This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2018.2877601, IEEE Transactions on Dependable and Secure Computing

Accountable Proxy Re-Encryption for Secure Data Sharing

Hui Guo, Zhenfeng Zhang, Jing Xu, Ningyu An, Xiao Lan

***Abstract*—Proxy re-encryption (PRE) provides a promising solution for encrypted data sharing in public cloud. When data owner Alice is going to share her encrypted data with data consumer Bob, Alice generates a re-encryption key and sends it to the cloud server (proxy); by using it, the proxy can transform Alice's ciphertexts into Bob's without learning anything about the underlying plaintexts. Despite that existing PRE schemes can prevent the proxy from recovering Alice's secret key by collusion attacks with Bob, due to the inherent functionality of PRE, it is inevitable that the proxy and Bob together are capable to gain and distribute Alices decryption capabilities. Even worse, the malicious proxy can deny that it has leaked the decryption capabilities and has very little risk of getting caught.**

To tackle this problem, we introduce the concept of *Accountable Proxy Re-Encryption* (APRE), whereby if the proxy is accused to abuse the re-encryption key for distributing Alice's decryption capability, a judge algorithm can decide whether it is innocent or not. We then present a non-interactive APRE scheme and prove its CPA security and accountability under DBDH assumption in the standard model. Finally, we show how to extend it to a CCA secure one.

I. Introduction

Cloud storage and data sharing have rapidly gained a central role in the digital society, serving as a building block of consumer-oriented applications such as Amazon S3 [1], iCloud, Dropbox, Microsoft SkyDrive, and Google Drive [2]. Moreover, more and more personal record management systems also rely on cloud platform collecting, storing and sharing information. For instance, personal health record (PHR) services are outsourced to or provided by third-party cloud service providers such as Microsoft HealthVault, PatientsLikeMe and ELGA, which makes the storage and sharing of the medical information more efficient and eases data synchronization across different hospitals.

Despite of its convenience and popularity, the cloud service poses a number of data security issues such as privacy and integrity, which has been the major concerns for users utilizing such services. The conventional solution is to encrypt user's data before uploading it to the cloud. Such a setting, however, brings the difficulty for sharing data among different users.

H. Guo is with the State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China. E-mail:guohtech@foxmail.com.

Z. Zhang and J. Xu are the corresponding authors, with Trusted Computing and Information Assurance Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China. E-mails:{zfzhang, xujing}@tca.iscas.ac.cn.

N. An is with the State Grid Corporation Joint Laboratory of Artificial Intelligence on Electric Power System, Global Energy Interconnection Research Institute co.,Ltd., Beijing 102209, China. E-mail: anningyu@geiri.sgcc.com.cn.

X. Lan is with Cybersecurity Research Institute, Sichuan University, Chengdu 610064, China. E-mail:lanxiao@scu.edu.cn.

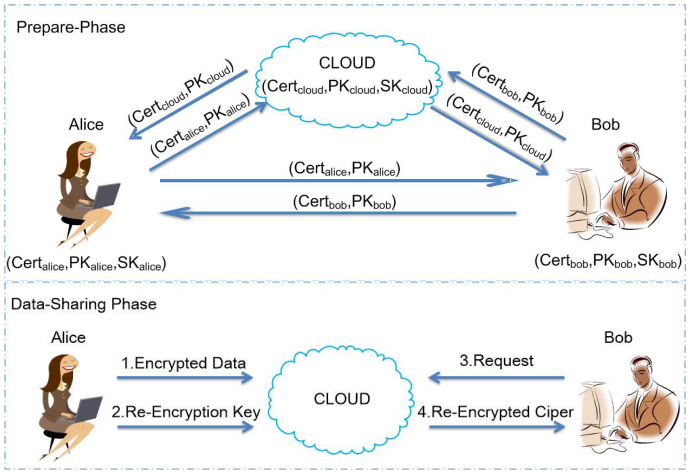

Fig. 1: PRE for Cloud Data Sharing

Obviously, the data owner can first download the ciphertext and decrypt it using his own secret key, and then encrypt it to different receivers respectively. Nonetheless, it is impractical since those operations greatly increase the data owner's computation and communication costs. Furthermore, this approach also suffers from a limitation that the data owner has to be online all the time.

In order to tackle the dilemma of data sharing, proxy re-encryption (PRE) was proposed by Blaze et al. [3] in 1998. In a PRE scheme, a proxy with specific information (re-encryption key) can transform a ciphertext intended for Alice (delegator) into another ciphertext that can be decrypted by Bob (delegatee). Besides cloud data sharing [4] [5] [6] [7] [8], PRE has many other practical applications such as email forwarding [9], distributed files systems [10] [11] [12], digital rights management [13] and publish-subscribe systems [14] [15].

Fig.1 illustrates typical PRE application in cloud data sharing. Alice, an acquisitor and provider of sensitive data, might wish to store the encrypted data on the cloud server and forward it to her paid users. Note that Alice has the data ownership and she does not hope that anyone including the cloud server could access the data without her authorization. In prepare phase, Alice, Bob and proxy transmit their certificates and public keys to each other for authentication. In data-sharing phase, Alice encrypts data and stores the ciphertexts on the cloud server. When Alice is going to share her encrypted data with a data consumer Bob, Alice generates a re-encryption key and sends it to the cloud server. On Bob's request, the cloud sever transforms the ciphertexts of Alice

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2018.2877601, IEEE Transactions on Dependable and Secure Computing

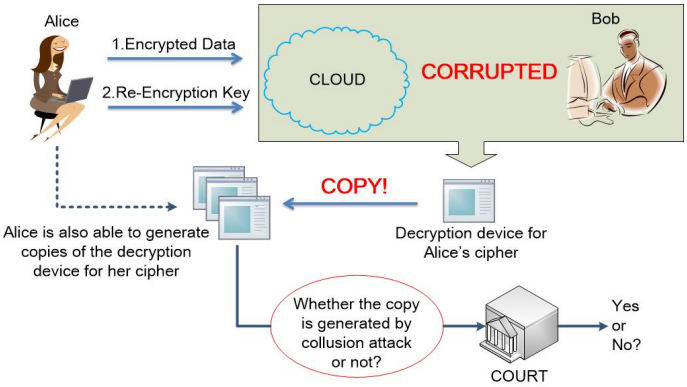

Fig. 2: Collusion Attack and Accountability

and forwards them to Bob.

The traditional security notion of PRE focuses on preventing the proxy from learning anything about the encrypted messages. However, it is not enough to realize the application requirement shown in the above example. Due to the inherent functionality of PRE, the cloud server and Bob together are able to obtain Alice's decryption capability and keep it on any form of carriers, such as a decryption program or a decryption device[1]. Therefore, Alice's decryption capability could be *sold* online and offline, which makes Alice suffering serious economic losses. This weakness is also called re-encryption key abuse problem.

If we regard a decrypted message as a *fish*, a decryption device can be used to *fish*: distributing an illegal decryption device is much more dangerous than a single message! Much worse, the malicious server does not have any risk of being caught in a court of law. Specifically, the decryption device itself cannot be used as a conclusive proof about who is guilty since Alice also has the decryption capability (see Fig.2).

To mitigate the above re-encryption key abuse problem, Ateniese et al. [10] introduced the notion of non-transferability in 2005. Non-transferability can be seen as a tradeoff solution to protect Alice's "benefit" in delegating her decryption right: when Bob and a proxy collude to transfer Alice's decryption capability, as a cost, Bob has to expose his own decryption capability. Constructing a non-transferable PRE scheme has long been an open problem until Guo et al. [16] gave a generic construction using indistinguishability obfuscation [17] and k-unforgeable authentication [18] as main tools.

Non-transferability provides a proactive deterrence of malicious users, and indeed it is an effective solution addressing the re-encryption key abuse problem. However, it is still insufficient for the above specific cloud data sharing situation. As an ordinary data consumer, Bob's secret key is usually less valuable than the data provider's. Thus, Bob might be willing to join collusion attacks, generating and selling decryption devices (or perhaps, Bob is a dummy user "born" for collusion attacks). Meanwhile, the proxy losses nothing.

[1]In this paper, a decryption capability's carrier is uniformly called a decryption device.

A. Our Contribution

In this paper, we explore a new approach to resolve the above trust problem. In particular, we introduce the concept of accountable PRE (APRE), where a convincing proof can be provided from which a judge can make a decision about who is guilty. Therefore, if the proxy (colluded with any delegatee) re-distributes a decryption device, it will have the risk of being caught and sued.

First, we put forth the formal model for APRE scheme. Accountability suggests there is a judge algorithm, which is able to tell unequivocally the creator of the device when given black-box access to a decryption device. An APRE scheme is deemed secure if it satisfies the following three requirements. (i) CPA/CCA security. It must satisfy the standard security notion for PRE schemes. (ii) Security against malicious proxy[2]. The malicious proxy and any delegatee cannot create a decryption device which makes the judge algorithm decide the delegator is malicious. (iii) Security against malicious delegator. It is infeasible for the malicious delegator to create a decryption device such that the judge algorithm implicates the proxy is malicious.

Then, we present the first APRE scheme based on the PRE scheme recently proposed by Guo et al. [19]. We prove that our scheme is CPA secure, secure against malicious proxy and secure against malicious delegator under the DBDH assumption in the standard model. Furthermore, we extend it to a CCA secure one by employing a generic transformation [20]. It is also worth mentioning that our constructions have the following features:

- Public accountability. The judge algorithm is public, that is, anyone can run the judge algorithm with no additional secret needed.
- Black-box accountability. Given an illegal decryption device, just by observing the input/output behavior of the decryption device, the judge algorithm could find out the one who create the decryption device.
- Non-interactiveness. The re-encryption key generation process is non-interactive and thus communication cost can be saved.

B. Our Techniques

An accountable PRE scheme does not prevent re-encryption key abuse attacks but provides a disincentive to the malicious party. With regard to the construction, our main idea is to let the proxy (with any delegatee) and the delegator have different "decryption capabilities". That is, for some irregular ciphertexts, only one of them can get the right decryption results. Thus, by feeding a given decryption device with such ciphertexts and observing its outputs we are able to decide which one, the proxy or the delegator, is malicious.

[2]In this paper, we only consider punishment on malicious-proxy. First, a collusion attack often involves many malicious delegatees, so it is difficult to punish all the malicious delegatees. Moreover, as long as the delegator is not malicious, a collusion attack cannot be launched no matter how many delegatees are malicious. Second, cloud service providers such as iCloud, Dropbox, and Google Drive etc., act as the role of proxy in practice. Accountability is a mechanism which deters malicious proxy by destroying their reputations in addition to providing evidence to court.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2018.2877601, IEEE Transactions on Dependable and Secure Computing

In order to achieve this property, both the re-encryption key and ciphertexts should be well designed.

Re-encryption key. For the generation of re-encryption key, the main technical difficulty is how to add "some salt" so that the decryption capability of the proxy (with any delegatee) can be distinguished from the delegator. In more detail, we should consider two aspects in constructing the re-encryption key. First, the salt in the re-encryption key should be integrated with the re-encryption key, namely, the proxy can not remove it from the re-encryption key; otherwise, a malicious proxy is able to have the same decryption capability as the delegator. Second, the re-encryption key should not reveal the salt, namely, only the proxy is allowed to have knowledge of the salt; otherwise, a malicious delegator obtaining the salt is able to simulate the proxy.

