, \omega_3 \xleftarrow{R} \mathbb{Z}_p^*$.
   b) Calculates $\pi_1 = (h^{\mathsf{x_R}}/\sigma_2)^{\omega_1}$, $\pi_2 = H_2(\pi_1, g^{\omega_2}/\mathsf{y_R}^{\omega_3}, h^{\omega_2}/\sigma_2^{\omega_3})$, $\pi_3 = \omega_3 + \omega_1\pi_2 \bmod p$ and $\pi_4 = \omega_2 + \omega_1\pi_2\mathsf{x_R} \bmod p$.
   c) Returns the evidence tag $\pi = \{\pi_1, \pi_2, \pi_3, \pi_4\}$.

With the same principle of $\mathsf{Proof_S}$ algorithm, the evidence tag π generated by $\mathsf{Proof_R}$ algorithm ensures that one is not able to prove "$\sigma_2 = h^{\mathsf{x_R}}$" if "$\sigma_2 = h^{\mathsf{x_S}}$" (or, "$\sigma_2 = h^{\mathsf{x_S}}$" if "$\sigma_2 = h^{\mathsf{x_R}}$").

$\mathsf{Judge}(\mathsf{PK_S}, \mathsf{PK_R}, \mathcal{M}, \sigma, \pi)$. $(\mathcal{M}, \sigma)$ is a valid data/signature pair, where $\sigma = \{\sigma_1, \sigma_2, \sigma_3\}$ and $\sigma_3 = \{\beta_1, \beta_2, \beta_3, \beta_4\}$. π is an evidence tag for $(\mathcal{M}, \sigma)$. To determine the responsible party for $(\mathcal{M}, \sigma)$, the auditor:

1) Calculates $h = H_1(\mathcal{M}, \sigma_1)$.
2) If $\pi = \{\pi_1, \pi_2\} \in \mathbb{Z}_p^* \times \mathbb{Z}_p^*$:
   a) If $\pi_1 = H_2(g^{\pi_2}\mathsf{y_S}^{\pi_1}, h^{\pi_2}\sigma_2^{\pi_1})$, returns Signer.
   b) If $\pi_1 = H_2(g^{\pi_2}\mathsf{y_R}^{\pi_1}, h^{\pi_2}\sigma_2^{\pi_1})$, returns Redactor.
3) Else, if $\pi = \{\pi_1, \pi_2, \pi_3, \pi_4\} \in \mathbb{G} \times \mathbb{Z}_p^* \times \mathbb{Z}_p^* \times \mathbb{Z}_p^*$:
   a) If $\pi_1 \neq 1$ and $\pi_2 = H_2(\pi_1, g^{\pi_4}/\mathsf{y_R}^{\pi_3}, h^{\pi_4}/(\sigma_2^{\pi_3}\pi_1^{\pi_2}))$, returns Signer.
   b) If $\pi_1 \neq 1$ and $\pi_2 = H_2(\pi_1, g^{\pi_4}/\mathsf{y_S}^{\pi_3}, h^{\pi_4}/(\sigma_2^{\pi_3}\pi_1^{\pi_2}))$, returns Redactor.
4) Otherwise, returns $\perp$.

The correctness of the generic RSS-IA can be verified straightforwardly. In our RSS-IA, only the designated redactor who possesses the secret key $\mathsf{x_R}$ can legally redact the originally signed data. Our RSS-IA satisfies all security requirements of RSS-IA.

4.2 Security Analysis of the Generic RSS-IA

In this subsection, we prove the security of the generic RSS-IA with a single designated redactor in terms of unforgeability, privacy, transparency, signer-accountability, redactor-accountability, and collusion-resistance.

4.2.1 Unforgeability of the Generic RSS-IA

Theorem 1. *Our* RSS-IA *satisfies unforgeability, if the underlying* RSS *satisfies unforgeability.*

Proof. Let $\mathcal{A}_{\mathsf{RSS-IA}}$ be a PPT attacker who can win the unforgeability game (defined in Section 3.2.1) of our RSS-IA with the success probability of ϵ_1. If $\mathcal{A}_{\mathsf{RSS-IA}}$ exists, we can construct an attacker $\mathcal{A}_{\mathsf{RSS}}$ who can break the unforgeability property of the underlying RSS with the success probability of ϵ_{RSS}. $\mathcal{A}_{\mathsf{RSS}}$ acts as the challenger of $\mathcal{A}_{\mathsf{RSS-IA}}$ as follows.

Setup. The challenger $\mathcal{C}_{\mathsf{RSS}}$ generates a key pair $(\mathsf{pk_{RSS}}, \mathsf{sk_{RSS}})$ of an RSS and sends $\mathsf{pk_{RSS}}$ to $\mathcal{A}_{\mathsf{RSS}}$. $\mathcal{A}_{\mathsf{RSS}}$ chooses a random number $\mathsf{x_S} \xleftarrow{R} \mathbb{Z}_p^*$, calculates $\mathsf{y_S} = g^{\mathsf{x_S}}$, and chooses two cryptographic hash functions: $H_1 : \{0,1\}^* \to \mathbb{G}$ and $H_2 : \{0,1\}^* \to \mathbb{Z}_p$. Then, $\mathcal{A}_{\mathsf{RSS}}$ sends $(\mathsf{pk_{RSS}}, \mathsf{y_S}, H_1, H_2)$ to $\mathcal{A}_{\mathsf{RSS-IA}}$. $\mathcal{A}_{\mathsf{RSS-IA}}$ generates the key pair $(\mathsf{y_R}, \mathsf{x_R})$ of the redactor of our RSS-IA and sends $\mathsf{y_R}$ to $\mathcal{A}_{\mathsf{RSS}}$.

Queries. $\mathcal{A}_{\mathsf{RSS-IA}}$ can adaptively query oracles $\{\mathsf{Sign}, \mathsf{Proof_S}\}$. Oracle Sign is simulated by $\mathcal{A}_{\mathsf{RSS}}$ querying $\mathcal{C}_{\mathsf{RSS}}$ and using $\mathsf{x_S}$. Oracle $\mathsf{Proof_S}$ is simulated by $\mathcal{A}_{\mathsf{RSS}}$ using $\mathsf{x_S}$. Specifically, $\mathcal{A}_{\mathsf{RSS-IA}}$ passes her ith signature query $\mathcal{M}_i$ to $\mathcal{A}_{\mathsf{RSS}}$. $\mathcal{A}_{\mathsf{RSS}}$ queries $\mathcal{M}_i$ to $\mathcal{C}_{\mathsf{RSS}}$, and obtains the signature response $\sigma_{1,i}$ from $\mathcal{C}_{\mathsf{RSS}}$. $\mathcal{A}_{\mathsf{RSS}}$ calculates $h_i = H_1(\mathcal{M}_i, \sigma_{1,i})$ and $\sigma_{2,i} = {h_i}^{\mathsf{x_S}}$. $\mathcal{A}_{\mathsf{RSS}}$ chooses $\{\nu_i, \beta_{3,i}, \beta_{4,i}\} \xleftarrow{R} \mathbb{Z}_p^*$ and calculates $\beta_{2,i} = H_2(g^{\nu_i}, {h_i}^{\nu_i}, g^{\beta_{3,i}}\mathsf{y_R}^{\beta_{4,i}}, {h_i}^{\beta_{3,i}}{\sigma_{2,i}}^{\beta_{4,i}}) - \beta_{4,i}$, $\beta_{1,i} = \nu_i - \beta_{2,i}\mathsf{x_S}$, and $\sigma_{3,i} = \{\beta_{1,i}, \beta_{2,i}, \beta_{3,i}, \beta_{4,i}\}$. $\mathcal{A}_{\mathsf{RSS}}$ returns $\sigma_i = \{\sigma_{1,i}, \sigma_{2,i}, \sigma_{3,i}\}$ to $\mathcal{A}_{\mathsf{RSS-IA}}$ as her response.

Output. $\mathcal{A}_{\mathsf{RSS-IA}}$ outputs a data/signature pair $(\mathcal{M}^*, \sigma^*)$ as her forgery, where $\sigma^* = ({\sigma_1}^*, {\sigma_2}^*, {\sigma_3}^*)$. Then $\mathcal{A}_{\mathsf{RSS}}$ outputs $(\mathcal{M}^*, {\sigma_1}^*)$ as her forgery.

As defined in Section 3.2.1, if $\mathcal{A}_{\mathsf{RSS-IA}}$ succeeds, then $\mathcal{M}^*$ has not been queried to Sign oracle and is not a subset of any queried data $\mathcal{M}_i$. Meanwhile, σ^* is valid. Hence, $\epsilon_{\mathsf{RSS}} = \epsilon_1$. If ϵ_1 is negligible, ϵ_{RSS} is also negligible. This completes the proof. □

4.2.2 Privacy of the Generic RSS-IA

Theorem 2. *Our* RSS-IA *satisfies privacy, if the underlying* RSS *satisfies privacy.*

Proof. Let $\mathcal{A}_{\mathsf{RSS-IA}}$ be a PPT attacker who can win the privacy game (defined in Section 3.2.2) of our RSS-IA with the success advantage of ϵ_2. If $\mathcal{A}_{\mathsf{RSS-IA}}$ exists, we can construct an attacker $\mathcal{A}_{\mathsf{RSS}}$ who can break the privacy property of the underlying RSS with the success advantage of ϵ_{RSS}. $\mathcal{A}_{\mathsf{RSS}}$ acts as the challenger of $\mathcal{A}_{\mathsf{RSS-IA}}$ as follows.

