Improved Anonymous Proxy Re-encryption with CCA Security

Qingji Zheng
Department of Computer Science
University of Texas at San Antonio, TX, USA
qingjizheng@gmail.com

Wei Zhu
Julymobile Tech Co., Ltd
Anhui, China
zhuweijm@sina.com

Jiafeng Zhu
Huawei Research Center
Santa Clara, CA, USA
edwardzhu95@gmail.com

Xinwen Zhang
Huawei Research Center
Santa Clara, CA, USA
xinwenzhang@gmail.com

ABSTRACT

Outsourcing private data and heavy computation tasks to the cloud may lead to privacy breach as attackers (e.g., malicious outsiders or cloud administrators) may correlate any relevant information to penetrate information of their interests. Therefore, how to preserve cloud users' privacy has been a top concern when adopting cloud solutions. In this paper, we investigate the identity privacy problem for the proxy re-encryption, which allows any third party (e.g., cloud) to re-encrypt ciphertexts in order to delegate the decryption right from one to another user. The relevant identity information, e.g., whose ciphertext was re-encrypted to the ciphertext under whose public key, may leak because re-encryption keys and ciphertexts (before and after re-encryption) are known to the third party. We review prior anonymity (identity privacy) notions, and find that these notions are either impractical or too weak. To address this problem thoroughly, we rigorously define the anonymity notion that not only embraces the prior anonymity notions but also captures the necessary anonymity requirement for practical applications. In addition, we propose a new and efficient proxy re-encryption scheme. The scheme satisfies the proposed anonymity notion under the Squared Decisional Bilinear Diffie-Hellman assumption and achieves security against chosen ciphertext attack under the Decisional Bilinear Diffie-Hellman assumption in the random oracle model. To the best of our knowledge, it is the *first* proxy re-encryption scheme attaining both chosen-ciphertext security and anonymity simultaneously. We implement a prototype based on the proposed proxy re-encryption scheme and the performance study shows that it is efficient.

ASIA CCS'14, June 4–6, 2014, Kyoto, Japan.

http://dx.doi.org/10.1145/2590296.2590322.

Categories and Subject Descriptors

E.3 [**Data Encryption**]: Public Key Cryptosystems; H.3.4 [**Information Storage and Retrieval**]: Distributed Systems

General Terms

Security

Keywords

proxy re-encryption, anonymity, chosen-ciphertext security, outsourced computation

1. INTRODUCTION

Cloud computing has become an increasingly popular computing paradigm as it offers numerous benefits, e.g., on-demand service, service elasticity and low maintenance cost. However, when outsourcing data and heavy computation tasks to the cloud, cloud users can enjoy these benefits only if they are willing to assume that the cloud is full trusted. This assumption is inevitably impractical because the cloud imposes great attack surface from the inside and outside attackers. Therefore, privacy concern might hinder cloud users deploying applications in the public cloud.

In this paper, we study the anonymity problem for Proxy Re-Encryption (PRE), which is to delegate the decryption right by re-encrypting ciphertexts decrypted by Alice to new ciphertexts that can be decrypted by Bob without revealing plaintexts. In order to understand the anonymity problem for PRE, consider the following application: Company A has an email gateway handling all incoming encrypted emails and forwarding them to appropriate recipients. Alice (an employee of Company A) can send a re-encryption request to the gateway in order to delegate the decryption right to Bob (another employee of Company A) when she is unable to manage her emails properly (e.g., when she is traveling or on vacation). Due to large amount of re-encryption requests and its own limited computation capability, the gateway will outsource those re-encryption operations to a public cloud. As re-encryption keys and corresponding ciphertexts need to be delivered to the cloud, it is expected that the re-encryption procedure does not reveal any participant's

identity, which otherwise might leak important business information (e.g., schedule information).