In this paper, we design the re-encryption key as a signature [21] on the proxy's public key. Informally, the secret key of the proxy plays the role of salt in the above analysis. Hence, the unforgeability of the underlying signature implies the proxy can not forge another re-encryption key to remove its key pair's information. Simultaneously, the delegator has no knowledge about the secret key of the proxy. As a result, a decryption device created by using the re-encryption key can be distinguished from the one generated by the delegator.

Ciphertexts. For the generation of ciphertexts (mainly the original ciphertexts of the delegator), the main technical difficulty is resolving the tension between the privacy of ciphertexts and the success of the judge procedure. That is, on one hand we require that during regular operation, the outcome of decrypting a ciphertext should not leak any information about the decryption key. On the other hand, during judging, it should be able to extract enough information about the decryption key of a given decryption device, in order to distinguish the device's creator.

In more detail, we take the following two aspects into consideration. Firstly and the most basically, the ciphertexts should fit the re-encryption key well to achieve the re-encryption functionality. Since the re-encryption key includes the public key of the proxy, the ciphertexts should enable that the re-encryption algorithm executes "encryption and decryption under the proxy's public key" simultaneously. As a result, the re-encrypted ciphertext contains no information of proxy. In particular, we design the ciphertext of the CPA secure construction to include four elements, two of which guarantee the delegator can decrypt normally, one of which is for removing the proxy's information of the re-encryption key in the re-encryption process, and the last of which helps the delegatee decrypt the re-encrypted ciphertext. Secondly, in order to make the judge algorithm tell the creator of a given decryption device, there should be an irregular form of ciphertexts which can only be decrypted correctly by the the proxy (with any delegatee). Moreover, the irregular form should not be distinguished by the proxy (with any delegatee). Namely, if a decryption device is created by the malicious proxy and any delegatee, it cannot verify a given ciphertext's validity.

We construct a judge algorithm using ideas from accountable authority identity-based encryption [22]. By observing the re-encryption and decryption algorithms, we find if we simultaneously modify two elements of the original ciphertext by multiplying a random value and its inverse, the decryption result remain unchanged. But the delegator cannot obtain any useful information from such an irregular ciphertext. Therefore, given a decryption device, we can decide who is the creator by feeding it with the irregular ciphertexts and observing the outputs. In other words, if the outputs are correct with a non-negligible probability, the device is generated by the proxy, and the delegator otherwise.

See more details in Section IV.

C. Related Work

Non-transferable PRE. Since Blaze et al. [3] first proposed the concept of PRE, there have been lots of work enhancing the ciphertexts security, such as CPA secure PREs [23] [24] [25] [26] and CCA secure PREs [9] [27] [28] [29] [30]. In addition, many PRE schemes with special security properties are proposed, such as type-based (conditional) PREs [31] [32] [33] [12], forward secure PRE [34] and PRE for revocation and key rotation [35]. All of the above schemes are assumed the proxy is semi-honest, and thus the re-encryption key abuse problem of PRE scheme cannot be captured. Ateniese et al. [10] first considered this problem and introduced the notion of non-transferability. They left an open problem on how to construct a non-transferable PRE scheme. Afterwards, there has been several works aiming at resolving this problem. Libert and Vergnaud [36] proposed a traceable proxy re-encryption scheme, where malicious proxy revealing the re-encryption key to the third party can be identified by the delegator. Their work assumes that the delegator is honest and the revealed re-encryption key cannot be leaked by the delegator himself. In this paper, instead of such assumption, the delegator may be malicious and thus the goal is to find malicious delegator or malicious proxy. Later, Hayashi et al. [37] and Guo et al. [19] attempted to give relaxed notions of non-transferability. Unfortunately, their security model could not capture all attacks of decryption rights' transference. Moreover, Isshiki et al. [38] pointed out that Hayashi et al.'s scheme is vulnerable to the forgeability attack of re-encryption keys and the security assumption employed in their proofs can be solved efficiently. Recently, Guo et al. [16] formalized the notion of non-transferability and proposed a concrete construction building on two primitives: an indistinguishability obfuscator for circuits and a k-unforgeable authentication scheme. The transferability issue in proxy re-encryption has also been considered in [39] [40] [41], but their work lacks formal security model and security proof.

As discussed above, non-transferable PRE aims at providing a proactive way of deterrence while accountable PRE provides a "detect-then-punish" way of deterrence. At first glance, non-transferability seems to be a stronger security notion than accountability. However, non-transferability only works in the case that delegatee's secret key is more valuable than delegator's. Otherwise, at a lower price, a delegatee might be willing to join collusion attacks with the proxy to create and sell illegal decryption devices of delegator, and moreover, the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2018.2877601, IEEE Transactions on Dependable and Secure Computing

malicious proxy loses nothing. On the contrary, in accountable PRE, the proxy can be detected and punished as long as collusion attacks are launched.

Accountable IBE. Identity based encryption (IBE) was first introduced by Shamir [42] in 1984 and not realized until 2001 by Boneh-Franklin [43] and Cock [44]. It is an approach to avoid certificate management of traditional public key infrastructure. In spite of its advantages, IBE suffers from the key escrow problem: a private key generator (PKG) can not only decrypt all the ciphertexts but also distribute decryption keys arbitrarily without being caught. In order to reduce trust in the PKG, Goyal [45] introduced the notion of accountable IBE in 2007. Following this work, several papers [46] [47] [48] [22] [49] made efforts to achieve stronger security notions of accountable IBE.

Our work has similarities with accountable IBE. Informally, accountable IBE aims at solving the key abuse problem of the PKG, and in this paper we try to address the key abuse issue of the proxy. But, actually, the PKG is considered to be a trusted party unconditionally in most identity based cryptosystems, while the proxy is only a semi-honest party in PRE. Hence, it is of great practical significance to restrict misbehavior of the proxy.

II. Preliminaries

Recall the definition of bilinear groups [43] [50]. Let $\mathbb{G}$ and $\mathbb{G}_T$ be two cyclic groups of prime order $p \geq 2^\lambda$ and g be the generator of the group $\mathbb{G}$, where λ is the security parameter. A map e: $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is said to be a bilinear map if it satisfies the following conditions:

1) for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2) e is non-degenerate (i.e. if $\mathbb{G} = \langle g \rangle$, then $\mathbb{G}_T = \langle e(g, g) \rangle$).
3) e is efficiently computable.

Let $\mathbb{G}$, $\mathbb{G}_T$ and e be bilinear groups defined above, we then recall the hardness assumptions required in our scheme:

DBDH Assumption. For an algorithm $\mathcal{B}$, define its advantage as

$$\mathrm{Adv}_{\mathcal{B}}^{\mathrm{DBDH}}(\lambda) = |\Pr[\mathcal{B}(g, g^a, g^b, g^c, e(g, g)^{abc}) = 1] - \Pr[\mathcal{B}(g, g^a, g^b, g^c, e(g, g)^t) = 1]|$$

where $a, b, c, t \leftarrow \mathbb{Z}_p$ are randomly chosen. We say that the DBDH (Decisional Bilinear Diffie-Hellman) assumption holds, if for any probabilistic polynomial time (PPT) algorithm $\mathcal{B}$, its advantage $\mathrm{Adv}_{\mathcal{B}}^{\mathrm{DBDH}}(\lambda)$ is negligible in λ.

III. Definition and Security Model

In our paper, we use λ to denote the security parameter. A PRE scheme is a tuple of algorithms [10] as follows.

- $\mathrm{Setup}(\lambda)$. Taking the security parameter λ as input, this algorithm outputs the public parameter param which specifies the plaintext space M, the ciphertext space C and the randomness space R.
- $\mathrm{ProxySet}(\mathsf{param})$. This algorithm outputs the proxy's public key and secret key pair $(\mathsf{pk_p}, \mathsf{sk_p})$.
- $\mathrm{KeyGen}(\mathsf{param})$. This algorithm outputs the user's public key and secret key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$. We omit param in the following algorithm's input.
- $\mathrm{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j, \mathsf{pk_p})$. Taking a delegator's secret key sk_i, a delegatee's public key pk_j and the proxy's public key $\mathsf{pk_p}$ as input, this algorithm outputs a re-encryption key $\mathsf{rk}_{i \rightarrow j}$.
- $\mathrm{Enc}_1(\mathsf{pk}_j, m)$. Taking a user's public key pk_j and a message $m \in \mathsf{M}$ as input, this algorithm outputs a ciphertext C'_j. It is a first level ciphertext and cannot be re-encrypted for another user.
- $\mathrm{Enc}_2(\mathsf{pk}_i, m)$. Taking a user's public key pk_i and a message $m \in \mathsf{M}$ as input, this algorithm outputs a ciphertext C_i. It is a second level ciphertext and can be re-encrypted for another user.
- $\mathrm{ReEnc}(\mathsf{rk}_{i \rightarrow j}, \mathsf{sk_p}, C_i)$. Taking the re-encryption key $\mathsf{rk}_{i \rightarrow j}$, the proxy's secret key and a second level ciphertext C_i as input, this algorithm outputs a re-encrypted ciphertext C'_j.
- $\mathrm{Dec}_1(\mathsf{sk}_j, C'_j)$. Taking the secret key sk_j and a first level ciphertext or a re-encrypted ciphertext C'_j as input, this algorithm outputs a message $m \in \mathsf{M}$.
- $\mathrm{Dec}_2(\mathsf{sk}_i, C_i)$. Taking the secret key sk_i and a second level ciphertext C_i as input, this algorithm outputs a message $m \in \mathsf{M}$.
- $\mathrm{Judge}^{D_{i,\mu}}(\mathsf{pk}_i, \mathsf{pk_p})$. With black-box access to a μ-useful decryption device $D_{i,\mu}$[3] and taking the pubic key pk_i and $\mathsf{pk_p}$ as input, this algorithm outputs Proxy or Delegator, indicating the one who generated the decryption device.

Correctness. For any proxy key pair $(\mathsf{pk_p}, \mathsf{sk_p}) \leftarrow \mathrm{ProxySet}(\mathsf{param})$, any users' key pairs $(\mathsf{pk}_i, \mathsf{sk}_i)$, $(\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathrm{KeyGen}(\mathsf{param})$, any re-encryption key $\mathsf{rk}_{i \rightarrow j} \leftarrow \mathrm{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j, \mathsf{pk_p})$ and any message $m \in \mathsf{M}$, the following conditions hold:

$$\mathrm{Dec}_1(\mathsf{sk}_j, \mathrm{Enc}_1(\mathsf{pk}_j, m)) = m;$$
$$\mathrm{Dec}_2(\mathsf{sk}_i, \mathrm{Enc}_2(\mathsf{pk}_i, m)) = m;$$
$$\mathrm{Dec}_1(\mathsf{sk}_j, \mathrm{ReEnc}(\mathsf{rk}_{i \rightarrow j}, \mathsf{sk_p}, \mathrm{Enc}_2(\mathsf{pk}_i, m))) = m.$$

Remark 1. In our definition, the proxy is associated with a public key (certificate), which makes it possible that the proxy embeds its "secret" in the re-encryption key while the re-encryption key is generated non-interactively. Besides, the public key (certificate) is inevitable in practice. On one hand, it is necessary for users to authenticate the proxy's legitimacy. On the other hand, it is essential for realizing the non-repudiation property of malicious proxy.