Setup. The challenger $\mathcal{C}_{\mathsf{RSS}}$ generates a key pair $(\mathsf{pk_{RSS}}, \mathsf{sk_{RSS}})$ of an RSS and sends $\mathsf{pk_{RSS}}$ to $\mathcal{A}_{\mathsf{RSS}}$. $\mathcal{A}_{\mathsf{RSS}}$ chooses two random numbers $\mathsf{x_S}, \mathsf{x_R} \xleftarrow{R} \mathbb{Z}_p^*$, calculates $\mathsf{y_S} = g^{\mathsf{x_S}}, \mathsf{y_R} = g^{\mathsf{x_R}}$, and chooses two cryptographic hash functions: $H_1 : \{0,1\}^* \to \mathbb{G}$ and $H_2 : \{0,1\}^* \to \mathbb{Z}_p$. Then, $\mathcal{A}_{\mathsf{RSS}}$ sends $(\mathsf{pk_{RSS}}, \mathsf{y_S}, \mathsf{y_R}, H_1, H_2)$ to $\mathcal{A}_{\mathsf{RSS-IA}}$.

Queries. $\mathcal{A}_{\mathsf{RSS-IA}}$ can adaptively query oracles $\{\mathsf{Sign}, \mathsf{Proof_S}, \mathsf{Redact}, \mathsf{Proof_R}\}$. Oracle Sign is simulated by $\mathcal{A}_{\mathsf{RSS}}$ querying $\mathcal{C}_{\mathsf{RSS}}$ and using $\mathsf{x_S}$. The response of oracle Sign is same with the response of the oracle Sign in Section 4.2.1, we omit it here. Oracle $\mathsf{Proof_S}$ is simulated by $\mathcal{A}_{\mathsf{RSS}}$ using $\mathsf{x_S}$. Oracles Redact and $\mathsf{Proof_R}$ are simulated by $\mathcal{A}_{\mathsf{RSS}}$ using $\mathsf{x_R}$.

Challenge. After adaptively querying, $\mathcal{A}_{\mathsf{RSS-IA}}$ chooses two data $(\mathcal{M}_0, \mathcal{M}_1)$ and two redaction subsets $(\mathcal{X}_0, \mathcal{X}_1)$, which satisfy $\mathcal{M}_0 \neq \mathcal{M}_1$ and $\mathcal{M}_0 \setminus \mathcal{X}_0 = \mathcal{M}_1 \setminus \mathcal{X}_1$. $\mathcal{A}_{\mathsf{RSS-IA}}$ passes $(\mathcal{M}_0, \mathcal{X}_0, \mathcal{M}_1, \mathcal{X}_1)$ to $\mathcal{A}_{\mathsf{RSS}}$ as her challenge. Then, $\mathcal{A}_{\mathsf{RSS}}$ passes $(\mathcal{M}_0, \mathcal{X}_0, \mathcal{M}_1, \mathcal{X}_1)$ to $\mathcal{C}_{\mathsf{RSS}}$ and obtains the response $(\mathcal{M}'_b, \sigma'_1)$, where $\mathcal{M}'_b = \mathcal{M}_b \setminus \mathcal{X}_b$ and $b \xleftarrow{R} \{0,1\}$. $\mathcal{A}_{\mathsf{RSS}}$ generates σ_2 and σ_3 by simulating the Redact algorithm of RSS-IA using $\mathsf{x_R}$. $\mathcal{A}_{\mathsf{RSS}}$ returns $(\mathcal{M}'_b, \sigma)$ to $\mathcal{A}_{\mathsf{RSS-IA}}$ as her response, where $\sigma = \{\sigma'_1, \sigma_2, \sigma_3\}$.

Output. $\mathcal{A}_{\mathsf{RSS-IA}}$ returns b' to $\mathcal{A}_{\mathsf{RSS}}$ as her guess. Then $\mathcal{A}_{\mathsf{RSS}}$ returns b' to $\mathcal{C}_{\mathsf{RSS}}$ as her guess.

As defined in Section 3.2.2, if $\mathcal{A}_{\mathsf{RSS-IA}}$ succeeds, then $b' = b$. Hence, $\epsilon_{\mathsf{RSS}} = \epsilon_2$. If ϵ_2 is negligible, ϵ_{RSS} is also negligible. This completes the proof. □

4.2.3 Transparency of the Generic RSS-IA

Theorem 3. *Our* RSS-IA *satisfies transparency in the random oracle model, if the underlying* RSS *satisfies transparency and the* DDH *problem is hard.*

Proof. Let $\mathcal{A}_{\mathsf{RSS-IA}}$ be a PPT attacker who can win the transparency game (defined in Section 3.2.3) of our RSS-IA with the success advantage of ϵ_3. If $\mathcal{A}_{\mathsf{RSS-IA}}$ exists, we can construct an attacker $\mathcal{A}_{\mathsf{RSS}}$ who can break the transparency property of the underlying RSS with the success advantage of ϵ_{RSS}. $\mathcal{A}_{\mathsf{RSS}}$ acts as the challenger of $\mathcal{A}_{\mathsf{RSS-IA}}$ as follows.

Setup. The challenger $\mathcal{C}_{\mathsf{RSS}}$ generates a key pair $(\mathsf{pk}_{\mathsf{RSS}}, \mathsf{sk}_{\mathsf{RSS}})$ of an RSS and sends $\mathsf{pk}_{\mathsf{RSS}}$ to $\mathcal{A}_{\mathsf{RSS}}$. $\mathcal{A}_{\mathsf{RSS}}$ chooses two random numbers $\mathsf{x_S}$ and $\mathsf{x_R}$ from $\mathbb{Z}_p^*$, and calculates $\mathsf{y_S} = g^{\mathsf{x_S}}$ and $\mathsf{y_R} = g^{\mathsf{x_R}}$. Then, $\mathcal{A}_{\mathsf{RSS}}$ sends $(\mathsf{pk}_{\mathsf{RSS}}, \mathsf{y_S}, \mathsf{y_R})$ to $\mathcal{A}_{\mathsf{RSS-IA}}$. $H_1 : \{0,1\}^* \rightarrow \mathbb{G}$ and $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_p$ are two random oracles simulated by $\mathcal{A}_{\mathsf{RSS}}$.

Queries. $\mathcal{A}_{\mathsf{RSS-IA}}$ can adaptively query oracles $\{H_1, H_2, \mathsf{Sign}, \mathsf{Proof_S}, \mathsf{Redact}, \mathsf{Proof_R}\}$. Oracle Sign is simulated by $\mathcal{A}_{\mathsf{RSS}}$ querying $\mathcal{C}_{\mathsf{RSS}}$ and using $\mathsf{x_S}$. The response of oracle Sign is same with the response of the oracle Sign in Section 4.2.1, we omit it here. Oracle $\mathsf{Proof_S}$ is simulated by $\mathcal{A}_{\mathsf{RSS}}$ using $\mathsf{x_S}$. Oracles Redact and $\mathsf{Proof_R}$ are simulated by $\mathcal{A}_{\mathsf{RSS}}$ using $\mathsf{x_R}$.

Challenge. After adaptively querying, $\mathcal{A}_{\mathsf{RSS-IA}}$ chooses a data $\mathcal{M}$ and a redaction subset $\mathcal{X} \subset \mathcal{M}$. $\mathcal{A}_{\mathsf{RSS-IA}}$ passes $(\mathcal{M}, \mathcal{X})$ to $\mathcal{A}_{\mathsf{RSS}}$ as her challenge. Then, $\mathcal{A}_{\mathsf{RSS}}$ passes $(\mathcal{M}, \mathcal{X})$ to $\mathcal{C}_{\mathsf{RSS}}$ and gets the response $(\mathcal{M}', \sigma'_{1,b})$, where $\mathcal{M}' = \mathcal{M} \setminus \mathcal{X}$ and $b \in \{0,1\}$. $\mathcal{A}_{\mathsf{RSS}}$ chooses σ_2 from $\mathbb{Z}_p^*$ and generates σ_3 in the random oracle model. Then, $\mathcal{A}_{\mathsf{RSS}}$ returns $(\mathcal{M}', \sigma)$ to $\mathcal{A}_{\mathsf{RSS-IA}}$ as her response, where $\sigma = \{\sigma'_{1,b}, \sigma_2, \sigma_3\}$.

Output. $\mathcal{A}_{\mathsf{RSS-IA}}$ returns b' to $\mathcal{A}_{\mathsf{RSS}}$ as her guess. Then $\mathcal{A}_{\mathsf{RSS}}$ returns b' to $\mathcal{C}_{\mathsf{RSS}}$ as her guess.