To address this concern, the anonymity notion (or key-private) for PRE was introduced by Ateniese et al. [5] [1]. They pointed out that many prior proxy re-encryption schemes (before their paper) cannot satisfy the intuition of anonymity: an adversary cannot learn identities by observing sufficient ciphertexts (before or after re-encryption) and re-encryption keys, even it colludes with some corrupted users. That is, for any probabilistically polynomial time adversary, it cannot distinguish a valid re-encryption key for a pair of users, from a random one selected from the re-encryption key space, although it can access re-encryption key generation oracle and re-encryption oracle (perhaps decryption oracle additionally).

Why prior anonymity notions are insufficient? There are two anonymity notions in the literature. The first anonymity notion of PRE in [5] suffers from the limitation that it only allows one re-encryption key for any pair of users. As stated in [5], the proposed scheme cannot achieve the anonymity definition if someone could obtain multiple re-encryption keys per pair of users. That is, if the cloud possesses multiple re-encryption keys for the same pair of users, then it can derive the identities involved in these re-encryption keys. Let us take the above motivational application as an example: If the cloud received two re-encryption keys that were used to delegate the decryption right from Alice to Bob at different times, then it is able to infer that the re-encryption key corresponds to Alice and Bob. Consequently, the constraint of allowing only one re-encryption key per pair of users does not meet the requirement in practice, because in reality it likely happens that multiple re-encryption keys coexist for any pair of users.

Another anonymity notion was introduced by [19, 20]. While their notion allows multiple re-encryption keys coexisting for any pair of users, it achieves weak anonymity because the adversary is not allowed to access the re-encryption oracle (section 2.2.4 [19] and section 2.3.3 [20]) to obtain re-encrypted ciphertexts. Instead, the adversary has to query the re-encryption key generation oracle and applies the returned re-encryption key to compute re-encrypted ciphertexts. Intuitively, their notion fails to capture the following attack: If knowing the identity corresponding to the re-encrypted ciphertext, then the attack may be possible to infer the identities involved in the re-encryption key that is used to generate that re-encrypted ciphertext. To further understand this attack, let us look at the scenario of the motivational example: The cloud receives a re-encryption key to delegate the decryption right from Alice to Bob, then it re-encrypts the ciphertext and sends the re-encrypted ciphertext to the gateway. Since the cloud communicates with the gateway without private channel (all communication data is encrypted), an attacker can eavesdrop the channel to obtain the re-encrypted ciphertext and might learn the related identity without seeing the re-encryption key (perhaps via some side channels). As long as the attacker sees the re-encryption key, he might be able to infer the participants's identity because the corresponding re-encrypted ciphertexts leaks that information. Especially, this attack always happens when the re-encryption algorithm is deterministic. Indeed, this is the reason why the scheme in [19] satisfies their anonymity notion while still suffering from this attack. Therefore, disallowing the adversary directly accessing re-encryption oracle weakens its capability of distinguishing the difference between a valid re-encryption key and a random one from the re-encryption key space.

[1]The anonymity for PRE has two-fold: the anonymity of ciphertexts and the anonymity of re-encryption keys. The former has been investigated by Bellare et al [7] and the latter one is the additional target in the context of PRE, which is the focus of this paper.

Our contributions. As prior anonymity notions are incomplete, we rigorously define the anonymity notion which adequately fixes these insufficiencies. We propose the first PRE construction that not only achieves chosen-ciphertext security (CCA) in the random oracle model, but also attains the anonymity property. To be specific, our contribution can be characterized as follows:

- We rigorously define the anonymity notion with a well-defined anonymous game, which cannot be extended trivially from that in [19, 20] by simply providing access to the re-encryption oracle.
- We present a single-hop, unidirectional PRE scheme satisfies the proposed anonymity notion. Our construction is anonymous under the Squared Decisional Bilinear Diffie-Hellman assumption and CCA secure under the Decisional Bilinear Diffie-Hellman assumption in the random oracle model, which resolves one of the open problems stated in [5].
- Towards building a generic cloud-based re-encryption service, we have built a prototype that integrates the proposed scheme and the Amazon Web Services (AWS), where EC2 is for re-encryption computation and S3 for data storage. We defined general web services interfaces that can be easily integrated into existing applications to provide anonymous re-encryption function. As a case study, we have implemented an anonymous email forwarding service built upon our scheme. Our experimental evaluation demonstrates that the proposed anonymous PRE scheme is efficient.