Remark 2. In our definition, the judge algorithm is public, which means the judge algorithm can be run by anyone without secret input. The judge algorithm is also black-box, which means given an illegal decryption device, it can find out the one who create the decryption device just by observing the input/output behavior of the decryption device. Therefore, CheckKey algorithm [36] used to define the re-encryption key in white-box trace algorithm is not needed.

[3]For non-negligible probability value μ, a PPT algorithm $D_{i,\mu}$ is a μ-useful decryption device for user i, if $\Pr[m \leftarrow \mathsf{M}, C_i \leftarrow \mathsf{Enc_2}(\mathsf{pk}_i, m), m' \leftarrow D_{i,\mu}(C_i) : m = m'] \geq \mu$, where M is the plaintext space.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2018.2877601, IEEE Transactions on Dependable and Secure Computing

A. Security Model

In this subsection, we first recall the CPA security. Then we propose the security definition of accountability.

We adopt the Knowledge of Secret Key (KOSK) model, where all entities including the proxy should prove knowledge of their secret keys before registering public keys. This assumption implies that every entity should provide certification authorities (CAs) with a proof of knowledge of his private key. We also assume a static model where adversaries are not allowed to adaptively corrupt users as adopted in [9] [27]. An adversary has access to the following oracles:

- Proxy key reveal oracle $\mathcal{O}_{\text{pyr}}(\lambda)$: Return the proxy's secret key $\mathsf{sk_p}$.
- Uncorrupted key generation oracle $\mathcal{O}_{\text{hkg}}(i)$: Compute $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \text{KeyGen}(i)$, return pk_i.
- Corrupted key generation oracle $\mathcal{O}_{\text{ckg}}(i)$: Compute $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \text{KeyGen}(i)$, return $(\mathsf{sk}_i, \mathsf{pk}_i)$.
- Re-encryption key generation oracle $\mathcal{O}_{\text{rkg}}(\mathsf{pk}_i, \mathsf{pk}_j)$: On input of $(\mathsf{pk}_i, \mathsf{pk}_j)$, where $\mathsf{pk}_i, \mathsf{pk}_j$ were generated before by KeyGen, return a re-encryption key $\mathsf{rk}_{i\to j} \leftarrow \text{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j, \mathsf{pk_p})$.

1) CPA Security: To capture the CPA security notion for PRE schemes, we associate a CPA adversary $\mathcal{A}$ with the following template security experiment:

Experiment $\text{Exp}^{\text{cpa-}\delta}_{\Pi,\mathcal{A}}(\lambda)$
  $\mathsf{param} \leftarrow \text{Setup}(\lambda)$;
  $\mathsf{pk_p}, \mathsf{sk_p} \leftarrow \text{ProxySet}(\mathsf{param})$;
  $(\mathsf{pk}^*, m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}'}(\mathsf{param}, \mathsf{pk_p})$;
  $d^* \leftarrow \{0,1\}$;
  $C^* = \text{Enc}_\delta(\mathsf{pk}^*, m_{d^*})$;
  $d' \leftarrow \mathcal{A}^{\mathcal{O}'}(\mathsf{param}, C^*)$;
  If $d' = d^*$ return 1;
  else return 0.

In the above experiment, $\delta \in \{1,2\}$ specifies which level ciphertext that $\mathcal{A}$ attacks and $\mathcal{O}' = \{\mathcal{O}_{\text{pyr}}, \mathcal{O}_{\text{hkg}}, \mathcal{O}_{\text{ckg}}, \mathcal{O}_{\text{rkg}}\}$. The advantage of $\mathcal{A}$ is defined as

$$\text{Adv}^{\text{cpa-}\delta}_{\Pi,\mathcal{A}}(\lambda) = |\Pr[\text{Exp}^{\text{cpa-}\delta}_{\Pi,\mathcal{A}}(\lambda) = 1] - \frac{1}{2}|.$$

Formally, we present the CPA security as follows.

Definition 1 *(CPA Security at the Second Level [19].)* For any PRE scheme Π_{s}, we instantiate the experiment with a CPA adversary $\mathcal{A}$ and $\delta = 2$. It is required that pk^* is uncorrupted and $|m_0| = |m_1|$. If C^* denotes the challenge ciphertext, $\mathcal{A}$ can never make re-encryption key generation query $\mathcal{O}_{\text{rkg}}(\mathsf{pk}^*, \mathsf{pk}_j)$, where pk_j is corrupted. Π_{s} is said to be secure against chosen plaintext attacks at the second level ciphertext if for any polynomial time adversary $\mathcal{A}$, the advantage function $\text{Adv}^{\text{cpa-2}}_{\Pi_{\text{s}},\mathcal{A}}(\lambda)$ is negligible in λ.

Definition 2 *(CPA Security at the First Level [19].)* For any PRE scheme Π_{s}, we instantiate the experiment with a CPA adversary $\mathcal{A}$ and $\delta = 1$. It is required that pk^* is uncorrupted and $|m_0| = |m_1|$. Π_{s} is said to be secure against chosen plaintext attacks at the first level ciphertext if for any polynomial time adversary $\mathcal{A}$, the advantage function $\text{Adv}^{\text{cpa-1}}_{\Pi_{\text{s}},\mathcal{A}}(\lambda)$ is negligible in λ.

2) Accountability: Accountability is twofold: firstly, if the proxy is malicious, it should not launch collusion attacks with any delegatee to create a decryption device without being caught; secondly, if the proxy is honest, it should not be framed by a malicious delegator.

First, let's consider the security definition against the malicious proxy. In the security experiment, the adversary holding the proxy's secret key can query and obtain a polynomial number of users' secret keys and re-encryption keys. We say that the adversary is successful if, it outputs a decryption device which misleads a judge to believe the honest user is guilty. The experiment is defined as follows:

Experiment $\text{Exp}^{\text{mp}}_{\Pi,\mathcal{A}}(\lambda)$
  $\mathsf{param} \leftarrow \text{Setup}(\lambda)$;
  $\mathsf{pk_p}, \mathsf{sk_p} \leftarrow \text{ProxySet}(\mathsf{param})$;
  $\mathsf{pk}^*, D_{*,\mu} \leftarrow \mathcal{A}^{\mathcal{O}_{\text{hkg}}, \mathcal{O}_{\text{ckg}}, \mathcal{O}_{\text{rkg}}}(\mathsf{param}, \mathsf{pk_p}, \mathsf{sk_p})$
  where pk^* is generated by $\mathcal{O}_{\text{hkg}}$
  and μ is a non-negligible probability value;
  If $\text{Judge}^{D_{*,\mu}}(\mathsf{pk}^*, \mathsf{pk_p}) = \text{Delegator}$
  return 1;
  else return 0.

The advantage of $\mathcal{A}$ is defined as

$$\text{Adv}^{\text{mp}}_{\Pi,\mathcal{A}}(\lambda) = |\Pr[\text{Exp}^{\text{mp}}_{\Pi,\mathcal{A}}(\lambda) = 1]|.$$

Definition 3 *(Malicious-Proxy Security.)* A PRE scheme Π_{s} is said to be Malicious-Proxy secure if for any polynomial time adversary $\mathcal{A}$, the advantage function $\text{Adv}^{\text{mp}}_{\Pi_{\text{s}},\mathcal{A}}(\lambda)$ is negligible in λ.

Similarly, we consider the security definition against malicious delegator. In the security experiment, the adversary can query and obtain a polynomial number of users' secret keys and re-encryption keys, but he cannot get the secret key of proxy. We say that the adversary is successful if, it outputs a decryption device which misleads a judge to believe the honest proxy is guilty. To formulate this security definition, we define the experiment as follows:

Experiment $\text{Exp}^{\text{md}}_{\Pi,\mathcal{A}}(\lambda)$
  $\mathsf{param} \leftarrow \text{Setup}(\lambda)$;
  $\mathsf{pk_p}, \mathsf{sk_p} \leftarrow \text{ProxySet}(\mathsf{param})$;
  $\mathsf{pk}^*, D_{*,\mu} \leftarrow \mathcal{A}^{\mathcal{O}_{\text{hkg}}, \mathcal{O}_{\text{ckg}}, \mathcal{O}_{\text{rkg}}}(\mathsf{param}, \mathsf{pk_p})$
  where pk^* is generated by $\mathcal{O}_{\text{ckg}}$
  and μ is a non-negligible probability value;
  If $\text{Judge}^{D_{*,\mu}}(\mathsf{pk}^*, \mathsf{pk_p}) = \text{Proxy}$
  return 1;
  else return 0.

The advantage of $\mathcal{A}$ is defined as

$$\text{Adv}^{\text{md}}_{\Pi,\mathcal{A}}(\lambda) = |\Pr[\text{Exp}^{\text{md}}_{\Pi,\mathcal{A}}(\lambda) = 1]|.$$

Definition 4 *(Malicious-Delegator Security.)* A PRE scheme Π_{s} is said to be Malicious-Delegator secure if for any polynomial time adversary $\mathcal{A}$, the advantage function $\text{Adv}^{\text{md}}_{\Pi_{\text{s}},\mathcal{A}}(\lambda)$ is negligible in λ.

A PRE scheme is said to be accountable, if it is both Malicious-Proxy secure and Malicious-Delegator secure.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2018.2877601, IEEE Transactions on Dependable and Secure Computing

IV. Our Construction

In this section, we will briefly give the idea behind the scheme before presenting it.

With regard to re-encryption keys, intuitively, in order to judge the malicious proxy abusing re-encryption keys, some information related to the proxy should be tightly bound to the re-encryption keys. Otherwise, the proxy might be able to forge a re-encryption key irrelevant to itself, and then the judge algorithm would be failed to identify its misbehaviour. Furthermore, the proxy's information in the re-encryption key should be kept private against the delegator. Otherwise, an honest proxy might be treated wrongly since a malicious delegator is able to act as it by using its private information. Taking the above into consideration, we generate the re-encryption key similar to a signature on the public key of the proxy.

With regard to ciphertexts, in order to decide who is guilty only by blackbox access to the decryption device, the ciphertext should have an irregular form which can distinguish the creator of the device. In other words, given an irregular ciphertext, the delegator and the proxy (with a delegatee) should obtain different results. Thus, by feeding the decryption device with delicately generated ciphertext, a judge is capable to tell its creator by observing its outputs.

We denote the following PRE scheme by SI.

$\mathrm{Setup}(\lambda)$. Let λ be the security parameter, $\mathbb{G}$ and $\mathbb{G}_T$ be groups of prime order $p \geq 2^\lambda$, and e: $\mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map. Randomly choose $g_1, g_2, h_1, h_2 \leftarrow \mathbb{G}$ and compute $L = e(h_1, h_2)$. The system parameters are

$$\mathsf{param} = (g_1, g_2, h_1, h_2, L).$$

$\mathrm{ProxySet}(\mathsf{param})$. Choose $z \leftarrow \mathbb{Z}_p$ uniformly at random and compute $Z = g_2^z$. Set $\mathsf{sk_p} = z$ as the secret key and $\mathsf{pk_p} = Z$ as the public key.

$\mathrm{KeyGen}(\mathsf{param})$. Choose $x_i, y_i \leftarrow \mathbb{Z}_p$ uniformly at random and compute

$$(X_i, Y_i) = (h_1^{x_i}, g_1^{y_i}).$$

Set $\mathsf{sk}_i = (x_i, y_i)$ as the secret key and $\mathsf{pk}_i = (X_i, Y_i)$ as the public key.