As defined in Section 3.2.3, if $\mathcal{A}_{\mathsf{RSS-IA}}$ succeeds, then $b' = b$. The hardness of the DDH problem ensures that $\mathcal{A}_{\mathsf{RSS-IA}}$ can obtain valid information only from $\sigma'_{1,b}$ to decide whether $\sigma'_{1,b}$ is the output of the Sign algorithm or the Redact algorithm. Hence, $\epsilon_{\mathsf{RSS}} = \epsilon_3$. If ϵ_3 is negligible, ϵ_{RSS} is also negligible. This completes the proof. □

4.2.4 Signer-Accountability of the Generic RSS-IA

Theorem 4. *Our* RSS-IA *satisfies signer-accountability in the random oracle model, if the* CDH *problem is hard.*

Proof. Let $\mathcal{A}_{\mathsf{RSS-IA}}$ be a PPT attacker who can win the signer-accountability game (defined in Section 3.2.4) of our RSS-IA with the success probability of ϵ_4. If $\mathcal{A}_{\mathsf{RSS-IA}}$ exists, we can construct an attacker $\mathcal{A}_{\mathsf{CDH}}$ who can solve the CDH problem with the success probability of ϵ_{CDH}. $\mathcal{A}_{\mathsf{CDH}}$ simulates the challenger of $\mathcal{A}_{\mathsf{RSS-IA}}$ as follows.

Setup. The challenger $\mathcal{C}_{\mathsf{CDH}}$ chooses an Abelian group $\mathbb{G}$ of prime order p with generator g and two random numbers $\{x_1, x_2\}$ from $\mathbb{Z}_p^*$. Then she passes $\{\mathbb{G}, p, g, g^{x_1}, g^{x_2}\}$ to $\mathcal{A}_{\mathsf{CDH}}$. $\mathcal{A}_{\mathsf{CDH}}$ sets $\mathsf{y_R} = g^{x_1}$, where the secret key $\mathsf{x_R}$ is equivalent to x_1. $\mathcal{A}_{\mathsf{CDH}}$ passes $\mathsf{y_R}$ to $\mathcal{A}_{\mathsf{RSS-IA}}$. $\mathcal{A}_{\mathsf{RSS-IA}}$ generates a key pair $(\mathsf{pk}_{\mathsf{RSS}}, \mathsf{sk}_{\mathsf{RSS}})$ of an RSS, chooses a random number $\mathsf{x_S}$ from $\mathbb{Z}_p^*$ and calculates $\mathsf{y_S} = g^{\mathsf{x_S}}$. $\mathcal{A}_{\mathsf{RSS-IA}}$ passes $(\mathsf{pk}_{\mathsf{RSS}}, \mathsf{y_S})$ to $\mathcal{A}_{\mathsf{CDH}}$. $H_1 : \{0,1\}^* \rightarrow \mathbb{G}$ and $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_p$ are two random oracles simulated by $\mathcal{A}_{\mathsf{CDH}}$.

Queries. $\mathcal{A}_{\mathsf{RSS-IA}}$ can adaptively query oracles $\{H_1, H_2, \mathsf{Redact}, \mathsf{Proof_R}\}$, which are simulated by $\mathcal{A}_{\mathsf{CDH}}$. Let q_s be the queried number of Redact oracle. Before receiving queries, $\mathcal{A}_{\mathsf{CDH}}$ initializes an empty hash list $\mathbb{H}$. The details of queries and responses are as follows.

Hash − Queries. Let $\{\mathcal{M}_i, \sigma_{1,i}\}$ be the ith hash query to H_1 oracle from $\mathcal{A}_{\mathsf{RSS-IA}}$. $\mathcal{A}_{\mathsf{CDH}}$ sets $h_i = H_1(\mathcal{M}_i, \sigma_{1,i}) = g^{x_2 + r_i}$ with probability $\psi = \frac{1}{q_s+1}$, and $h_i = H_1(\mathcal{M}_i, \sigma_{1,i}) = g^{r_i}$ with probability $1 - \psi$, where $r_i \xleftarrow{R} \mathbb{Z}_p^*$. $\mathcal{A}_{\mathsf{CDH}}$ returns h_i to $\mathcal{A}_{\mathsf{RSS-IA}}$ as her response, and adds $(\mathcal{M}_i, \sigma_{1,i}, r_i, h_i)$ into the hash list $\mathbb{H}$.

Redact−Queries. Let $(\mathcal{M}_j, \sigma_j, \mathcal{X}_j)$ be the jth query to Redact oracle from $\mathcal{A}_{\mathsf{RSS-IA}}$, where $\sigma_j = \{\sigma_{1,j}, \sigma_{2,j}, \sigma_{3,j}\}$. $\mathcal{A}_{\mathsf{CDH}}$ searches the hash list $\mathbb{H}$ and obtains $(\mathcal{M}_j, \sigma_{1,j}, r_j, h_j)$. If $h_j \neq g^{r_j}$, the simulation of $\mathcal{A}_{\mathsf{CDH}}$ aborts. Otherwise, i.e., $h_j = g^{r_j}$, $\mathcal{A}_{\mathsf{CDH}}$ sets $\mathcal{M}'_j \leftarrow \mathcal{M}_j \setminus \mathcal{X}_j$ and $\sigma'_{2,j} = (g^{x_1})^{r_j} = {h_j}^{x_1}$. $\mathcal{A}_{\mathsf{CDH}}$ generates $\sigma'_{3,j}$ in the random oracle model. Then $\mathcal{A}_{\mathsf{CDH}}$ returns $(\mathcal{M}'_j, \sigma'_j)$ to $\mathcal{A}_{\mathsf{RSS-IA}}$ as her response, where $\sigma'_j = \{\sigma'_{1,j}, \sigma'_{2,j}, \sigma'_{3,j}\}$.

$\mathsf{Proof_R}$−Queries. Let $(\mathcal{M}_k, \sigma_k)$ be the kth query to $\mathsf{Proof_R}$ oracle from $\mathcal{A}_{\mathsf{RSS-IA}}$, where $\sigma_k = \{\sigma_{1,k}, \sigma_{2,k}, \sigma_{3,k}\}$. If $\sigma_{2,k} \neq H_1(\mathcal{M}_k, \sigma_{1,k})^{\mathsf{x_R}}$, $\mathcal{A}_{\mathsf{CDH}}$ generates evidence tag $\pi_k = \{\pi_{1,k}, \pi_{2,k}, \pi_{3,k}, \pi_{4,k}\}$ in the random oracle model. Otherwise, i.e., $\sigma_{2,k} = H_1(\mathcal{M}_k, \sigma_{1,k})^{\mathsf{x_R}}$, $\mathcal{A}_{\mathsf{CDH}}$ generates evidence tag $\pi_k = \{\pi_{1,k}, \pi_{2,k}\}$ in the random oracle model. $\mathcal{A}_{\mathsf{CDH}}$ returns π_k to $\mathcal{A}_{\mathsf{RSS-IA}}$ as her response.

Output. $\mathcal{A}_{\mathsf{RSS-IA}}$ outputs her forgery evidence tag π^* for a data/signature pair $(\mathcal{M}^*, \sigma^*)$, where $\sigma^* = ({\sigma_1}^*, {\sigma_2}^*, {\sigma_3}^*)$.

As defined in Section 3.2.4, if $\mathcal{A}_{\mathsf{RSS-IA}}$ succeeds, then $\mathcal{M}^*$ is not the response of Redact oracle, σ^* is valid, and $\mathsf{Judge}(\mathsf{pk}_{\mathsf{RSS}}, \mathsf{y_S}, \mathsf{y_R}, \mathcal{M}^*, \sigma^*, \pi^*) = \mathsf{Redactor}$.

If $\pi^* = \{\pi_1, \pi_2\}$, let (ω, h^*) be the hash query to oracle H_2 from $\mathcal{A}_{\mathsf{RSS-IA}}$, where $\omega \in \mathbb{Z}_p^*$ and $h^* = H_1(\mathcal{M}^*, {\sigma_1}^*)$. $\mathcal{A}_{\mathsf{CDH}}$ chooses a random number π_1 from $\mathbb{Z}_p^*$, sets $\pi_1 = H_2(g^\omega, (h^*)^\omega)$, and returns π_1 as the response. $\mathcal{A}_{\mathsf{RSS-IA}}$ calculates $\pi_2 = \omega - \pi_1 \mathsf{x_S}$. Then $\mathcal{A}_{\mathsf{CDH}}$ obtains $\pi_1 = H_2(g^\omega, (h^*)^\omega) = H_2(g^{\pi_2} \mathsf{y_R}^{\pi_1}, (h^*)^{\pi_2} (\sigma_2^*)^{\pi_1})$. Using the forking lemma in [24], $\mathcal{A}_{\mathsf{CDH}}$ resets the hash response for (ω, h^*) in the random oracle model. $\mathcal{A}_{\mathsf{CDH}}$ chooses a random number π'_1 $(\pi'_1 \neq \pi_1)$ from $\mathbb{Z}_p^*$, sets $\pi'_1 = H_2(g^\omega, (h^*)^\omega)$, and returns π'_1 as the response. $\mathcal{A}_{\mathsf{RSS-IA}}$ calculates $\pi'_2 = \omega - \pi'_1 \mathsf{x_S}$. Then $\mathcal{A}_{\mathsf{CDH}}$ obtains $\pi'_1 = H_2(g^\omega, (h^*)^\omega) = H_2(g^{\pi'_2} \mathsf{y_R}^{\pi'_1}, (h^*)^{\pi'_2} (\sigma_2^*)^{\pi'_1})$. Since $\pi'_1 \neq \pi_1$, thus $\pi'_2 \neq \pi_2$. $\mathcal{A}_{\mathsf{CDH}}$ gets $g^\omega = g^{\pi_2} \mathsf{y_R}^{\pi_1} = g^{\pi'_2} \mathsf{y_R}^{\pi'_1}$ and $(h^*)^\omega = (h^*)^{\pi_2} (\sigma_2^*)^{\pi_1} = (h^*)^{\pi'_2} (\sigma_2^*)^{\pi'_1}$. Then $\mathcal{A}_{\mathsf{CDH}}$ obtains $\mathsf{y_R} = g^{(\pi'_2 - \pi_2)(\pi_1 - \pi'_1)^{-1}}$ and $\sigma_2^* = (h^*)^{(\pi'_2 - \pi_2)(\pi_1 - \pi'_1)^{-1}}$. $\mathcal{A}_{\mathsf{CDH}}$ gains $x_1 = (\pi'_2 - \pi_2)(\pi_1 - \pi'_1)^{-1}$ and solves the DL problem. Thereby, $\mathcal{A}_{\mathsf{CDH}}$ obtains $g^{x_1 \cdot x_2}$ and solves the CDH problem. In this case, $\epsilon_{\mathsf{CDH}} = \epsilon_4$. If ϵ_4 is negligible, ϵ_{CDH} is also negligible.