Paper organization. Section 2 reviews the definitions of single-hop, unidirectional PRE and its CCA security, and then proposes a strong anonymity notion. Section 3 constructs an anonymous and CCA secure PRE and shows its security analysis. Section 4 presents the detail of our implementation and performance study. Section 5 summarizes related work and Section 6 concludes this paper.

2. PRE: DEFINITION AND SECURITY NOTIONS

Let $s \stackrel{R}{\leftarrow} S$ denote selecting element s from set S uniformly at random, $\perp$ denote an error message, and $||$ denote string concatenation. In the present paper we consider single-hop, unidirectional proxy re-encryption only, and name an original ciphertext as a *second level ciphertext* and the ciphertext after re-encryption as a *first level ciphertext* as in [6, 17]. Let $\mathbb{U}$ be the set of users, denoted by $\mathbb{U} = \{1, \cdots, n\}$. Denote the set of uncorrupted users by $\mathbb{U}_h$ and the set of corrupted users by $\mathbb{U}_e$, such that $\mathbb{U}_h \bigcup \mathbb{U}_e = \mathbb{U}$. The following PRE definition and the CCA notion are based on prior work [6, 5, 9].

DEFINITION 1. *A single-hop, unidirectional* PRE *consists of following algorithms:*

- $\mathsf{Param} \leftarrow \mathsf{Setup}(1^\ell)$*: Given a security parameter* 1^ℓ*, this algorithm is run by the trusted party to generate the public parameter* Param*. For brevity, we assume that the following algorithms implicitly take* Param *as parts of inputs.*
- $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(i)$*: This algorithm is run by the user* i *to generate public and private keys* $(\mathsf{pk}_i, \mathsf{sk}_i)$*.*
- $\mathsf{rk}_{i\to j} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$*: This algorithm is run by user* i *to output a re-encryption key* $\mathsf{rk}_{i\to j}$*, which can be used to re-encrypt second level ciphertexts decrypted by user* i *to first level ciphertexts decrypted by user* j*.*
- $\mathsf{C} \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mathsf{m})$*: This algorithm is run to encrypt message* m *to a second level ciphertext* C *which can be decrypted by user* i*.*
- $\mathsf{C}' \leftarrow \mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{C})$*: This algorithm is run to re-encrypt a second level ciphertext* C *to a first level ciphertext* C' *which can be decrypted by user* j*.*
- $\{\mathsf{m}, \perp\} \leftarrow \mathsf{Dec}(\mathsf{sk}_i, \mathsf{C}(\mathsf{C}'))$*: This algorithm is run to decrypt a second level ciphertext* C *(or a first level ciphertext* C'*), which might output the message* m *or a error message* $\perp$*.*

Correctness: A single-hop, unidirectional PRE is correct if: given any m from the message space, (i) $\forall\, i \in \mathbb{U}, (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(i)$, so that $\mathsf{Dec}(\mathsf{sk}_i, \mathsf{Enc}(\mathsf{pk}_i, \mathsf{m})) = \mathsf{m}$, and (ii) $\forall\, i, j \in \mathbb{U}, (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(i)$, $(\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{KeyGen}(j)$, and $\mathsf{rk}_{i\to j} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$, so that $\mathsf{Dec}(\mathsf{sk}_j, \mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{Enc}(\mathsf{pk}_i, \mathsf{m}))) = \mathsf{m}$.

CCA Security: The CCA notion [9] can be formalized by the following CCA-security game between an adversary $\mathcal{A}$ and the challenger $\mathcal{C}$.