$\mathrm{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j, \mathsf{pk_p})$. Given $\mathsf{sk}_i, \mathsf{pk}_j$ and $\mathsf{pk_p}$ as input, compute

$$W = (h_2 Y_j Z)^{1/x_i}.$$

Set $\mathsf{rk}_{i \to j} = W$ as the re-encryption key.

$\mathrm{Enc}_1(\mathsf{pk}_j, m)$. Given pk_j and m as input, choose $r \leftarrow \mathbb{Z}_p$ uniformly at random and compute

$$c_0' = L^r \cdot m, c_1' = g_1^r, c_2' = L^r \cdot e(h_1, Y_j)^r.$$

Set $C_j' = (c_0', c_1', c_2')$ as the first level ciphertext.

$\mathrm{Enc}_2(\mathsf{pk}_i, m)$. Given pk_i and m as input, choose $r \leftarrow \mathbb{Z}_p$ uniformly at random and compute

$$c_0 = L^r \cdot m, c_1 = g_1^r, c_2 = e(h_1, g_2)^r, c_3 = X_i^r.$$

Set $C_i = (c_0, c_1, c_2, c_3)$ as the second level ciphertext.

$\mathrm{ReEnc}(\mathsf{rk}_{i \to j}, \mathsf{sk_p}, C_i)$. Given $\mathsf{rk}_{i \to j}, \mathsf{sk_p}$ and C_i as input, compute

$$c_0' = c_0, c_1' = c_1, c_2' = e(c_3, W)/c_2^z.$$

Set $C_j' = (c_0', c_1', c_2')$ as the re-encrypted ciphertext.

$\mathrm{Dec}_1(\mathsf{sk}_j, C_j')$. Given sk_j and C_j' as input, compute the message

$$m = c_0' \cdot e(h_1, c_1')^{y_j} / c_2'.$$

$\mathrm{Dec}_2(\mathsf{sk}_i, C_i)$. Given sk_i and C_i as input, compute the message

$$m = c_0 / e(c_3, h_2)^{1/x_i}.$$

$\mathrm{Judge}^{D_{i,\mu}}(\mathsf{pk}_i, \mathsf{pk_p})$. Given the decryption device $D_{i,\mu}$ as an oracle and $\mathsf{pk}_i, \mathsf{pk_p}$ as input, this algorithm proceeds as follows.

1) Repeat the following experiment $n = \lambda/\mu$ times[4].
   - Choose $r, r' \in \mathbb{Z}_p$ and $m \in \mathbb{G}_T$ uniformly at random. Compute
   $$C = (c_0 = m \cdot L^r \cdot e(h_1, Z)^{r-r'}, c_1 = g_1^r,$$
   $$c_2 = e(h_1, g_2)^{r'}, c_3 = X_i^r).$$
   - Run the decryption device $D_{i,\mu}$ with C as input and obtain an output m'.
   - If $m' = m$, output "Proxy" and exit.
2) Otherwise, output "Delegator".

Correctness. SI satisfies the correctness property as follows.

- Decryption of a ciphertext $C_j' = (c_0', c_1', c_2')$ output by Enc_1 is correct, since
$$c_0' \cdot e(h_1, c_1')^{y_j} / c_2' = L^r \cdot m \cdot \frac{e(h_1, g_1^r)^{y_j}}{L^r \cdot e(h_1, g_1^r)^{y_j}} = m.$$
- Decryption of a ciphertext $C_i = (c_0, c_1, c_2, c_3)$ output by Enc_2 is correct, since
$$\begin{aligned} c_0 / e(c_3, h_2)^{1/x_i} &= L^r \cdot m / e(X_i^r, h_2)^{1/x_i} \\ &= e(h_1, h_2)^r \cdot m / e(h_1, h_2)^r \\ &= m. \end{aligned}$$
- Decryption of a ciphertext $C_j' = (c_0', c_1', c_2')$ output by ReEnc is correct, since the re-encrypted ciphertext and the first level ciphertext have the same form:
$$\begin{aligned} c_2' &= e(c_3, W)/c_2^z \\ &= e(X_i^r, (hY_jZ)^{1/x_i})/(e(h_1, g_2)^r)^z \\ &= L^r \cdot e(h_1, Y_j)^r. \end{aligned}$$
Thus, the decryption correctness of a first level ciphertext implies decryption correctness of a re-encrypted ciphertext.

Remark 3. In SI, if the delegatee or the proxy is allowed to choose public keys with unknown private keys, the proxy's secret may be removed from a re-encryption key (for example, the delegatee chooses his public key to be the inverse of the proxy's public key), which makes accountability impossible. That is one of the reasons why we adopt KOSK model.

V. Security Analysis Of Our Construction

In this section, we prove SI is CPA secure, Malicious-Delegator secure and Malicious-Proxy secure.

[4]We set $n = \lambda/\mu$ such that the probability of decryption device taking normal ciphertext as input and decrypting correctly at least once is overwhelming (since $(1-\mu)^n = (1-\mu)^{\lambda/\mu} \leq \frac{1}{e^\lambda}$, where $1-\mu$ is the probability of decryption error for a single run). More details can be found in security proofs of Lemma 4 and Lemma 6.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2018.2877601, IEEE Transactions on Dependable and Secure Computing

A. CPA Security

Theorem 1. SI is CPA secure at both the second level and the first level ciphertexts under the DBDH assumption.

Theorem 1 is obtained by combining the following Lemma 1 and Lemma 2, separately stating the CPA security at the second level ciphertext and the first level ciphertexts.

Lemma 1. SI is CPA secure at the second level ciphertext under the DBDH assumption.

Proof Let $\mathcal{A}$ be a CPA adversary attacking SI at level 2 with advantage ϵ. We construct an algorithm $\mathcal{B}$ to solve the DBDH problem by interacting with $\mathcal{A}$. The algorithm $\mathcal{B}$ takes a random challenge (g, g^a, g^b, g^c, T) as input, and $\mathcal{B}$'s goal is to decide whether $T = e(g,g)^{abc}$ or $T = e(g,g)^t$, where $a, b, c, t \leftarrow \mathbb{Z}_p$ are chosen uniformly at random.

$\mathsf{Setup}(\lambda)$. Choose $\delta \leftarrow \mathbb{Z}_p$ uniformly at random, set

$$g_1 = g, g_2 = g^\delta, h_1 = g^a, h_2 = g^b, L = e(g^a, g^b).$$

Return $\mathsf{param} = (g_1, g_2, h_1, h_2, L)$ as the system parameter.

$\mathsf{ProxySet}(\mathsf{param})$. Choose $z \leftarrow \mathbb{Z}_p$ uniformly at random and compute $Z = g_2^z$. Set the proxy's secret key as $\mathsf{sk_p} = z$ and public key as $\mathsf{pk_p} = Z$. Return $\mathsf{pk_p}$.

Phase 1. $\mathcal{B}$ answers $\mathcal{A}$'s queries as follows.

- $\mathcal{O}_{\text{pyr}}(\lambda)$. Return the proxy's secret key $\mathsf{sk_p} = z$.
- $\mathcal{O}_{\text{hkg}}(i)$. Choose $x_i, y_i \leftarrow \mathbb{Z}_p$ uniformly at random.
  - If it is the k-th key generation query (that $\mathcal{B}$ guesses it will be the target user), let $i^* = i$, compute
  $$\mathsf{pk}_{i^*} = (X_i = g^{x_{i^*}}, Y_i = g^{y_{i^*}}),$$
  implicitly setting $\mathsf{sk}_{i^*} = (x_{i^*}/a, y_{i^*})$. Return pk_{i^*}.
  - Otherwise, compute
  $$\mathsf{pk}_i = (X_i = h_1^{x_i}, Y_i = h_2^{-1} g^{y_i}),$$
  implicitly setting $\mathsf{sk}_i = (x_i, y_i - b)$. Return pk_i.
- $\mathcal{O}_{\text{ckg}}(i)$. Choose $x_i, y_i \leftarrow \mathbb{Z}_p$ uniformly at random and set $\mathsf{sk}_i = (x_i, y_i)$. Compute
  $$\mathsf{pk}_i = (X_i = h_1^{x_i}, Y_i = g^{y_i}).$$
  Return $(\mathsf{sk}_i, \mathsf{pk}_i)$.
- $\mathcal{O}_{\text{rkg}}(\mathsf{pk}_i, \mathsf{pk}_j, \mathsf{pk_p})$. Given $(\mathsf{pk}_i, \mathsf{pk}_j)$ where pk_i and pk_j were generated by $\mathcal{O}_{\text{hkg}}$ or $\mathcal{O}_{\text{ckg}}$, $\mathcal{B}$ distinguishes the following cases.
  - If $\mathsf{pk}_i \neq \mathsf{pk}_{i^*}$, run $\mathrm{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j, \mathsf{pk_p})$ and return whatever it outputs;
  - If $\mathsf{pk}_i = \mathsf{pk}_{i^*}$, pk_j is uncorrupted, compute
  $$W = (h_2 Y_j Z)^{(\frac{x_{i^*}}{a})^{-1}} = (g^a)^{\frac{y_j + z \cdot \delta}{x_{i^*}}}.$$
  Return $\mathsf{rk}_{i \to j} = W$.
  - If $\mathsf{pk}_i = \mathsf{pk}_{i^*}$, pk_j is corrupted, return $\perp$.

Challenge. When $\mathcal{A}$ decides to finish Phase 1, it outputs m_0, m_1 and a target public key pk^*. If $\mathsf{pk}_{i^*} \neq \mathsf{pk}^*$, $\mathcal{B}$ outputs a random bit and aborts. Otherwise, $\mathcal{B}$ picks a random coin $b \in \{0,1\}$ and computes

$$c_0 = T \cdot m_b, c_1 = g_1^r = g^c,$$

$$c_2 = e(h_1, g_2)^r = e(g^a, g^c)^\delta, c_3 = X_i^r = (g^c)^{x_{i^*}},$$

implicitly setting $r = c$. Suppose $\mathcal{A}$ makes $\mathcal{O}_{\text{hkg}}$ queries at most q_{hkg} times. Then, the probability that $\mathcal{B}$ guesses right i^* in Challenge phase is at least $1/q_{\text{hkg}}$.

Phase 2. $\mathcal{B}$ answers $\mathcal{A}$'s queries as Phase 1. Finally, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. If $b = b'$, $\mathcal{B}$ outputs 1, else outputs 0. The advantage that $\mathcal{B}$ solves the DBDH problem is at least $\frac{1}{q_{\text{hkg}}} \cdot \epsilon$, which completes the proof. □

Lemma 2. SI is CPA secure at the first level ciphertext under the DBDH assumption.

Proof Let $\mathcal{A}$ be a CPA adversary attacking SI at level 1 with advantage ϵ. We construct an algorithm $\mathcal{B}$ to solve the DBDH problem by interacting with $\mathcal{A}$. The algorithm $\mathcal{B}$ takes a random challenge (g, g^a, g^b, g^c, T) as input, and $\mathcal{B}$'s goal is to decide whether $T = e(g,g)^{abc}$ or $T = e(g,g)^t$, where $a, b, c, t \leftarrow \mathbb{Z}_p$ are chosen uniformly at random. The algorithm $\mathcal{B}$ proceeds as follows.

$\mathsf{Setup}(\lambda)$. Choose $\delta \leftarrow \mathbb{Z}_p$ uniformly at random. Set

$$g_1 = g, g_2 = g^\delta, h_1 = g^a, h_2 = g^b, L = e(g^a, g^b).$$

Return $\mathsf{param} = (g_1, g_2, h_1, h_2, L)$ as the system parameter.