If $\pi^* = \{\pi_1, \pi_2, \pi_3, \pi_4\}$, since $\pi_1 = ((h^*)^{\mathsf{x_S}} / (\sigma_2^*))^{\omega_1} \neq 1$, thus $\sigma_2^* \neq (h^*)^{\mathsf{x_S}}$. Hence, if $\mathcal{A}_{\mathsf{RSS-IA}}$ succeeds, there must be $\sigma_2^* = (h^*)^{\mathsf{x_R}}$. If $\mathcal{A}_{\mathsf{CDH}}$ succeeds, then $h^* = H_1(\mathcal{M}^*, {\sigma_1}^*) = g^{x_2 + r^*}$, where $(\mathcal{M}^*, \sigma_1^*, r^*, h^*)$ exists in the hash list $\mathbb{H}$. Actually, $\sigma_2^* = (h^*)^{x_1} = (g^{x_2 + r^*})^{x_1} = g^{x_1 \cdot x_2 + x_1 \cdot r^*}$. $\mathcal{A}_{\mathsf{CDH}}$ obtains $\frac{\sigma_2^*}{(g^{x_1})^{r^*}} = \frac{g^{x_1 \cdot x_2 + x_1 \cdot r^*}}{(g^{x_1})^{r^*}} = g^{x_1 \cdot x_2}$ and solves the CDH problem. In this case, $\epsilon_{\mathsf{CDH}} = (1 - \psi)^{q_s} \psi \epsilon_4 = \frac{q_s^{q_s} \epsilon_4}{(q_s+1)^{(q_s+1)}}$. If ϵ_4 is negligible, ϵ_{CDH} is also negligible. This completes the proof. □

4.2.5 Redactor-Accountability of the Generic RSS-IA

Theorem 5. *Our* RSS-IA *satisfies redactor-accountability in the random oracle model, if the* CDH *problem is hard.*

Proof. Let $\mathcal{A}_{\mathsf{RSS-IA}}$ be a PPT attacker who can win the redactor-accountability game (defined in Section 3.2.5) of our $\mathsf{RSS\text{-}IA}$ with the success probability of ϵ_5. If $\mathcal{A}_{\mathsf{RSS-IA}}$ exists, we can construct an attacker $\mathcal{A}_{\mathsf{CDH}}$ who can solve the CDH problem with the success probability of ϵ_{CDH}. $\mathcal{A}_{\mathsf{CDH}}$ simulates the challenger of $\mathcal{A}_{\mathsf{RSS-IA}}$ as follows.

Setup. The challenger $\mathcal{C}_{\mathsf{CDH}}$ chooses an Abelian group $\mathbb{G}$ of prime order p with generator g and two random numbers $\{x_1, x_2\}$ from $\mathbb{Z}_p^*$. Then she passes $\{\mathbb{G}, p, g, g^{x_1}, g^{x_2}\}$ to $\mathcal{A}_{\mathsf{CDH}}$. $\mathcal{A}_{\mathsf{CDH}}$ generates a key pair $(\mathsf{pk}_{\mathsf{RSS}}, \mathsf{sk}_{\mathsf{RSS}})$ of an RSS and sets $\mathsf{y_S} = g^{x_1}$, where the secret key $\mathsf{x_S}$ is equivalent to x_1. $\mathcal{A}_{\mathsf{CDH}}$ passes $(\mathsf{pk}_{\mathsf{RSS}}, \mathsf{y_S})$ to $\mathcal{A}_{\mathsf{RSS-IA}}$. $\mathcal{A}_{\mathsf{RSS-IA}}$ chooses a random number $\mathsf{x_R}$ from $\mathbb{Z}_p^*$ and calculates $\mathsf{y_R} = g^{\mathsf{x_R}}$. $\mathcal{A}_{\mathsf{RSS-IA}}$ passes $\mathsf{y_R}$ to $\mathcal{A}_{\mathsf{CDH}}$. $H_1 : \{0,1\}^* \to \mathbb{G}$ and $H_2 : \{0,1\}^* \to \mathbb{Z}_p$ are two random oracles simulated by $\mathcal{A}_{\mathsf{CDH}}$.

Queries. $\mathcal{A}_{\mathsf{RSS-IA}}$ can adaptively query oracles $\{H_1, H_2, \mathsf{Sign}, \mathsf{Proof_S}\}$, which are simulated by $\mathcal{A}_{\mathsf{CDH}}$. Before receiving queries, $\mathcal{A}_{\mathsf{CDH}}$ first initializes an empty hash list $\mathbb{H}$. The details of queries and responses are as follows.

$\mathsf{Sign}$$-$$\mathsf{Queries}$. Let $\mathcal{M}_i$ be the ith signature query to Sign oracle from $\mathcal{A}_{\mathsf{RSS-IA}}$. $\mathcal{A}_{\mathsf{CDH}}$ first generates $\sigma_{1,i}$ for $\mathcal{M}_i$ by using $\mathsf{sk}_{\mathsf{RSS}}$. Then she chooses a random number l_i from $\mathbb{Z}_p^*$ and sets $h_i = H_1(\mathcal{M}_i, \sigma_{1,i}) = g^{l_i}$. $\mathcal{A}_{\mathsf{CDH}}$ generates $\sigma_{2,i} = (h_i)^{x_1} = (g^{l_i})^{x_1} = \mathsf{y_S}^{l_i}$ and $\sigma_{3,i}$ in the random oracle model. $\mathcal{A}_{\mathsf{CDH}}$ returns $\sigma_i = \{\sigma_{1,i}, \sigma_{2,i}, \sigma_{3,i}\}$ to $\mathcal{A}_{\mathsf{RSS-IA}}$ as her response, and adds $(\mathcal{M}_i, \sigma_{1,i}, l_i, h_i)$ into the hash list $\mathbb{H}$.

Hash $-$ $\mathsf{Queries}$. Let $\{\mathcal{M}_j, \sigma_{1,j}\}$ be the jth hash query to H_1 oracle from $\mathcal{A}_{\mathsf{RSS-IA}}$, where $1 \le j \le q_h$. If $\{\mathcal{M}_j, \sigma_{1,j}\}$ exists in the hash list $\mathbb{H}$, $\mathcal{A}_{\mathsf{CDH}}$ gets $h_j = H_1(\mathcal{M}_j, \sigma_{1,j}) = g^{l_j}$. Otherwise, $\mathcal{A}_{\mathsf{CDH}}$ sets $h_j = H_1(\mathcal{M}_j, \sigma_{1,j}) = g^{x_2 + r_j}$ (in which $r_j \overset{R}{\leftarrow} \mathbb{Z}_p^*$), and adds $(\mathcal{M}_j, \sigma_{1,j}, r_j, h_j)$ into the hash list $\mathbb{H}$. Then $\mathcal{A}_{\mathsf{CDH}}$ returns h_j to $\mathcal{A}_{\mathsf{RSS-IA}}$ as her response. Let q_1 be the number of $h_j = g^{x_2 + r_j}$.

$\mathsf{Proof_S}$$-$$\mathsf{Queries}$. Let $(\mathcal{M}_k, \sigma_k)$ be the kth query to $\mathsf{Proof_S}$ oracle from $\mathcal{A}_{\mathsf{RSS-IA}}$, where $\sigma_k = \{\sigma_{1,k}, \sigma_{2,k}, \sigma_{3,k}\}$. If $\sigma_{2,k} \neq H_1(\mathcal{M}_k, \sigma_{1,k})^{\mathsf{x_S}}$, $\mathcal{A}_{\mathsf{CDH}}$ generates evidence tag $\pi_k = \{\pi_{1,k}, \pi_{2,k}, \pi_{3,k}, \pi_{4,k}\}$ in the random oracle model. Otherwise, i.e., $\sigma_{2,k} = H_1(\mathcal{M}_k, \sigma_{1,k})^{\mathsf{x_S}}$, $\mathcal{A}_{\mathsf{CDH}}$ generates evidence tag $\pi_k = \{\pi_{1,k}, \pi_{2,k}\}$ in the random oracle model. $\mathcal{A}_{\mathsf{CDH}}$ returns π_k to $\mathcal{A}_{\mathsf{RSS-IA}}$ as her response.

Output. $\mathcal{A}_{\mathsf{RSS-IA}}$ outputs her forgery evidence tag π^* for a message/signature pair $(\mathcal{M}^*, \sigma^*)$, where $\sigma^* = ({\sigma_1}^*, {\sigma_2}^*, {\sigma_3}^*)$.