Phase 1: $\mathcal{C}$ runs $\mathsf{Setup}(1^\ell)$ to initialize the public parameter. $\mathcal{A}$ can access oracles below in polynomially many times. Note that before querying $\mathcal{O}_{\mathsf{ReKeyGen}}(\mathsf{pk}_i, \mathsf{pk}_j)$ and $\mathcal{O}_{\mathsf{ReEnc}}(\mathsf{pk}_i, \mathsf{pk}_j, \mathsf{C})$, $(\mathsf{pk}_i, \mathsf{sk}_i)$ and $(\mathsf{pk}_j, \mathsf{sk}_j)$ have been generated.

- key generation oracle $\mathcal{O}_{\mathsf{KeyGen}}(i)$: (i) If i has not been queried before, $\mathcal{C}$ runs KeyGen to obtain $(\mathsf{pk}_i, \mathsf{sk}_i)$ and records $(i, \mathsf{pk}_i, \mathsf{sk}_i)$. If i is a corrupted user, s.t. $i \in \mathbb{U}_e$, $\mathcal{C}$ returns $(\mathsf{pk}_i, \mathsf{sk}_i)$ to $\mathcal{A}$, and pk_i otherwise. (ii) Otherwise $\mathcal{C}$ retrieves $(\mathsf{pk}_i, \mathsf{sk}_i)$ and returns $(\mathsf{pk}_i, \mathsf{sk}_i)$ to $\mathcal{A}$ if $i \in \mathbb{U}_e$, and pk_i otherwise.
- Re-encryption key generation oracle $\mathcal{O}_{\mathsf{ReKeyGen}}(\mathsf{pk}_i, \mathsf{pk}_j)$: If $i \in \mathbb{U}_h$ and $j \in \mathbb{U}_e$, $\mathcal{C}$ outputs $\perp$; otherwise $\mathcal{C}$ runs ReKeyGen to obtain $\mathsf{rk}_{i\to j}$, and returns it to $\mathcal{A}$.
- Re-encryption oracle $\mathcal{O}_{\mathsf{ReEnc}}(\mathsf{pk}_i, \mathsf{pk}_j, C)$: $\mathcal{C}$ runs $\mathsf{rk}_{i\to j} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$, computes $\mathsf{C}' \leftarrow \mathsf{ReEnc}(\mathsf{rk}_{i\to j}, C)$, and returns C' to $\mathcal{A}$.
- Decryption oracle $\mathcal{O}_{\mathsf{Dec}}(\mathsf{pk}_i, \mathsf{C})$: Given the ciphertext C (either a first level ciphertext or second level ciphertext), $\mathcal{C}$ runs $\{\mathsf{m}, \perp\} \leftarrow \mathsf{Dec}(\mathsf{sk}_i, \mathsf{C})$ and returns the result to $\mathcal{A}$.

Challenge: Once $\mathcal{A}$ decides that Phase 1 is finished, it outputs a public key pk^*, and two messages $\mathsf{m}_0, \mathsf{m}_1$ of equal length, and sends them to $\mathcal{C}$. $\mathcal{C}$ selects $\lambda \xleftarrow{R} \{0,1\}$, runs $\mathsf{C}^* \leftarrow \mathsf{Enc}(\mathsf{pk}^*, \mathsf{m}_\lambda)$, and returns C^* to $\mathcal{A}$. Here the user with respect to pk^* should be uncorrupted, namely belonging to $\mathbb{U}_h$.

Phase 2: $\mathcal{A}$ queries oracles in polynomially many times with these restrictions.