$\mathsf{ProxySet}(\mathsf{param})$. Choose $z \leftarrow \mathbb{Z}_p$ uniformly at random and compute $Z = g_2^z$. Set $\mathsf{sk_p} = z$ and $\mathsf{pk_p} = Z$. Return the proxy's public key $\mathsf{pk_p}$.

Phase 1. $\mathcal{B}$ answers $\mathcal{A}$'s queries as follows.

- $\mathcal{O}_{\text{pyr}}(\lambda)$. Return the proxy's secret key $\mathsf{sk_p} = z$.
- $\mathcal{O}_{\text{hkg}}(i)$. Choose $x_i, y_i \leftarrow \mathbb{Z}_p$ uniformly at random.
  - If it is the k-th key generation query (that $\mathcal{B}$ guesses it will be the target user), let $i^* = i$, compute
  $$\mathsf{pk}_{i^*} = (X_i = h_1^{x_{i^*}}, Y_i = h_2^{-1} g^{y_{i^*}}),$$
  implicitly setting $\mathsf{sk}_{i^*} = (x_{i^*}, y_{i^*} - b)$. Return pk_{i^*}
  - Otherwise, compute
  $$\mathsf{pk}_i = (X_i = h_1^{x_i}, Y_i = g^{y_i}),$$
  implicitly setting $\mathsf{sk}_i = (x_i, y_i)$. Return pk_i.
- $\mathcal{O}_{\text{ckg}}(i)$. Choose $x_i, y_i \leftarrow \mathbb{Z}_p$ uniformly at random and set $\mathsf{sk}_i = (x_i, y_i)$. Compute
  $$\mathsf{pk}_i = (X_i = h_1^{x_i}, Y_i = g^{y_i}).$$
  Return $(\mathsf{sk}_i, \mathsf{pk}_i)$.
- $\mathcal{O}_{\text{rkg}}(\mathsf{pk}_i, \mathsf{pk}_j)$. Given $(\mathsf{pk}_i, \mathsf{pk}_j)$ where pk_i and pk_j were generated by $\mathcal{O}_{\text{hkg}}$ or $\mathcal{O}_{\text{ckg}}$, $\mathcal{B}$ runs $\mathrm{ReKeyGen}$ $(\mathsf{sk}_i, \mathsf{pk}_j)$ and return whatever it outputs.

Challenge. When $\mathcal{A}$ decides to finish Phase 1, it outputs m_0, m_1 and a target public key pk^*. If $\mathsf{pk}_{i^*} \neq \mathsf{pk}^*$, $\mathcal{B}$ outputs a random bit and aborts. Otherwise, $\mathcal{B}$ picks a random coin $b \in \{0,1\}$, and computes

$$c_0' = T \cdot m_b, c_1' = g_1^r = g^c,$$

$$c_2' = L^r \cdot e(h_1^r, Y_{i^*}) = e(g^a, g^c)^{y_{i^*}},$$

implicitly setting $r = c$. Suppose $\mathcal{A}$ makes $\mathcal{O}_{\text{hkg}}$ queries at most q_{hkg} times, then the probability that $\mathcal{B}$ guesses right i^* in Challenge phase is at least $1/q_{\text{hkg}}$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2018.2877601, IEEE Transactions on Dependable and Secure Computing

Phase 2. $\mathcal{B}$ answers $\mathcal{A}$'s queries as Phase 1.

Finally, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. If $b = b'$, $\mathcal{B}$ outputs 1, else outputs 0. The advantage that $\mathcal{B}$ solves the DBDH problem is at least $1/q_{\text{hkg}} \cdot \epsilon$, which completes the proof. □

B. Accountability

In order to prove SI is accountable, we prove it to be Malicious-Proxy secure and Malicious-Delegator secure, separately.

Theorem 2. SI is Malicious-Proxy secure under the DBDH assumption.

To prove the Malicious-Proxy security, we consider the following two hybrid experiments.

-*Hyb*$_0$ The original Malicious-Proxy security experiment. Note that when the adversary outputs a μ-useful decryption device $D_{*,\mu}$ for a target delegator, the challenger runs $D_{*,\mu}$ with n malformed ciphertexts. The malformed ciphertexts are generated as

$$c_0 = m \cdot L^r \cdot e(h_1, Z)^{r-r'}, c_1 = g_1^r,$$

$$c_2 = e(h_1, g_2)^{r'}, c_3 = X_i^r,$$

where $r, r' \leftarrow \mathbb{Z}_p, m \leftarrow \mathbb{G}_T$.

-*Hyb*$_1$ The same as Hyb$_0$ except that in the judge algorithm, the ciphertext is generated as

$$c_0 = m \cdot L^r, c_1 = g_1^r, c_2 = e(h_1, g_2)^r, c_3 = X_i^r,$$

where $r \leftarrow \mathbb{Z}_p, m \leftarrow \mathbb{G}_T$.

We prove this theorem by the following two lemmas. Lemma 3 states that Hyb$_0$ and Hyb$_1$ are indistinguishable and Lemma 4 states that in Hyb$_1$ the advantage of the adversary is negligible. Therefore, we conclude that the advantage of the adversary in the original Malicious-Delegator security experiment is negligible.

Lemma 3. Under the DBDH assumption, Hyb$_0$ and Hyb$_1$ are computationally indistinguishable.

Proof Let $\mathcal{A}$ be an adversary that distinguishes Hyb$_0$ and Hyb$_1$ with advantage ϵ. We construct an algorithm $\mathcal{B}$ to solve the DBDH problem by interacting with $\mathcal{A}$. The algorithm $\mathcal{B}$ takes a random challenge (g, g^a, g^b, g^c, T) as input and tries to decide whether $T = e(g,g)^{abc}$ or $T = e(g,g)^t$, where $a, b, c, t \leftarrow \mathbb{Z}_p$ are uniformly chosen at random. The algorithm $\mathcal{B}$ proceeds as follows.

$\mathsf{Setup}(\lambda)$. Choose $\delta, z \leftarrow \mathbb{Z}_p$ uniformly at random. Set

$$g_1 = g, g_2 = g^b, h_1 = g^a,$$

$$h_2 = g^{\delta - bz}, L = e(g^a, g^{\delta - bz}).$$

Return $\mathsf{param} = (g_1, g_2, h_1, h_2, L)$ as the system parameter.

$\mathsf{ProxySet}(\mathsf{param})$**:** Let $\mathsf{sk}_\mathsf{p} = z$ and compute $\mathsf{pk}_\mathsf{p} = Z = g_2^z$. Return $\mathsf{pk}_\mathsf{p}, \mathsf{sk}_\mathsf{p}$.

Phase 1. $\mathcal{B}$ answers $\mathcal{A}$'s queries as follows.

- $\mathcal{O}_{\text{hkg}}(i)$. Choose $x_i, y_i \leftarrow \mathbb{Z}_p$ uniformly at random.
  - If it is the k-th key generation query (that $\mathcal{B}$ guesses it will be the target user), let $i^* = i$, compute
  $$\mathsf{pk}_{i^*} = (X_i = g^{x_{i^*}}, Y_i = g^{y_{i^*}}),$$
  implicitly setting $\mathsf{sk}_{i^*} = (\frac{x_{i^*}}{a}, y_{i^*})$. Return pk_{i^*}.
  - Otherwise, compute
  $$\mathsf{pk}_i = (X_i = h_1^{x_i}, Y_i = g^{y_i}),$$
  implicitly setting $\mathsf{sk}_i = (x_i, y_i)$. Return pk_i.
- $\mathcal{O}_{\text{ckg}}(i)$. Choose $x_i, y_i \leftarrow \mathbb{Z}_p$ uniformly at random and set $\mathsf{sk}_i = (x_i, y_i)$. Compute
  $$\mathsf{pk}_i = (X_i = h_1^{x_i}, Y_i = g^{y_i}).$$
  Return $(\mathsf{sk}_i, \mathsf{pk}_i)$.
- $\mathcal{O}_{\text{rkg}}(\mathsf{pk}_i, \mathsf{pk}_j, \mathsf{pk}_\mathsf{p})$. Given $(\mathsf{pk}_i, \mathsf{pk}_j)$ where pk_i and pk_j were generated by $\mathcal{O}_{\text{hkg}}$ or $\mathcal{O}_{\text{ckg}}$, $\mathcal{B}$ distinguishes the following cases.
  - If $\mathsf{pk}_i \neq \mathsf{pk}_{i^*}$, run $\mathrm{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j, \mathsf{pk}_\mathsf{p})$ and return whatever it outputs;
  - If $\mathsf{pk}_i = \mathsf{pk}_{i^*}$, compute
  $$W = (h_2 Y_j Z)^{(\frac{x_{i^*}}{a})^{-1}} = (g^a)^{\frac{\delta + y_j}{x_{i^*}}}.$$
  Return $\mathsf{rk}_{i \to j} = W$.

Guess. When $\mathcal{A}$ decides to finish Phase 1, it outputs a μ-useful decryption device $D_{*,\mu}$ and a target public key pk^*. If $\mathsf{pk}_{i^*} \neq \mathsf{pk}^*$, $\mathcal{B}$ outputs a random bit and aborts. Otherwise, $\mathcal{B}$ runs $\mathsf{Judge}^{\mathsf{D}_{*,\mu}}(\mathsf{pk}^*, \mathsf{pk}_\mathsf{p})$ as follows.

1) Repeat the following experiment $n = \lambda/\mu$ times.
   - Randomly choose $r_1 \leftarrow \mathbb{Z}_p$, and a message $m \in \mathbb{G}_T$. Compute
   $$c_0 = m \cdot e(g^a, g^c)^\delta \cdot e(g^a, g)^{\delta r_1} \cdot e(g^a, g^b)^{-z r_1} \cdot T^{-z},$$
   $$c_1 = g^c g^{r_1}, c_2 = e(g^a, g^b)^{r_1} T, c_3 = (g^c)^{x_{i^*}} g^{x_{i^*} r_1}.$$
   - Run $D_{*,\mu}$ with the ciphertext $C = (c_0, c_1, c_2, c_3)$ and get the output m'.
   - If $m' = m$, output "Proxy" and exit.
2) Otherwise, output "Delegator".

Observe that in the judge algorithm, if $T = e(g,g)^t$ where t is random, let $r = c + r_1, r' = t(ab)^{-1} + r_1$, then we have

$$c_0 = m \cdot L^r \cdot e(h_1, Z)^{r-r'}, c_1 = g_1^r,$$

$$c_2 = e(h_1, g_2)^{r'}, c_3 = X_i^r.$$

Namely, $\mathcal{B}$ properly simulates in Hyb$_0$. Otherwise, if $T = e(g,g)^{abc}$, let $r = c + r_1$, then we have

$$c_0 = m \cdot L^r, c_1 = g_1^r, c_2 = e(h_1, g_2)^r, c_3 = X_i^r.$$

$\mathcal{B}$ properly simulates in Hyb$_1$. Suppose $\mathcal{A}$ makes $\mathcal{O}_{\text{hkg}}$ queries at most q_{hkg} times, then the probability that $\mathcal{B}$ guesses right i^* in Guess phase is at least $1/q_{\text{hkg}}$. Hence, $\mathcal{B}$ can use $\mathcal{A}$ to solve the DBDH problem and its advantage is at least $1/q_{\text{hkg}} \cdot \epsilon$, which completes the proof. □

Lemma 4. In Hyb$_1$, the advantage of the adversary is negligible.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2018.2877601, IEEE Transactions on Dependable and Secure Computing

Proof In Hyb_1, the adversary outputs a decryption device $D_{*,\mu}$. Then the advantage of the adversary is the probability that the judge algorithm outputs "Delegator".