As defined in Section 3.2.5, if $\mathcal{A}_{\mathsf{RSS-IA}}$ succeeds, then $\mathcal{M}^*$ has not been queried to Sign oracle, σ^* is valid, and $\mathsf{Judge}(\mathsf{pk}_{\mathsf{RSS}}, \mathsf{y_S}, \mathsf{y_R}, \mathcal{M}^*, \sigma^*, \pi^*) = \mathsf{Signer}$.

If $\pi^* = \{\pi_1, \pi_2\}$, let (ω, h^*) be the hash query to oracle H_2 from $\mathcal{A}_{\mathsf{RSS-IA}}$, where $\omega \in \mathbb{Z}_p^*$ and $h^* = H_1(\mathcal{M}^*, {\sigma_1}^*)$. $\mathcal{A}_{\mathsf{CDH}}$ chooses a random number π_1 from $\mathbb{Z}_p^*$, sets $\pi_1 = H_2(g^\omega, (h^*)^\omega)$, and returns π_1 as the response. $\mathcal{A}_{\mathsf{RSS-IA}}$ calculates $\pi_2 = \omega - \pi_1 \mathsf{x_R}$. Then $\mathcal{A}_{\mathsf{CDH}}$ obtains $\pi_1 = H_2(g^\omega, (h^*)^\omega) = H_2(g^{\pi_2}\mathsf{y_S}^{\pi_1}, (h^*)^{\pi_2}(\sigma_2^*)^{\pi_1})$. Using the forking lemma in [24], $\mathcal{A}_{\mathsf{CDH}}$ resets the hash response for (ω, h^*) in the random oracle model. $\mathcal{A}_{\mathsf{CDH}}$ chooses a random number π_1' $(\pi_1' \neq \pi_1)$ from $\mathbb{Z}_p^*$, sets $\pi_1' = H_2(g^\omega, (h^*)^\omega)$, and returns π_1' as the response. $\mathcal{A}_{\mathsf{RSS-IA}}$ calculates $\pi_2' = \omega - \pi_1' \mathsf{x_R}$. Then $\mathcal{A}_{\mathsf{CDH}}$ gains $\pi_1' = H_2(g^\omega, (h^*)^\omega) = H_2(g^{\pi_2'}\mathsf{y_S}^{\pi_1'}, (h^*)^{\pi_2'}(\sigma_2^*)^{\pi_1'})$. Since $\pi_1' \neq \pi_1$, thus $\pi_2' \neq \pi_2$. $\mathcal{A}_{\mathsf{CDH}}$ obtains $g^\omega = g^{\pi_2}\mathsf{y_S}^{\pi_1} = g^{\pi_2'}\mathsf{y_S}^{\pi_1'}$ and $(h^*)^\omega = (h^*)^{\pi_2}(\sigma_2^*)^{\pi_1} = (h^*)^{\pi_2'}(\sigma_2^*)^{\pi_1'}$. Then $\mathcal{A}_{\mathsf{CDH}}$ gains $\mathsf{y_S} = g^{(\pi_2' - \pi_2)(\pi_1 - \pi_1')^{-1}}$ and $\sigma_2^* = (h^*)^{(\pi_2' - \pi_2)(\pi_1 - \pi_1')^{-1}}$. $\mathcal{A}_{\mathsf{CDH}}$ obtains $x_1 = (\pi_2' - \pi_2)(\pi_1 - \pi_1')^{-1}$ and solves the DL problem. Thereby, $\mathcal{A}_{\mathsf{CDH}}$ obtains $g^{x_1 \cdot x_2}$ and solves the CDH problem. In this case, $\epsilon_{\mathsf{CDH}} = \epsilon_5$. If ϵ_5 is negligible, ϵ_{CDH} is also negligible.

If $\pi^* = \{\pi_1, \pi_2, \pi_3, \pi_4\}$, since $\pi_1 = ((h^*)^{\mathsf{x_R}}/(\sigma_2^*))^{\omega_1} \neq 1$, thus $\sigma_2^* \neq (h^*)^{\mathsf{x_R}}$. Hence, if $\mathcal{A}_{\mathsf{RSS-IA}}$ succeeds, there must be $\sigma_2^* = (h^*)^{\mathsf{x_S}}$. If $\mathcal{A}_{\mathsf{CDH}}$ succeeds, then $h^* = H_1(\mathcal{M}^*, {\sigma_1}^*) = g^{x_2 + r^*}$, where $(\mathcal{M}^*, \sigma_1^*, r^*, h^*)$ is in the hash list $\mathbb{H}$. Actually, $\sigma_2^* = (h^*)^{x_1} = (g^{x_2 + r^*})^{x_1} = g^{x_1 \cdot x_2 + x_1 \cdot r^*}$. $\mathcal{A}_{\mathsf{CDH}}$ gets $\frac{\sigma_2^*}{(g^{x_1})^{r^*}} = \frac{g^{x_1 \cdot x_2 + x_1 \cdot r^*}}{(g^{x_1})^{r^*}} = g^{x_1 \cdot x_2}$ and solves the CDH problem. In this case, $\epsilon_{\mathsf{CDH}} = \frac{q_1}{q_h}\epsilon_5$. If ϵ_5 is negligible, ϵ_{CDH} is also negligible. This completes the proof. □

4.2.6 Collusion-Resistance of the Generic RSS-IA

Theorem 6. *Our* $\mathsf{RSS\text{-}IA}$ *satisfies collusion-resistance in the random oracle model.*

Proof. Let $\mathcal{A}_{\mathsf{RSS-IA}}$ be a PPT attacker who can win the collusion-resistance game defined in Section 3.2.6. $\mathcal{A}_{\mathsf{RSS-IA}}$ possesses the key pairs of the signer and the redactor of $\mathsf{RSS\text{-}IA}$, i.e., $\{(\mathsf{pk}_{\mathsf{RSS}}, \mathsf{sk}_{\mathsf{RSS}}), (\mathsf{y_S}, \mathsf{x_S}), (\mathsf{y_R}, \mathsf{x_R})\}$. $\mathcal{A}_{\mathsf{RSS-IA}}$ can generate two evidence tags for a valid data/signature $(\mathcal{M}, \sigma)$, where $\sigma = \{\sigma_1, \sigma_2, \sigma_3\}$ and $\sigma_3 = \{\beta_1, \beta_2, \beta_3, \beta_4\}$. The evidence tags convince the third party that both the signer and the redactor are the responsible parties for $(\mathcal{M}, \sigma)$ or both are not. $H_1 : \{0,1\}^* \to \mathbb{G}$ and $H_2 : \{0,1\}^* \to \mathbb{Z}_p$ are two random oracles. If $\mathcal{A}_{\mathsf{RSS-IA}}$ exists, $\mathcal{A}_{\mathsf{RSS-IA}}$ can finish the following matters in the random oracle model.

Hash-Query-Output. Let $\{\nu_1, \beta_3, \beta_4\}$ be the hash query to oracle H_2 from $\mathcal{A}_{\mathsf{RSS-IA}}$, where $\nu_1 \in \mathbb{Z}_p^*$, $\beta_3 = \nu_2 - \beta_4 \mathsf{x_R}$ and $\nu_2 \in \mathbb{Z}_p^*$. To response the hash query, we choose $\rho \in \mathbb{Z}_p^*$, set $\rho = H_2(g^{\nu_1}, h^{\nu_1}, g^{\beta_3}\mathsf{y_R}^{\beta_4}, h^{\beta_3}\sigma_2^{\beta_4})$, and return ρ as the response. According to ρ, $\mathcal{A}_{\mathsf{RSS-IA}}$ calculates $\beta_2 = \rho - \beta_4$ and $\beta_1 = \nu_1 - \beta_2 \mathsf{x_S}$. $\mathcal{A}_{\mathsf{RSS-IA}}$ outputs a data/signature pair $(\mathcal{M}, \sigma)$, where $\sigma = \{\sigma_1, \sigma_2, \sigma_3\}$ and $\sigma_3 = \{\beta_1, \beta_2, \beta_3, \beta_4\}$. Then we obtain $\rho = \beta_2 + \beta_4 = H_2(g^{\beta_1}\mathsf{y_S}^{\beta_2}, h^{\beta_1}\sigma_2^{\beta_2}, g^{\beta_3}\mathsf{y_R}^{\beta_4}, h^{\beta_3}\sigma_2^{\beta_4})$.

Reset-Hash-Query-Output. Using the forking lemma in [24], we reset the hash response for $\{\nu_1, \beta_3, \beta_4\}$. We choose $\rho' \in \mathbb{Z}_p^*$ $(\rho' \neq \rho)$, set $\rho' = H_2(g^{\nu_1}, h^{\nu_1}, g^{\beta_3}\mathsf{y_R}^{\beta_4}, h^{\beta_3}\sigma_2^{\beta_4})$, and return ρ' as the response. Similarly, according to ρ', $\mathcal{A}_{\mathsf{RSS-IA}}$ calculates $\beta_2' = \rho' - \beta_4'$ and $\beta_1' = \nu_1 - \beta_2' \mathsf{x_S}$. Finally $\mathcal{A}_{\mathsf{RSS-IA}}$ outputs a data/signature pair $(\mathcal{M}, \sigma)$, where $\sigma = \{\sigma_1', \sigma_2', \sigma_3'\}$ and $\sigma_3' = \{\beta_1', \beta_2', \beta_3, \beta_4\}$. Then we obtain $\rho' = \beta_2' + \beta_4' = H_2(g^{\beta_1'}\mathsf{y_S}^{\beta_2'}, h^{\beta_1'}\sigma_2^{\beta_2'}, g^{\beta_3}\mathsf{y_R}^{\beta_4}, h^{\beta_3}\sigma_2^{\beta_4})$.