- $\mathcal{O}_{\mathsf{KeyGen}}(i)$ and $\mathcal{O}_{\mathsf{ReKeyGen}}(\mathsf{pk}_i, \mathsf{pk}_j)$: Work as in Phase 1.
- $\mathcal{O}_{\mathsf{ReEnc}}(\mathsf{pk}_i, \mathsf{pk}_j, \mathsf{C})$: If $i \in \mathbb{U}_h$ and $j \in \mathbb{U}_e$, output $\perp$, otherwise work as in Phase 1.
- $\mathcal{O}_{\mathsf{Dec}}(\mathsf{pk}_i, \mathsf{C})$: Work as in Phase 1, subject to the conditions that (i)$\mathcal{A}$ cannot query $\mathcal{O}_{\mathsf{Dec}}$ if $\mathsf{pk}_i = \mathsf{pk}^*$ and $\mathsf{C} = \mathsf{C}^*$, and (ii) $\mathcal{A}$ cannot query $\mathcal{O}_{\mathsf{Dec}}$ if C is the ciphertext computed with a re-encrypted key of pk^* and pk_i.

Guess: Finally, $\mathcal{A}$ outputs a guess $\lambda' \in \{0,1\}$. $\mathcal{A}$ wins the game if $\lambda' = \lambda$.

DEFINITION 2. *Given any polynomial time adversary* $\mathcal{A}$*, we say a single-hop, unidirectional* PRE *scheme is* CCA *secure for second level ciphertexts, if the probability of* $\mathcal{A}$ *winning* CCA*-security game is at most* $\frac{1}{2} + \epsilon(\ell)$*, where* ϵ *is negligible in security parameter* ℓ*.*

Definition 2 describes the CCA security for second level ciphertexts. We also can define a complementary definition of CCA security for first level ciphertexts. We skip this and refer readers to [17].

Anonymity notion: Now we define the anonymity notion for re-encryption keys, which is partially inspired by prior anonymity notions in [5, 19, 20]. The anonymity notion for second level ciphertexts can be referred to [7].

We simulate the following anonymous game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, where $\mathcal{C}$ maintains a re-encryption key list storing all $\mathsf{rk}_{i\to j}$ for users i and j with the form $(i \to j, \{(\mathsf{rk}_{i\to j}, query)\})$. $i \to j$ is the search key to identify each pair of users, and $\{(\mathsf{rk}_{i\to j}, query)\}$ is a set of re-encryption keys, each of which is associated with a boolean indicator $query$. If $\mathsf{rk}_{i\to j}$ has been queried via $\mathcal{O}_{\mathsf{ReKeyGen}}$, then $query = 1$; otherwise $query = 0$. Initially, $\{(\mathsf{rk}_{i\to j}, query)\}$ is empty for the search key (i, j). The anonymous game proceeds as follows, where $\mathcal{C}$ runs $\mathsf{Setup}(1^\ell)$ to set up the public parameter.

Phase 1: The adversary $\mathcal{A}$ queries oracles below in polynomially many times.

- $\mathcal{O}_{\mathsf{KeyGen}}(i)$: It works the same as $\mathcal{O}_{\mathsf{KeyGen}}(i)$ described in CCA-security game.
- $\mathcal{O}_{\mathsf{ReKeyGen}}(\mathsf{pk}_i, \mathsf{pk}_j)$: Given pk_i and pk_j, $\mathcal{C}$ fetches $(i \to j, \{(\mathsf{rk}_{i\to j}, query)\})$ and works as follows:
  1. If $\{(\mathsf{rk}_{i\to j}, query)\}$ is `null` or all $\mathsf{rk}_{i\to j}$ have been queried (i.e., all indicators $query = 1$), $\mathcal{C}$ runs $\mathsf{rk}_{i\to j} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$, adds $(\mathsf{rk}_{i\to j}, query = 1)$ to the set $\{(\mathsf{rk}_{i\to j}, query)\}$, and returns $\mathsf{rk}_{i\to j}$;
  2. Otherwise $\mathcal{C}$ selects $\mathsf{rk}_{i\to j}$ from $\{(\mathsf{rk}_{i\to j}, query = 0)\}$ uniformly at random, updates $query = 1$ for $\mathsf{rk}_{i\to j}$, and passes $\mathsf{rk}_{i\to j}$ to $\mathcal{A}$.