In a single execution, $D_{*,\mu}$ is fed with ciphertext as

$$c_0 = m \cdot L^r, c_1 = g_1^r, c_2 = e(h_1, g_2)^r, c_3 = X_i^r,$$

where $r \in \mathbb{Z}_p, m \in \mathbb{G}_T$. Note that, the ciphertexts for running the decryption device $D_{*,\mu}$ are all normal ciphertexts. Therefore, $D_{*,\mu}$ returns m' such that $m' = m$ with the probability μ. Finally, we conclude that the advantage of the adversary in Hyb_1 is negligible, since $(1-\mu)^n = (1-\mu)^{\lambda/\mu} \leq \frac{1}{e^\lambda}$. □

Theorem 3. SI is Malicious-Delegator secure under the DBDH assumption.

In order to prove the Malicious-Delegator security, we consider the following two hybrid experiments.

-*Hyb*$_0$ The original Malicious-Delegator security experiment. Note that when the adversary outputs a μ-useful decryption device $D_{*,\mu}$ for a target delegator, the challenger runs $D_{*,\mu}$ with n malformed ciphertexts. The malformed ciphertexts are generated as

$$c_0 = m \cdot L^r \cdot e(h_1, Z)^{r-r'}, c_1 = g_1^r, c_2 = e(h_1, g_2)^{r'}, c_3 = X_i^r,$$

where $r, r' \leftarrow \mathbb{Z}_p, m \leftarrow \mathbb{G}_T$.

-*Hyb*$_1$ The same as Hyb_0 except that in the judge algorithm, the ciphertext is generated as

$$c_0 = m \cdot L^r \cdot e(g, g)^s, c_1 = g_1^r, c_2 = e(h_1, g_2)^{r'}, c_3 = X_i^r,$$

where $r, r', s \leftarrow \mathbb{Z}_p, m \leftarrow \mathbb{G}_T$.

We prove this theorem by the following two lemmas. Lemma 5 states that Hyb_0 and Hyb_1 are indistinguishable and Lemma 6 states that in Hyb_1 the advantage of the adversary is negligible. Therefore, we conclude that the advantage of the adversary in the original Malicious-Delegator security experiment is negligible.

Lemma 5. Under the DBDH assumption, Hyb_0 and Hyb_1 are computationally indistinguishable.

Proof Let $\mathcal{A}$ be an adversary that distinguishes Hyb_0 and Hyb_1 with advantage ϵ. We construct an algorithm $\mathcal{B}$ to solve the DBDH problem by interacting with $\mathcal{A}$. The algorithm $\mathcal{B}$ takes a random challenge (g, g^a, g^b, g^c, T) as input and tries to decide whether $T = e(g,g)^{abc}$ or $T = e(g,g)^t$, where $a, b, c, t \leftarrow \mathbb{Z}_p$. The algorithm $\mathcal{B}$ proceeds as follows.

$\mathsf{Setup}(\lambda)$. Choose $\sigma, \delta \leftarrow \mathbb{Z}_p$ uniformly at random. Set

$$g_1 = g, g_2 = g^\sigma, h_1 = g^a, h_2 = g^\delta, L = e(g^a, g^\delta).$$

Return $\mathsf{param} = (g_1, g_2, h_1, h_2, L)$ as the system parameter.

$\mathsf{ProxySet}(\mathsf{param})$**:** Let $\mathsf{pk_p} = Z = g^b$, implicitly setting $\mathsf{sk_p} = b/\sigma$. Return $\mathsf{pk_p}$.

Phase 1. $\mathcal{B}$ answers $\mathcal{A}$'s queries as follows.

- $\mathcal{O}_{\text{hkg}}(i)$. Choose $x_i, y_i \leftarrow \mathbb{Z}_p$ uniformly at random and set $\mathsf{sk}_i = (x_i, y_i)$. Compute

$$\mathsf{pk}_i = (X_i = h_1^{x_i}, Y_i = g_1^{y_i}).$$

Return pk_i.

- $\mathcal{O}_{\text{ckg}}(i)$. Choose $x_i, y_i \leftarrow \mathbb{Z}_p$ uniformly at random and set $\mathsf{sk}_i = (x_i, y_i)$. Compute

$$\mathsf{pk}_i = (X_i = h_1^{x_i}, Y_i = g_1^{y_i}).$$

Return $(\mathsf{sk}_i, \mathsf{pk}_i)$.

- $\mathcal{O}_{\text{rkg}}(\mathsf{pk}_i, \mathsf{pk}_j, \mathsf{pk_p})$. Given $(\mathsf{pk}_i, \mathsf{pk}_j)$ where pk_i and pk_j were generated by $\mathcal{O}_{\text{hkg}}$ or $\mathcal{O}_{\text{ckg}}$, run $\text{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j, \mathsf{pk_p})$ and return whatever it outputs.

Guess. When $\mathcal{A}$ decides to finish Phase 1, it outputs a μ-useful decryption device $D_{*,\mu}$ and a target public key pk^*. $\mathcal{B}$ runs $\mathsf{Judge}^{\mathsf{D}_{*,\mu}}(\mathsf{pk}^*, \mathsf{pk_p})$ as follows.

1) Repeat the following experiment $n = \lambda/\mu$ times.
   - Choose $m \in \mathbb{G}_T, r, \hat{r} \in \mathbb{Z}_p$ uniformly at random and compute

$$c_0 = m \cdot e(g^a, g^\delta)^r \cdot e(g^a, g^b)^{r-\hat{r}} \cdot T^{-1},$$

$$c_1 = g^r, c_2 = e(g^a, g^c g^{\hat{r}})^\sigma, c_3 = (g^a)^{x_{i^*} r}.$$

   - Run the μ-useful box $D_{*,\mu}$ with the ciphertext $C = (c_0, c_1, c_2, c_3)$ and get the output m'.
   - If $m' = m$, the judge algorithm outputs "Proxy" and exits.
2) Otherwise, the judge algorithm outputs "Delegator".

Observe that in the judge algorithm, let $r' = c + \hat{r}$, if $T = e(g,g)^{abc}$, we have

$$c_0 = m \cdot L^r \cdot e(h_1, Z)^{r-r'},$$

$$c_1 = g_1^r, c_2 = e(h_1, g_2)^{r'}, c_3 = X_i^r.$$

Thus, $\mathcal{B}$ properly simulates in Hyb_0. Otherwise, if $T = e(g,g)^t$ where t is random, let $s = ab(r - \hat{r}) - t$, then we have

$$c_0 = m \cdot L^r \cdot e(g,g)^s, c_1 = g_1^r, c_2 = e(h_1, g_2)^{r'}, c_3 = X_i^r.$$

$\mathcal{B}$ properly simulates in Hyb_1. Hence, $\mathcal{B}$ can use $\mathcal{A}$ to solve the DBDH problem and its advantage is ϵ, which completes the proof. □

Lemma 6. In Hyb_1, the advantage of the adversary is negligible.

Proof In Hyb_1, the adversary outputs a decryption device $D_{*,\mu}$. Then the advantage of the adversary is the probability that the judge algorithm outputs "Proxy".

Assume $h_1 = g^a$ and $h_2 = g^\delta$, where $a, \delta \in \mathbb{Z}_p$. In a single execution, $D_{*,\mu}$ is fed with

$$c_0 = m \cdot L^r \cdot e(g,g)^s = m \cdot L^{r + \frac{s}{a\delta}}, c_1 = g_1^r,$$

$$c_2 = e(h_1, g_2)^{r'}, c_3 = X_i^r,$$

where $r, r', s \in \mathbb{Z}_p, m \in \mathbb{G}_T$. Since r, r', s are uniformly random and independent to the adversary, $D_{*,\mu}$ returns m' such that $m' = m$ with the probability $1/|\mathsf{M}| = \frac{1}{2^\lambda}$. Moreover, according to the second level decryption algorithm, the decryption result would be $m \cdot e(g,g)^r$ (r is uniformly random), which reveals no information of m. Hence, we conclude that the advantage of the adversary in Hyb_1 is negligible, since $(\frac{1}{|\mathsf{M}|})^n < \frac{1}{2^\lambda}$. □

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2018.2877601, IEEE Transactions on Dependable and Secure Computing

VI. Extension and Comparison

In this section, we first give a CCA secure extension. Then, we compare the proposed two constructions with several related PRE schemes in the literature.

A. A CCA Secure Extension

We extend accountable PRE scheme SI to CCA secure, denoted by SII.

$\mathrm{Setup}(\lambda)$: Let λ be the security parameter, $\mathbb{G}$ and $\mathbb{G}_T$ be groups of prime order $p \geq 2^\lambda$, and e: $\mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map. Randomly choose $g_1, g_2, h_1, h_2, u, v, w \leftarrow \mathbb{G}$ and compute $L = e(h_1, h_2)$. Let $H_0 : \{0,1\}^l \times \mathbb{G} \to \mathbb{Z}_p$ be a collision-resistant hash function, $H_1 : \mathbb{G}_T \to \{0,1\}^{l_1}$ be a collision-resistant and one-way hash function, $H_2 : \mathbb{G}_T \to \{0,1\}^{l_2}$ be a universal hash function and $H_3 : \{0,1\}^l \times \mathbb{G} \times \mathbb{G}_T \to \mathbb{Z}_p$ be a collision-resistant hash function, where $l = l_1 + l_2$ and $l < \log p$ such that the generalized leftover hash lemma holds [51] in respect to $\mathrm{negl}(\lambda)$ for security. For simplicity, we let the message space $\mathcal{M} = \{0,1\}^{l_2}$. The system parameters are

$$\mathsf{param} = (g_1, g_2, h_1, h_2, u, v, w, L, H_0, H_1, H_2, H_3).$$

$\mathrm{ProxySet}(\mathsf{param}), \mathrm{KeyGen}(\mathsf{param}), \mathrm{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j, \mathsf{pk}_\mathsf{p})$. The same as in SI.

$\mathrm{Enc}_1(\mathsf{pk}_j, m)$. Given pk_j and m as input, choose $r, \gamma \leftarrow \mathbb{Z}_p$ uniformly at random and compute

$$c_0' = H_1(L^r)||H_2(L^r) \oplus m, c_1' = g_1^r,$$
$$c_2' = L^r \cdot e(h_1, Y_j)^r, c_3' = (u^\psi v^\gamma w)^r$$

where $\psi = H_0(c_0', c_1')$. Set $C_j' = (\gamma, c_0', c_1', c_2', c_3')$ as the first level ciphertext.

$\mathrm{Enc}_2(\mathsf{pk}_i, m)$. Given pk_i and m as input, choose $r, \gamma, \gamma' \leftarrow \mathbb{Z}_p$ uniformly at random and compute

$$c_0 = H_1(L^r)||H_2(L^r) \oplus m, c_1 = g_1^r, c_2 = e(h_1, g_2)^r,$$
$$c_3 = X_i^r, c_4 = (u^\psi v^\gamma w)^r, c_5 = (u^{\psi'} v^{\gamma'} w)^r$$

where $\psi = H_0(c_0, c_1)$ and $\psi' = H_3(c_0, c_1, c_2)$. Set $C_i = (\gamma, c_0, c_1, c_2, c_3, c_4)$ as the second level ciphertext.