Since $\rho' \neq \rho$, there must be $\beta_2 \neq \beta_2'$ or $\beta_4 \neq \beta_4'$. We obtain $g^{\nu_1} = g^{\beta_1}\mathsf{y_S}^{\beta_2} = g^{\beta_1'}\mathsf{y_S}^{\beta_2'}$ and $h^{\nu_1} = h^{\beta_1}\sigma_2^{\beta_2} = h^{\beta_1'}\sigma_2^{\beta_2'}$. Then we obtain $\mathsf{y_S} = g^{(\beta_1 - \beta_1')(\beta_2' - \beta_2)^{-1}}$ and $\sigma_2 = h^{(\beta_1 - \beta_1')(\beta_2' - \beta_2)^{-1}}$. Similarly, we can also get $\mathsf{y_R} = g^{(\beta_3 - \beta_3')(\beta_4' - \beta_4)^{-1}}$ and $\sigma_2 = h^{(\beta_3 - \beta_3')(\beta_4' - \beta_4)^{-1}}$. It means that if a data/signature pair is valid, there must be $\sigma_2 = h^{\mathsf{x_S}}$ or $\sigma_2 = h^{\mathsf{x_R}}$. Thus, we obtain a contradiction. This completes the proof. □

5 EXTENDED RSS-IA

In this section, we propose an extended generic RSS-IA with multiple designated redactors. Then, we discuss its security.

5.1 Generic RSS-IA With Multi-Redactor

The extended design is also a generic transformation that adds implicit accountability to any transparent RSSs. In the extended design, the evidence tag can be generated by the original signer and any one of the N designated redactors, independently. This is the major difference from our first RSS-IA constructed in Section 4.1, in which there is only one designated redactor.

The extended RSS-IA with multiple designated redactors consists of seven algorithms: KGen, Sign, Redact, Verify, $\mathsf{Proof_S}$, $\mathsf{Proof_R}$ and Judge. Denote the identity of the signer by Signer, and the identity of the ith redactor by Redactor$-$i, where $i \in \{1, 2, \ldots, N\}$.

$\mathsf{KGen}(1^\lambda)$. The key generation is similar to the KGen algorithm in Section 4.1.

1) The secret key $\mathsf{SK_S}$ of the signer remains as $(\mathsf{sk_{RSS}}, \mathsf{x_S})$, and public key $\mathsf{PK_S}$ remains as $(\mathsf{pk_{RSS}}, \mathsf{y_S})$.
2) The secret key $\mathsf{SK_{R,i}}$ of Redactor$-$i is $\mathsf{x_{R,i}} \xleftarrow{R} \mathbb{Z}_p^*$, and public key $\mathsf{PK_{R,i}}$ is $\mathsf{y_{R,i}} = g^{\mathsf{x_{R,i}}}$.

$\mathsf{Sign}(\mathsf{SK_S}, \bigcup_{i=1}^{N}\mathsf{PK_{R,i}}, \mathcal{M})$. The signature σ of a data $\mathcal{M}$ consists of three components: $(\sigma_1, \sigma_2, \sigma_3)$.

1) σ_1 is the output of the RSS.Sign: It is generated by running the Step 1 of Sign algorithm in Section 4.1.
2) σ_2 is an auxiliary tag for accountability: It is generated by running the Step 2 of Sign algorithm in Section 4.1.
3) σ_3 is a proof of the statement $\sigma_2 = h^{\mathsf{x_S}}$ or $\sigma_2 = h^{\mathsf{x_{R,i}}}$:
   a) Chooses $\{\nu, \bigcup_{i=1}^{N}\beta_{3,i}, \bigcup_{i=1}^{N}\beta_{4,i}\} \xleftarrow{R} \mathbb{Z}_p^*$, and calculates $\beta_2 = H_2(g^\nu, h^\nu, g^{\beta_{3,1}}\mathsf{y_{R,1}}^{\beta_{4,1}}, h^{\beta_{3,1}}\sigma_2^{\beta_{4,1}}, g^{\beta_{3,2}}\mathsf{y_{R,2}}^{\beta_{4,2}}, h^{\beta_{3,2}}\sigma_2^{\beta_{4,2}}, \ldots, g^{\beta_{3,i}}\mathsf{y_{R,i}}^{\beta_{4,i}}, h^{\beta_{3,i}}\sigma_2^{\beta_{4,i}}, \ldots, g^{\beta_{3,N}}\mathsf{y_{R,N}}^{\beta_{4,N}}, h^{\beta_{3,N}}\sigma_2^{\beta_{4,N}}) - \sum_{i=1}^{N}\beta_{4,i} \bmod p$, and $\beta_1 = \nu - \beta_2\mathsf{x_S} \bmod p$.
   b) Denotes $\sigma_3 = \{\beta_1, \beta_2, \bigcup_{i=1}^{N}\beta_{3,i}, \bigcup_{i=1}^{N}\beta_{4,i}\}$.
4) Returns the signature σ for the data $\mathcal{M}$, where $\sigma = \{\sigma_1, \sigma_2, \sigma_3\}$.

$\mathsf{Redact}(\mathsf{SK_{R,i}}, \mathsf{PK_S}, \bigcup_{i=1}^{N}\mathsf{PK_{R,i}}, \mathcal{M}, \sigma, \mathcal{X})$. $(\mathcal{M}, \sigma)$ is a valid data/signature pair, where $\sigma = \{\sigma_1, \sigma_2, \sigma_3\}$ and $\sigma_3 = \{\beta_1, \beta_2, \bigcup_{i=1}^{N}\beta_{3,i}, \bigcup_{i=1}^{N}\beta_{4,i}\}$. The signature σ' of a redacted data $\mathcal{M}'$ consists of three components: $(\sigma'_1, \sigma'_2, \sigma'_3)$. $\mathcal{X} \subseteq \mathcal{M}$ is a redaction subset. Redactor$-$i does as follows.

1) σ'_1 is the output of the RSS.Redact: It is generated by running the Step 1 of Redact algorithm in Section 4.1.
2) σ'_2 is an auxiliary tag for accountability:
   a) Calculates $h = H_1(\mathcal{M}', \sigma'_1)$.
   b) Calculates $\sigma'_2 = h^{\mathsf{x_{R,i}}}$.
3) σ'_3 is a proof of the statement $\sigma'_2 = h^{\mathsf{x_S}}$ or $\sigma'_2 = h^{\mathsf{x_{R,i}}}$:
   a) Chooses $\{\nu', \beta'_1, \beta'_2\} \xleftarrow{R} \mathbb{Z}_p^*$, calculate $\beta'_{4,i} = H_2(g^{\beta'_1}\mathsf{y_S}^{\beta'_2}, h^{\beta'_1}{\sigma'_2}^{\beta'_2}, g^{\beta_{3,1}}\mathsf{y_{R,1}}^{\beta_{4,1}}, h^{\beta_{3,1}}{\sigma'_2}^{\beta_{4,1}}, \ldots, g^{\beta_{3,i-1}}\mathsf{y_{R,i-1}}^{\beta_{4,i-1}}, h^{\beta_{3,i-1}}{\sigma'_2}^{\beta_{4,i-1}}, g^{\nu'}, h^{\nu'}, g^{\beta_{3,i+1}}\mathsf{y_{R,i+1}}^{\beta_{4,i+1}}, h^{\beta_{3,i+1}}{\sigma'_2}^{\beta_{4,i+1}}, \ldots, g^{\beta_{3,N}}\mathsf{y_{R,N}}^{\beta_{4,N}}, h^{\beta_{3,N}}{\sigma'_2}^{\beta_{4,N}}) - \beta'_2 \bmod p$, and $\beta'_{3,i} = \nu' - \beta'_{4,i}\mathsf{x_{R,i}} \bmod p$.
   b) Denotes $\sigma'_3 = \{\beta'_1, \beta'_2, \beta'_{3,i}, \beta'_{4,i}, \bigcup_{j=1}^{N}\beta_{3,j}\backslash\beta_{3,i}, \bigcup_{j=1}^{N}\beta_{4,j}\backslash\beta_{4,i}\}$.
4) Returns the signature σ' for the data $\mathcal{M}'$, where $\sigma' = \{\sigma'_1, \sigma'_2, \sigma'_3\}$.