- $\mathcal{O}_{\mathsf{ReEnc}}(\mathsf{pk}_i, \mathsf{pk}_j, \mathsf{C})$: Given pk_i and pk_j, $\mathcal{C}$ fetches $(i \to j, \{(\mathsf{rk}_{i\to j}, query)\})$ and works as follows:
  1. If $\{(\mathsf{rk}_{i\to j}, query)\}$ is $\mathtt{null}$, $\mathcal{C}$ runs $\mathsf{rk}_{i\to j} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$ and adds $(\mathsf{rk}_{i\to j}, query = 0)$ to the set $\{(\mathsf{rk}_{i\to j}, query)\}$;
  2. Otherwise $\mathcal{C}$ randomly selects $\mathsf{rk}_{i\to j}$ from the set $\{(\mathsf{rk}_{i\to j}, query)\}$.

  Finally, $\mathcal{C}$ runs $\mathsf{C}' \leftarrow \mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{C})$ and returns C' to $\mathcal{A}$.
- $\mathcal{O}_{\mathsf{Dec}}(\mathsf{pk}_i, \mathsf{C})$: $\mathcal{C}$ executes $\{\mathsf{m}, \perp\} \leftarrow \mathsf{Dec}(\mathsf{sk}_i, \mathsf{C})$ and returns the result to $\mathcal{A}$. Note that C can be either a first level ciphertext or a second level ciphertext.

Challenge: Once $\mathcal{A}$ decides that Phase 1 is finished, it outputs a pair of users (i^*, j^*). $\mathcal{C}$ fetches $(i^* \to j^*, \{(\mathsf{rk}_{i^*\to j^*}, query)\})$ and works as follows:

- If $i^* \in \mathbb{U}_e$ or $j^* \in \mathbb{U}_e$, $\mathcal{C}$ aborts;
- If the set $\{(\mathsf{rk}_{i^*\to j^*}, query)\}$ is $\mathtt{null}$ or all $\mathsf{rk}_{i^*\to j^*}$ have been queried (i.e., all indicators $query = 1$), $\mathcal{C}$ runs $\mathsf{rk}_{i^*\to j^*} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_{i^*}, \mathsf{pk}_{j^*})$, and adds $(\mathsf{rk}_{i^*\to j^*}, query = 1)$ to the set $\{(\mathsf{rk}_{i^*\to j^*}, query)\}$;
- Otherwise $\mathcal{C}$ selects a $\mathsf{rk}_{i^*\to j^*}$ from $\{(\mathsf{rk}_{i^*\to j^*}, query = 0)\}$ uniformly at random, and updates $query = 1$ for $\mathsf{rk}_{i^*\to j^*}$.

$\mathcal{C}$ selects $\lambda \overset{R}{\leftarrow} \{0,1\}$. If $\lambda = 0$, $\mathcal{C}$ delivers $\mathsf{rk}_{i^*\to j^*}$ to $\mathcal{A}$, otherwise returns a re-encrypted key selected from the re-encryption key space uniformly at random.

Phase 2: $\mathcal{A}$ can continually query the oracles in polynomially many times.

- $\mathcal{O}_{\mathsf{KeyGen}}(i)$, $\mathcal{O}_{\mathsf{ReKeyGen}}(\mathsf{pk}_i, \mathsf{pk}_j)$ and $\mathcal{O}_{\mathsf{ReEnc}}(\mathsf{pk}_i, \mathsf{pk}_j, \mathsf{C})$: Works as in Phase 1.
- $\mathcal{O}_{\mathsf{Dec}}(\mathsf{pk}_j, \mathsf{C})$: Work as in Phase 1 while being subject to the condition that $\mathcal{A}$ cannot query $\mathcal{O}_{\mathsf{Dec}}$, where $\mathsf{pk}_j = \mathsf{pk}_{j^*}$. [2]

Guess: Eventually, $\mathcal{A}$ outputs a guess $\lambda' \in \{0,1\}$ and wins the game if $\lambda' = \lambda$.