$\mathrm{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{sk}_\mathsf{p}, C_i)$. Given $\mathsf{rk}_{i\to j} = W, \mathsf{sk}_\mathsf{p} = z$ and C_i as input, compute $\psi = H_0(c_0, c_1)$ and check the following relations

$$\begin{aligned} e(c_1, X_i) &= e(g_1, c_3), \\ e(c_1, u^\psi v^\gamma w) &= e(g_1, c_4), \\ e(c_1, u^{\psi'} v^{\gamma'} w) &= e(g_1, c_5). \end{aligned} \quad (1)$$

If the above relations do not hold, return "$\perp$". Otherwise, compute

$$c_0' = c_0, c_1' = c_1, c_2' = e(c_3, W)/c_2^z, c_3' = c_4$$

and set $C_j' = (\gamma, c_0', c_1', c_2', c_3')$ as the re-encrypted ciphertext.

$\mathrm{Dec}_1(\mathsf{sk}_j, C_j')$. Given sk_j and C_j' as input, compute $\psi = H_0(c_0, c_1)$ and check the following relation

$$e(c_1', u^\psi v^\gamma w) = e(g_1, c_3'). \quad (2)$$

If the above relation does not hold, return "$\perp$". Otherwise, parse $c_0' = \tau_1||\tau_2$, compute

$$K = c_2'/e(h_1, c_1')^{y_j}.$$

If $\tau_1 = H_1(K)$, return $m = \tau_2 \oplus H_2(K)$, else return "$\perp$".

$\mathrm{Dec}_2(\mathsf{sk}_i, C_i)$. Given sk_i and C_i as input, compute $\psi = H_0(c_0, c_1)$ and check the relations (1). If the above relations do not hold, return "$\perp$". Otherwise, parse $c_0 = \tau_1||\tau_2$ and compute

$$K = e(c_3, h_2)^{1/x_i}.$$

If $\tau_1 = H_1(K)$, return $m = \tau_2 \oplus H_2(K)$, else return "$\perp$".

$\mathrm{Judge}^{D_{i,\mu}}(\mathsf{pk}_i, \mathsf{pk}_\mathsf{p})$. Given the decryption device $D_{i,\mu}$ as an oracle and $\mathsf{pk}_i, \mathsf{pk}_\mathsf{p}$ as input, this algorithm proceeds as follows.

1) Repeat the following experiment $n = \lambda/\mu$ times.
   - Choose $r, r', \gamma, \gamma' \leftarrow \mathbb{Z}_p$ and $m \leftarrow \mathbb{G}_T$ uniformly at random. Compute $C = (c_0, c_1, c_2, c_3, c_4, c_5)$:

$$c_0 = H_1(L^r \cdot e(h_1, Z)^{r-r'})||H_2(L^r \cdot e(h_1, Z)^{r-r'}) \oplus m,$$
$$c_1 = g_1^r, c_2 = e(h_1, g_2)^{r'}, c_3 = X_i^r,$$
$$c_4 = (u^\psi v^\gamma w)^r, c_5 = (u^{\psi'} v^{\gamma'} w)^r$$

     where $\psi = H_0(c_0, c_1)$ and $\psi' = H_3(c_0, c_1, c_2)$.
   - Run the decryption device $D_{i,\mu}$ with C as input and obtain an output m'.
   - If $m' = m$, output "Proxy" and exit.
2) Otherwise, output "Delegator".

Correctness is similar to that of SI.

Theorem 4. Assume H_0 and H_3 are collision-resistant hash functions, H_1 is a collision-resistant and one-way hash function and H_2 is a universal hash function, SII is CCA secure at both the 2nd level and the 1st level ciphertexts under the DBDH assumption.

The CCA security definition and proofs are in Appendix B and Appendix C.

Theorem 5. Assume H_0 and H_3 are collision-resistant hash functions, H_1 is a collision-resistant and one-way hash function and H_2 is a universal hash function, SII is both Malicious-Delegator and Malicious-Delegator secure under the DBDH assumption.

The proof of Theorem 5 is completely similar to the proofs of Theorem 2 and Theorem 3. The only difference is that, to simulate the security experiment, the challenger should choose $u, v, w \in \mathbb{G}$ with known discrete logarithms. We omit the proof here.

B. Comparison

Table I compares our schemes with those in [16] [19] [36] in terms of security and performance, all of which are related to defend against re-encryption key abuse attack. From Table I, it is clear that only in our scheme and [19] the computational cost and ciphertext/key size are constant.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2018.2877601, IEEE Transactions on Dependable and Secure Computing

TABLE I: **General Comparison**

	[36]	[19]	[16]	SI	SII
Security of Ciphertext	CPA	CPA/CCA	CPA	CPA	CCA
Assumption	augmented DBDH	DBDH	i$\mathcal{O}$, k-authentication, PRG	DBDH	DBDH
Against Key Abuse Attack	traceability	unforgeability of re-encryption key	non-transferability	accountability	accountability
Assumption	2-3-CDH	q-SDH	i$\mathcal{O}$, k-authentication, PRG	DBDH	DBDH
Constant Key Size	×	√	×	√	√
Constant Ciphertext Size	×	√	×	√	√
Constant Computational Cost	×	√	×	√	√

TABLE II: **Efficiency Comparison**

Schemes	Key and Ciphertext Size				Computational Cost									
	PK	RK	first level	second level	Enc				ReEnc (ms)		Dec			
					first level (ms)		second level (ms)				first level (ms)		second level (ms)	
[36]	$O(\log N)$	$2\|\mathbb{G}_T\|$	$2t'_e$	$O(\log N)$	$2t'_e$	1.24	$O(\log N)$	–	$2t_p$	12.10	t'_e	0.62	$t_p + t'_e$	6.67
[19]^a	$3\|\mathbb{G}\|$	$\|\mathbb{G}\| + 2\|\mathbb{Z}_p\|$	$2\|\mathbb{G}_T\| + \|\mathbb{G}\|$	$\|\mathbb{G}_T\| + 4\|\mathbb{G}\|$	$2t'_e + t_e$	3.64	$t'_e + 4t_e$	10.22	$2t_p + t_e$	14.5	$t_p + t'_e$	6.67	$t_p + t'_e$	6.67
[19]^b	$3\|\mathbb{G}\|$	$\|\mathbb{G}\| + 2\|\mathbb{Z}_p\|$	$\|\mathbb{G}_T\|+2\|\mathbb{G}\|+ l + \|\mathbb{Z}_p\|$	$5\|\mathbb{G}\|+l+\|\mathbb{Z}_p\|$	$t_m + 2t'_e + t_e$	6.68	$t_m+t'_e+ 4t_e$	13.26	$4t_p + 2t_m + t_e$	32.68	$2t_p + t_m + t_e$	15.76	$3t_p + 2t_m + t'_e$	24.85
[16]	poly(λ)	poly(λ)	poly(λ)	poly(λ)	–	poly(λ)	–	poly(λ)	–	poly(λ)	–	poly(λ)	–	poly(λ)
SI	$2\|\mathbb{G}\|$	$\|\mathbb{G}\|$	$2\|\mathbb{G}_T\| + \|\mathbb{G}\|$	$2\|\mathbb{G}_T\| + 2\|\mathbb{G}\|$	$2t'_e + t_e$	3.64	$2t'_e + 2t_e$	6.04	$t_p + t'_e$	6.67	$t_p + t'_e$	6.67	$t_p + t'_e$	6.67
SII	$2\|\mathbb{G}\|$	$\|\mathbb{G}\|$	$\|\mathbb{G}_T\|+2\|\mathbb{G}\|+ l + \|\mathbb{Z}_p\|$	$\|\mathbb{G}_T\|+4\|\mathbb{G}\|+ l + \|\mathbb{Z}_p\|$	$t_m + 2t'_e + t_e$	6.68	$2t_m + 2t'_e + 2t_e$	12.12	$3t_p + 2t_m + t'_e$	24.85	$2t_p + t_m + t'_e$	15.76	$3t_p + 2t_m + t'_e$	24.85

([19]^a and [19]^b denote the CPA scheme and the CCA scheme proposed in [19], respectively.)

Next, we look into the efficiency comparison in more detail in Table II. t_p, t_m, t'_e and t_e denote the time for computing a bilinear pairing, a multi-exponentiation in group $\mathbb{G}$, an exponentiation in group $\mathbb{G}_T$ and an exponentiation in group $\mathbb{G}$, respectively. $|\mathbb{G}|$, $|\mathbb{G}_T|$ and $|\mathbb{Z}_p|$ denote the bit-length of an element in $\mathbb{G}$, an element in $\mathbb{G}_T$ and an integer in $\mathbb{Z}_p$, respectively. The notation $l = l_1 + l_2$ in our scheme denotes the total length of H_1's output length and the bit-length of the plaintext space. N denotes the maximum number of delegatees for each delegator in [36]. poly(λ) denotes the computational cost or ciphertext/key size is polynomial with respect to the security parameter λ.

For all the compared schemes, we take similar optimizations by pre-computing some bilinear pairings offline[5]. To make it more clearly, we use an implementation [29] of MIRACL CryptoSDK at 80-bit security level as a benchmark to estimate efficiency of pairings and exponentiations, where t_p =6.05ms, t_m =3.04ms, t'_e =0.62ms and t_e =2.4ms[6].

The results show that our scheme has better computation and communication efficiency, meanwhile it also provides additional accountability under the standard security assumption.

VII. Conclusion

Due to the nature of PRE schemes, proxy and any delegatee can collude to derive and distribute the delegator's decryption capability, which has been the major concerns for users utilizing cloud data sharing services. In this paper, we introduced the concept of accountable PRE to resolve this problem. We first formalized the notion of accountable PRE, in which the proxy that abuses its re-encryption key can be identified by the judge algorithm. Then, we presented the first accountable PRE scheme which is non-interactive and public accountable and proved its CPA security and accountability under DBDH assumption in the standard model. When compared with previously related schemes, our scheme offers better performances. A worthwhile direction is to propose an efficient generic transformation with accountable properties of PRE, which may potentially stimulate the adoption of PRE schemes in practice.

[5]In addition, for SII scheme, the relation check (1) is replaced with $e(c_1, X_i^{r_1}(u^{\psi}v^{\gamma}w)^{r_2}(u^{\psi'}v^{\gamma'}w)^{r_3}) = e(g_1, c_3^{r_1}c_4^{r_2}c_5^{r_3})$ by randomly choosing $r_1, r_2, r_3 \in \mathbb{Z}_p$. Moreover, because of the collision resistance of H_1, we omit checking the relation (2) and compute $K = c'_2 \cdot e(c'_3, g_1^{r_1})/e((u^{\psi}v^{\gamma}w)^{r_1}h_1^{y_j}, c'_1)$, where $r_1 \in \mathbb{Z}_p$ is randomly chosen.

[6]Note that even without the estimated running time, the comparison in Table II shows our scheme is relatively more computationally efficient.

Acknowledgment

The authors would like to thank the anonymous reviewers for their very constructive comments. This work was supported by the National Natural Science Foundation of China under Grants 61802021, U1536205, 61572485, and 61802270, and the science and technology projects of SGCC.