$\mathsf{Verify}(\mathsf{PK_S}, \bigcup_{i=1}^{N}\mathsf{PK_{R,i}}, \mathcal{M}, \sigma)$. Given a data/signature pair $(\mathcal{M}, \sigma)$, where $\sigma = \{\sigma_1, \sigma_2, \sigma_3\}$ and $\sigma_3 = \{\beta_1, \beta_2, \bigcup_{i=1}^{N}\beta_{3,i}, \bigcup_{i=1}^{N}\beta_{4,i}\}$, the verifier does:

1) Verify the validity of σ_1: σ_1 is verified by running the Step 1 of Verify algorithm in Section 4.1.
2) Verify the validity of σ_2:
   a) Calculates $h = H_1(\mathcal{M}, \sigma_1)$.
   b) If $\beta_2 + \sum_{i=1}^{N}\beta_{4,i} = H_2(g^{\beta_1}\mathsf{y_S}^{\beta_2}, h^{\beta_1}\sigma_2^{\beta_2}, g^{\beta_{3,1}} y_{R,1}^{\beta_{4,1}}, h^{\beta_{3,1}}\sigma_2^{\beta_{4,1}}, g^{\beta_{3,2}}\mathsf{y_{R,2}}^{\beta_{4,2}}, h^{\beta_{3,2}}\sigma_2^{\beta_{4,2}}, \ldots, g^{\beta_{3,i}}\mathsf{y_{R,i}}^{\beta_{4,i}}, h^{\beta_{3,i}}\sigma_2^{\beta_{4,i}}, \ldots, g^{\beta_{3,N}}\mathsf{y_{R,N}}^{\beta_{4,N}}, h^{\beta_{3,N}}\sigma_2^{\beta_{4,N}})$, it indicates σ_2 is valid, returns 1.
3) Otherwise, returns 0.

$\mathsf{Proof_S}(\mathsf{SK_S}, \bigcup_{i=1}^{N}\mathsf{PK_{R,i}}, \mathcal{M}, \sigma)$. $(\mathcal{M}, \sigma)$ is a valid data/signature pair, where $\sigma = \{\sigma_1, \sigma_2, \sigma_3\}$ and $\sigma_3 = \{\beta_1, \beta_2, \bigcup_{i=1}^{N}\beta_{3,i}, \bigcup_{i=1}^{N}\beta_{4,i}\}$. The signer generates the evidence tag π for $(\mathcal{M}, \sigma)$ by running the $\mathsf{Proof_S}$ algorithm in Section 4.1.

$\mathsf{Proof_R}(\mathsf{SK_{R,i}}, \mathsf{PK_S}, \bigcup_{i=1}^{N}\mathsf{PK_{R,i}}, \mathcal{M}, \sigma)$. $(\mathcal{M}, \sigma)$ is a valid data/signature pair, where $\sigma = \{\sigma_1, \sigma_2, \sigma_3\}$ and $\sigma_3 = \{\beta_1, \beta_2, \bigcup_{i=1}^{N}\beta_{3,i}, \bigcup_{i=1}^{N}\beta_{4,i}\}$. Using her key pair $(\mathsf{y_{R,i}}, \mathsf{x_{R,i}})$, Redactor$-$i generates the evidence tag π for $(\mathcal{M}, \sigma)$ by running the $\mathsf{Proof_R}$ algorithm in Section 4.1.

$\mathsf{Judge}(\mathsf{PK_S}, \bigcup_{i=1}^{N}\mathsf{PK_{R,i}}, \mathcal{M}, \sigma, \pi)$. $(\mathcal{M}, \sigma)$ is a valid message/signature pair, where $\sigma = \{\sigma_1, \sigma_2, \sigma_3\}$ and $\sigma_3 = \{\beta_1, \beta_2, \bigcup_{i=1}^{N}\beta_{3,i}, \bigcup_{i=1}^{N}\beta_{4,i}\}$. π is an evidence tag for $(\mathcal{M}, \sigma)$. To determine the responsible party for $(\mathcal{M}, \sigma)$, the auditor does:

1) Calculates $h = H_1(\mathcal{M}, \sigma_1)$.
2) If $\pi = \{\pi_1, \pi_2\} \in \mathbb{Z}_p^* \times \mathbb{Z}_p^*$:
   a) If $\pi_1 = H_2(g^{\pi_2}\mathsf{y_S}^{\pi_1}, h^{\pi_2}\sigma_2^{\pi_1})$, returns Signer.
   b) If $\pi_1 = H_2(g^{\pi_2}\mathsf{y_{R,i}}^{\pi_1}, h^{\pi_2}\sigma_2^{\pi_1})$, returns Redactor$-$i.
3) Else, if $\pi = \{\pi_1, \pi_2, \pi_3, \pi_4\} \in \mathbb{G} \times \mathbb{Z}_p^* \times \mathbb{Z}_p^* \times \mathbb{Z}_p^*$:
   a) If $\pi_1 \neq 1$ and for $\forall i \in \{1, 2, \cdots, N\}$, $\pi_2 = H_2(\pi_1, g^{\pi_4}/\mathsf{y_{R,i}}^{\pi_3}, h^{\pi_4}/(\sigma_2^{\pi_3}\pi_1^{\pi_2}))$, returns Signer.
   b) If $\pi_1 \neq 1$, $\pi_2 = H_2(\pi_1, g^{\pi_4}/\mathsf{y_S}^{\pi_3}, h^{\pi_4}/(\sigma_2^{\pi_3}\pi_1^{\pi_2}))$, and for $\forall j \in \{1, 2, \cdots, i-1, i+1, \cdots, N\}$, $\pi_2 = H_2(\pi_1, g^{\pi_4}/\mathsf{y_{R,j}}^{\pi_3}, h^{\pi_4}/(\sigma_2^{\pi_3}\pi_1^{\pi_2}))$, returns Redactor$-$i.
4) Otherwise, returns $\perp$.

5.2 Analysis of the Extended RSS-IA

The correctness of the extended RSS-IA with multiple redactors can be verified straightforwardly. Its security depends on the security properties of our first generic RSS-IA constructed in Section 4.1, which have been proved in Section 4.2. Thus, the extended RSS-IA provides unforgeability, privacy, transparency, signer-accountability, redactor-accountability, and collusion-resistance. The security proof of the extended RSS-IA is similar to our first generic RSS-IA, and thus is omitted due to space limitation.

6 APPLICATIONS AND EXPERIMENTS

Our RSS-IA designs are generic transformations that add implicit-accountability to any transparent RSSs. In this section, we first give examples to show the applications of our RSS-IA range from privacy protection to accountable and transparent authenticated data redaction. Then, we implement experiments to validate the effectiveness of the proposed generic transformations.

TABLE 1
The Original EHR of Bob

Name	Gender	ID	Day	Symptom	Prescription
Bob	Male	1234	8 Mar., 1990	Symptom 1	Prescription 1
			12 May., 1991	Symptom 2	Prescription 2
			⋮	⋮	⋮
			1 Jul., 2019	Symptom 99	Prescription 99
			4 Sep., 2019	Symptom 100	Prescription 100

TABLE 2
A Redacted EHR of Bob

Gender	Day	Symptom	Prescription
Male	8 Mar., 1990	Symptom 1	Prescription 1
	12 May., 1991	Symptom 2	Prescription 2
	⋮	⋮	⋮
	1 Jul., 2019	Symptom 99	Prescription 99
	4 Sep., 2019	Symptom 100	Prescription 100

6.1 Applications

As a simple application example, let $\mathcal{M}$ be the original EHR of Bob, which is shown in Table 1. Let σ be the signature of $\mathcal{M}$ generated by the original signer. $(\mathcal{M}, \sigma)$ is kept by Bob or some authorised parties. In real life, even if the original signer is offline, $(\mathcal{M}, \sigma)$ can still be used for various situations with our RSS-IA.

Use Case #1: Our RSS-IA satisfies the privacy protection requirement of authenticated EHR redaction in the case that EHR is used for scientific research. It ensures that the redacted EHR/signature pair $(\mathcal{M}', \sigma')$ reveals no information about Bob's name and ID, which are deleted from $\mathcal{M}$. The redacted EHR $\mathcal{M}'$ is shown in Table 2.

Use Case #2: Our RSS-IA satisfies the transparency requirement of authenticated EHR redaction in the case that EHR is used as the supporting material of applying for subvention. It ensures that no third party can determine whether the submitted EHR is the original version or redacted one, in which some sensitive symptoms and treatments are removed. It avoids discrimination against patients with certain medical history.

Use Case #3: Our RSS-IA satisfies the accountability requirement in authenticated EHR redaction in the case that EHR is used as the supporting material of health insurance purchasing. It prevents the buyer from maliciously removing his sensitive medical history from his EHR, to reduce insurance price and claim higher insurance compensation. With our scheme, Bob can only remove the unrelevant parts from his original EHR and generate a signature to prove the authenticity of the redacted EHR. Accountability ensures that any malicious redaction, such as withholding part of his medical history, can be traced back to him.

6.2 Experiments

To evaluate the effectiveness of our RSS-IA, we first implement the transparent RSS for set-data presented by Johnson *et al.* [3]. Then we implement the proposed generic transformation to convert the transparent RSS [3] into an RSS-IA with a single designated redactor. The experiment algorithms are coded using the Miracl Library (https://github.com/miracl/MIRACL), and the resulting software is compiled using Visual Studio 2017. The tests are performed on a Lenovo PC with Intel(R) Core(TM) i5-7500 CPU @3.40 GHz, 8.00 GiB RAM and Windows 10 @64 bits.

In the experiments, the original data $\mathcal{M}$ and the redacted data $\mathcal{M}'$ are respectively shown in Tables 1 and 2. The signatures/evidence tags generated by the signer and the redactor are denoted as $\sigma_{\mathsf{S}}/\pi_{\mathsf{S}}$ and $\sigma_{\mathsf{R}}/\pi_{\mathsf{R}}$, respectively. The modulus of the RSA in [3] and the order p of group $\mathbb{Z}_p$ in our RSS-IA are set as 2048 bits. The accumulator in RSS [3] and the hash functions H_1 and H_2 in our RSS-IA are set as SHA512. The experimental results are shown in Table 3, where the times are all the average running time (in seconds) by running the schemes 100 times.

As we can see in Table 3, the implementation of our generic transformation which adds implicit-accountability to the transparent RSS [3] only costs extra 0.0323 s computation time in the Sign algorithm and 0.1221 s computation time in the Redact algorithm, respectively. As a result, the experimental results validate that our generic designs are effective, and are practical in achieving accountable and transparent authenticated data redaction.

7 Conclusion

We proposed a generic design of Redactable Signature Scheme with Implicit Accountability (RSS-IA), as a novel solution to authenticated data redaction. In our design, not only the data signer but also the redactor can generate an evidence tag as the proof of the generator of a data/signature pair. The redaction is accountable with the evidence tag. Without the evidence tag, the redaction operation is transparent. We formally defined the relevant security notions in order to capture the essence of the various security requirements, including resistance to collusion between the signer and redactor. Neither the original signer nor the redactor can generate an evidence tag to blame others. Even if the signer and the redactor collude with each other, they cannot both claim (or deny) themselves as the generator of a valid data/signature pair. We further extended our RSS-IA to the multi-redactor setting. The applications and experiments analyses show that our designs are effective and practical in achieving accountable and transparent authenticated data redaction.

TABLE 3
Computation Cost Comparison (in Seconds)

Schemes	Accountability	Sign	Verify (σ_S)	Redact	Verify (σ_R)	Proof$_S$ (σ_S)	Judge (π_S)	Proof$_R$ (σ_S)	Judge (π_R)
RSS [2]	×	0.1185	0.1355	0.0217	0.1131	Not Support			
Our RSS-IA	√	0.1508	0.1799	0.1438	0.1786	0.0504	0.0481	0.1211	0.1392

Differential privacy provides a mathematically rigorous mean to protect data privacy. It involves other types of data modification than the "delete operation". Our design only supports "delete operation" during data modification. As a result, it does not support differential privacy and other advanced privacy protection approaches. Data authentication supporting other kinds of data modification operations in differential privacy is our future research direction.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (61822202, 61872089, and 61872090).

References

[1] S. Goldwasser, S. Micali, and R. L. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM J. Comput.*, vol. 17, no. 2, pp. 281–308, 1988.

[2] C. Dwork, "Differential privacy: A survey of results," in *International Conference on Theory and Applications of Models of Computation*. Berlin, Germany: Springer, 2008, pp. 1–19.

[3] R. Johnson, D. Molnar, D. Song, and D. Wagner, "Homomorphic signature schemes," in *Cryptographers' Track at the RSA Conference*. Berlin, Germany: Springer, 2002, pp. 244–262.

[4] R. Steinfeld, L. Bull, and Y. Zheng, "Content extraction signatures," in *International Conference on Information Security and Cryptology*. Berlin, Germany: Springer, 2001, pp. 285–304.

[5] K. Miyazaki, M. Iwamura, T. Matsumoto, R. Sasaki, H. Yoshiura, and S. Tezuka, "Digitally signed document sanitizing scheme with disclosure condition control," *IEICE Trans. Fundamentals Electron. Commun. Comput. Sci.*, vol. 88, no. 1, pp. 239–246, 2005.

[6] K. Miyazaki, G. Hanaoka, and H. Imai, "Digitally signed document sanitizing scheme based on bilinear maps," in *Proc. ACM Symp. Inf. Comput. Commun. Secur.*, 2006, pp. 343–354.

[7] J. Ma, J. Liu, M. Wang, and W. Wu, "An efficient and secure design of redactable signature scheme with redaction condition control," in *Proc. Int. Conf. Green Pervasive Cloud Comput.*, 2017, pp. 38–52.

[8] L. Bull, D. M. Squire, J. Newmarch, and Y. Zheng, "Grouping verifiable content for selective disclosure," in *Proc. Australas. Conf. Inf. Secur. Privacy*, 2003, pp. 1–12.

[9] L. Bull, D. McG. Squire, and Y. Zheng, "A hierarchical extraction policy for content extraction signatures," *Int. J. Digital Libraries*, vol. 4, no. 3, pp. 208–222, 2004.

[10] J. Ma, J. Liu, X. Huang, Y. Xiang, and W. Wu, "Authenticated data redaction with fine-grained control," *IEEE Trans. Emerg. Topics Comput.*, online version, doi: 10.1109/TETC.2017.2754646.

[11] J. Liu, J. Ma, W. Zhou, Y. Xiang, and X. Huang, "Dissemination of authenticated tree-structured data with privacy protection and fine-grained control in outsourced databases," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2018, pp. 167–186.

[12] J. Liu, J. Ma, Y. Xiang, W. Zhou, and X. Huang, "Authenticated medical documents releasing with privacy protection and release control," *IEEE Trans. Dependable Secure Comput.*, online version, doi: 10.1109/TDSC.2019.2892446.

[13] K. Samelin, H. C. Pöhls, A. Bilzhause, J. Posegga, and H. De Meer, "Redactable signatures for independent removal of structure and content," in *Proc. Int. Conf. Inf. Secur. Practice Experience*, 2012, pp. 17–33.

[14] H. de Meer, H. C. Pöhls, J. Posegga, and K. Samelin, "Redactable signature schemes for trees with signer-controlled non-leaf-redactions," in *Proc. Int. Conf. E-Business Telecommun.*, 2012, pp. 155–171.

[15] K. Samelin, H. C. Pöhls, A. Bilzhause, J. Posegga, and H. de Meer, "On structural signatures for tree data structures," in *Proc. Int. Conf. Appl. Cryptography Netw. Secur.*, 2012, pp. 171–187.

[16] H. De Meer, H. C. Pöhls, J. Posegga, and K. Samelin, "On the relation between redactable and sanitizable signature schemes," in *Proc. Int. Symp. Eng. Secure Softw. Syst.*, 2014, pp. 113–130.

[17] H. C. Pöhls and K. Samelin, "Accountable redactable signatures," in *Proc. 10th Int. Conf. Availability Rel. Secur.*, 2015, pp. 60–69.

[18] C. Brzuska *et al.*, "Security of sanitizable signatures revisited," in *Proc. Int. Workshop Public Key Cryptogr.*, 2009, pp. 317–336.

[19] C. Brzuska, H. C. Pöhls, and K. Samelin, "Non-interactive public accountability for sanitizable signatures," in *Proc. Eur. Public Key Infrastructure Workshop*, 2012, pp. 178–193.

[20] J. J. Rotman, *An Introduction to the Theory of Groups*, vol. 148, Berlin, Germany: Springer, 2012.

[21] D. Derler, H. C. Pöhls, K. Samelin, and D. Slamanig, "A general framework for redactable signatures and new constructions," in *Proc. Inf. Secur. Cryptology*, 2015, pp. 3–19.

[22] J. Camenisch and M. Stadler, "Proof systems for general statements about discrete logarithms," PhD Thesis, ETH Zürich, Zurich. Switzerland, 1997.

[23] W. Ogata, K. Kurosawa, and S.-H. Heng, "The security of the FDH variant of chaum's undeniable signature scheme," in *Proc. Int. Workshop Public Key Cryptogr.*, 2005, pp. 328–345.

[24] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures," *J. Cryptol.*, vol. 13, no. 3, pp. 361–396, 2000.

Jinhua Ma received the MS degree from the School of Mathematics and Informatics, Fujian Normal University, China, in 2016. She is currently working toward the PhD degree with the School of Mathematics and Informatics, Fujian Normal University, China. Her research interests include cryptography and information security.

Xinyi Huang (Member, IEEE) received the PhD degree from the School of Computer Science and Software Engineering, University of Wollongong, Australia, in 2009. He is currently a professor with the School of Mathematics and Informatics, Fujian Normal University, China. His research interests include cryptography and information security. He has published more than 160 research papers in refereed international conferences and journals, such as ACM CCS, the *IEEE Transactions on Computers*, *IEEE Transactions on Parallel and Distributed Systems*, and *IEEE Transactions on Information Security and Forensics*. His work has been cited more than 6200 times at Google Scholar. He is in the editorial board of *International Journal of Information Security* and *Science China Information Sciences*. He has served as the program/general chair or program committee member in more than 120 international conferences.

Yi Mu (Senior Member, IEEE) received the PhD degree from Australian National University, in 1994. He is currently a full professor with Fujian Normal University, China. Prior to his position at Fujian Normal University, he was a full professor wit the University of Wollongong (UOW), Australia. He also served as the head of School of Computer Science and Software Engineering at UOW during 2011 and 2015. His research Interests include cryptography and cyber security.

Robert H. Deng (Fellow, IEEE) has been an AXA chair professor of cybersecurity and a professor of information systems with the School of Information Systems, Singapore Management University, Singapore, since 2004. His current research interests include data security and privacy, multimedia security, and network and system security. He was a recipient of the Distinguished Paper Award from NDSS, in 2012, the Best Paper Award from CMS, in 2012, and the Best Journal Paper Award from IEEE Communications Society, in 2017. He has served on the Editorial Boards of many international journals, including the *IEEE Transactions on Information Forensics and Security* and the *IEEE Transactions on Dependable and Secure Computing*.