References

[1] "Amazon S3," http://aws.amazon.com/s3/.
[2] A. Covert, "Google Drive, iCloud, Dropbox and more compared: what's the best cloud option?" http://gizmodo.com/5904739.
[3] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *Advances in Cryptology-EUROCRYPT'98*. Springer, 1998, pp. 127–144.
[4] L. Xu, X. Wu, and X. Zhang, "A certificateless proxy re-encryption scheme for secure data sharing with public cloud," in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*. ACM, 2012, pp. 1–10.
[5] O. Blazy, X. Bultel, and P. Lafourcade, "Two secure anonymous proxy-based data storages." in *SECRYPT*, 2016, pp. 251–258.
[6] P. Xu, J. Xu, W. Wang, H. Jin, W. Susilo, and D. Zou, "Generally hybrid proxy re-encryption: a secure data sharing among cryptographic clouds," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*. ACM, 2016, pp. 913–918.
[7] C. Zuo, J. Shao, J. K. Liu, G. Wei, and Y. Ling, "Fine-grained two-factor protection mechanism for data sharing in cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 1, pp. 186–196, 2018.
[8] S. Myers and A. Shull, "Practical revocation and key rotation," in *Cryptographers Track at the RSA Conference*. Springer, 2018, pp. 157–178.
[9] R. Canetti and S. Hohenberger, "Chosen-ciphertext secure proxy re-encryption," in *Proceedings of the 14th ACM conference on Computer and communications security*. ACM, 2007, pp. 185–194.

[10] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," in *NDSS*, 2005.

[11] ——, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Transactions on Information and System Security (TISSEC)*, vol. 9, no. 1, pp. 1–30, 2006.

[12] J. Zhang, Z. Zhang, and H. Guo, "Towards secure data distribution systems in mobile cloud computing," *IEEE Trans. Mob. Comput*, vol. 16, no. 11, pp. 3222–3235, 2017.

[13] G. Taban, A. A. Cárdenas, and V. D. Gligor, "Towards a secure and interoperable drm architecture," in *Proceedings of the ACM workshop on Digital rights management*. ACM, 2006, pp. 69–78.

[14] C. Borcea, Y. Polyakov, K. Rohloff, G. Ryan *et al.*, "Picador: End-to-end encrypted publish–subscribe information distribution with proxy re-encryption," *Future Generation Computer Systems*, vol. 71, pp. 177–191, 2017.

[15] Y. Polyakov, K. Rohloff, G. Sahu, and V. Vaikuntanathan, "Fast proxy re-encryption for publish/subscribe systems," *ACM Transactions on Privacy and Security (TOPS)*, vol. 20, no. 4, p. 14, 2017.

[16] H. Guo, Z. Zhang, and J. Xu, "Non-transferable proxy re-encryption," *IACR Cryptology ePrint Archive*, 2015.

[17] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters, "Candidate indistinguishability obfuscation and functional encryption for all circuits," in *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*. IEEE, 2013, pp. 40–49.

[18] D. Boneh, D. Gupta, I. Mironov, and A. Sahai, "Hosting services on an untrusted cloud," in *Advances in Cryptology-EUROCRYPT 2015*. Springer, 2015, pp. 404–436.

[19] H. Guo, Z. Zhang, and J. Zhang, "Proxy re-encryption with unforgeable re-encryption keys," in *Cryptology and Network Security*. Springer, 2014, pp. 20–33.

[20] D. Boneh and J. Katz, "Improved efficiency for cca-secure cryptosystems built using identity-based encryption," in *Topics in Cryptology–CT-RSA 2005*. Springer, 2005, pp. 87–103.

[21] M. H. Au, W. Susilo, and Y. Mu, "Constant-size dynamic k-TAA," in *Security and Cryptography for Networks*. Springer, 2006, pp. 111–125.

[22] J. Lai, R. H. Deng, Y. Zhao, and J. Weng, "Accountable authority identity-based encryption with public traceability," in *Topics in Cryptology–CT-RSA 2013*. Springer, 2013, pp. 326–342.

[23] N. Chandran, M. Chase, and V. Vaikuntanathan, "Functional re-encryption and collusion-resistant obfuscation," in *Theory of Cryptography*. Springer, 2012, pp. 404–421.

[24] N. Chandran, M. Chase, F.-H. Liu, R. Nishimaki, and K. Xagawa, "Re-encryption, functional re-encryption, and multi-hop re-encryption: A framework for achieving obfuscation-based security and instantiations from lattices," in *International Workshop on Public Key Cryptography*. Springer, 2014, pp. 95–112.

[25] S. Hohenberger, G. N. Rothblum, V. Vaikuntanathan *et al.*, "Securely obfuscating re-encryption," in *Theory of Cryptography*. Springer, 2007, pp. 233–252.

[26] M. Green and G. Ateniese, "Identity-based proxy re-encryption," in *Applied Cryptography and Network Security*. Springer, 2007, pp. 288–306.

[27] B. Libert and D. Vergnaud, "Unidirectional chosen-ciphertext secure proxy re-encryption," in *Public Key Cryptography-PKC 2008*. Springer, 2008, pp. 360–379.

[28] S. S. Chow, J. Weng, Y. Yang, and R. H. Deng, "Efficient unidirectional proxy re-encryption," in *Progress in Cryptology-AFRICACRYPT 2010*. Springer, 2010, pp. 316–332.

[29] J. Zhang, Z. Zhang, and Y. Chen, "PRE: Stronger security notions and efficient construction with non-interactive opening," *Theoretical Computer Science*, 2014.

[30] X. Fan and F.-H. Liu, "Proxy re-encryption and re-signatures from lattices," Cryptology ePrint Archive, Report 2017/456, Tech. Rep., 2017.

[31] Q. Tang, "Type-based proxy re-encryption and its construction," in *International Conference on Cryptology in India*. Springer, 2008, pp. 130–144.

[32] J. Weng, M. Chen, Y. Yang, R. Deng, K. Chen, and F. Bao, "CCA-secure unidirectional proxy re-encryption in the adaptive corruption model without random oracles," *Science China Information Sciences*, vol. 53, no. 3, pp. 593–606, 2010.

[33] K. Liang, W. Susilo, and J. K. Liu, "Privacy-preserving ciphertext multi-sharing control for big data storage," *IEEE transactions on information forensics and security*, vol. 10, no. 8, pp. 1578–1589, 2015.

[34] D. Derler, S. Krenn, T. Lorünser, S. Ramacher, D. Slamanig, and C. Striecks, "Revisiting proxy re-encryption: Forward secrecy, improved security, and applications," in *IACR International Workshop on Public Key Cryptography*. Springer, 2018, pp. 219–250.

[35] S. Myers and A. Shull, "Efficient hybrid proxy re-encryption for practical revocation and key rotation," Cryptology ePrint Archive, Report 2017/833, Tech. Rep., 2017.

[36] B. Libert and D. Vergnaud, "Tracing malicious proxies in proxy re-encryption," in *Pairing-Based Cryptography-Pairing 2008*. Springer, 2008, pp. 332–353.

[37] R. Hayashi, T. Matsushita, T. Yoshida, Y. Fujii, and K. Okada, "Unforgeability of re-encryption keys against collusion attack in proxy re-encryption," in *Advances in Information and Computer Security*. Springer, 2011, pp. 210–229.

[38] T. Isshiki, M. H. Nguyen, and K. Tanaka, "Attacks to the proxy re-encryption schemes from iwsec2011," in *Advances in Information and Computer Security*. Springer, 2013, pp. 290–302.

[39] Y.-J. He, T. W. Chim, L. C. K. Hui, and S.-M. Yiu, "Non-transferable proxy re-encryption scheme," in *New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on*. IEEE, 2012, pp. 1–4.

[40] X. A. Wang, F. Xhafa, W. Hao, and W. He, "Non-transferable unidirectional proxy re-encryption scheme for secure social cloud storage sharing," in *International Conference on Intelligent Networking and Collaborative Systems (INCoS)*. IEEE, 2016, pp. 328–331.

[41] E. M. Cho, L. San, and T. Koshiba, "Secure non-transferable proxy re-encryption for group membership and non-membership," in *International Conference on Network-Based Information Systems*. Springer, 2017, pp. 876–887.

[42] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Advances in cryptology*. Springer, 1984, pp. 47–53.

[43] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Advances in Cryptology-CRYPTO 2001*. Springer, 2001, pp. 213–229.

[44] C. Cocks, "An identity based encryption scheme based on quadratic residues," in *Cryptography and coding*. Springer, 2001, pp. 360–363.

[45] V. Goyal, "Reducing trust in the PKG in identity based cryptosystems," in *Advances in Cryptology-CRYPTO 2007*. Springer, 2007, pp. 430–447.

[46] V. Goyal, S. Lu, A. Sahai, and B. Waters, "Black-box accountable authority identity-based encryption," in *Proceedings of the 15th ACM conference on Computer and communications security*. ACM, 2008, pp. 427–436.

[47] B. Libert and D. Vergnaud, "Towards black-box accountable authority IBE with short ciphertexts and private keys," in *Public Key Cryptography–PKC 2009*. Springer, 2009, pp. 235–255.

[48] A. Sahai and H. Seyalioglu, "Fully secure accountable-authority identity-based encryption," in *Public Key Cryptography–PKC 2011*. Springer, 2011, pp. 296–316.

[49] A. Kiayias and Q. Tang, "Making any identity-based encryption accountable, efficiently," in *Computer Security–ESORICS 2015*. Springer, 2015, pp. 326–346.

[50] D. Boneh and X. Boyen, "Efficient selective-ID secure identity-based encryption without random oracles," in *Advances in Cryptology-EUROCRYPT 2004*. Springer, 2004, pp. 223–238.

[51] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," *SIAM Journal on Computing*, vol. 38, no. 1, pp. 97–139, 2008.

Hui Guo received the PhD degree from Institute of Software, Chinese Academy of Sciences in 2016. She is currently an assistant researcher with State Key Laboratory of Cryptology. Her research interests include applied cryptography, proxy re-encryption, and obfuscation.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2018.2877601, IEEE Transactions on Dependable and Secure Computing

Zhenfeng Zhang received the PhD degree from Academy of Mathematics and Systems Science, Chinese Academy of Sciences in 2001. He is currently a research professor and director with Trusted Computing and Information Assurance Laboratory, Institute of Software, Chinese Academy of Sciences. His main research interests include applied cryptography, security protocol and trusted computing. He has published a series of papers at CCS, S&P, CRYPTO, EUROCRYPT, ASIACRYPT etc, and has 1 technology standardized in ISO/IEC 20009-4. He is a senior member of Chinese Association for Cryptologic Research.

Jing Xu received the PhD degree in Computer Theory from Academy of Mathematics and Systems Science, Chinese Academy of Sciences in June 2002. She is currently a research professor with Trusted Computing and Information Assurance Laboratory, Institute of Software, Chinese Academy of Sciences. Her research interests include applied cryptography and security protocol. She is a senior member of Chinese Association for Cryptologic Research.

Ningyu An received the MS degree from Institute of Information Engineering, Chinese Academy of Sciences in July 2013. He is currently an engineer with Global Energy Interconnection Research Institute co.,Ltd. His research interests include information security on electric power system.

Xiao Lan received the PhD degree from Institute of Information Engineering, Chinese Academy of Sciences in January 2018. She is currently an assistant researcher with Cybersecurity Research Institute, Sichuan University. Her research interests include applied cryptography and authenticated key exchange protocol.