REMARK 1. *Let us consider the challenge phase. If $\lambda = 0$, then $\mathcal{C}$ returns the valid re-encryption key $\mathsf{rk}_{i^*\to j^*}$. Access to $\mathcal{O}_{\mathsf{ReEnc}}(\mathsf{pk}_i, \mathsf{pk}_j, \mathsf{C})$ enables $\mathcal{A}$ to obtain enough first level ciphertexts re-encrypted with $\mathsf{rk}_{i^*\to j^*}$. However, the anonymity notion [19, 20] cannot model this case, because $\mathcal{O}_{\mathsf{ReKeyGen}}(\mathsf{pk}_i, \mathsf{pk}_j)$ does not return this $\mathsf{rk}_{i^*\to j^*}$ to $\mathcal{A}$.*

DEFINITION 3. *We say a single hop, unidirectional* PRE *scheme is anonymous for re-encryption keys, if for any polynomial time adversary $\mathcal{A}$, the probability of $\mathcal{A}$ winning the* anonymous *game is at most $\frac{1}{2} + \epsilon(\ell)$, where ϵ is negligible in security parameter ℓ.*

Definition 3 characterizes the anonymity of the re-encrypted key and [5] has discussed that the anonymity of re-encryption keys implies the anonymity of first level ciphertexts.

[2]The reason is that it is hard for the challenger to distinguish whether C is a first level ciphertext computed by the challenge re-encrypted key or not. This problem also exists in [19, 20] in order to simulate the decryption oracle.

3. ANONYMOUS AND CCA SECURE PRE CONSTRUCTION

3.1 Cryptographic Assumptions

Let G, G_T be two cyclic groups of order q, a ℓ_1-bit prime, and g be a generator of G. Let e be a bilinear map: $e : G\times G \to G_T$ satisfying: (i) $\forall x, y \overset{R}{\leftarrow} \mathbb{Z}_q$, $e(g^x, g^y) = e(g,g)^{xy}$, (ii) $e(g,g) \neq 1$, and (iii) e can be computed efficiently.

Let H_0, H_1, H_2, H_3 be secure hash functions modeled as random oracles, s.t. $H_0 : G \to G, H_1 : \{0,1\}^{\ell_2} \times G_T \to \mathbb{Z}_q, H_2 : G\times G_T \times G \times \{0,1\}^{\ell_2} \to G$, $H_3 : \{0,1\}^{6\ell_1+\ell_2}\times G_T \to \mathbb{Z}_q$ and $H_4 : \mathbb{Z}_q \to \mathbb{Z}_q$, where ℓ_2 is another security parameter. Let F_1, F_2 be two secure pseudorandom generators, where $F_1 : G_T \to \{0,1\}^{\ell_2}$ and $F_2 : G_T \to \{0,1\}^{6\ell_1+\ell_2}$.

Decisional Bilinear Diffie-Hellman (DBDH) Given (g, g^x, g^y, g^z, Q) where $x, y, z \overset{R}{\leftarrow} \mathbb{Z}_q$ and $Q \overset{R}{\leftarrow} G_T$, for any probabilistic polynomial algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ determining $Q \overset{?}{=} e(g,g)^{xyz}$ is negligible to security parameter ℓ_1 at most, where the advantage is defined as

$$|\mathsf{Pr}[\mathcal{A}(g, g^x, g^y, g^z, e(g,g)^{xyz}) = 1] - \mathsf{Pr}[\mathcal{A}(g, g^x, g^y, g^z, Q) = 1]|.$$

Squared Decisional Bilinear Diffie-Hellman (SDBDH) [25] Given (g, g^x, g^y, Q) where $x, y \overset{R}{\leftarrow} \mathbb{Z}_q$ and $Q \overset{R}{\leftarrow} G_T$, for any probabilistic polynomial algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ determining $Q \overset{?}{=} e(g,g)^{x^2y}$ is negligible to security parameter ℓ_1 at most, where the advantage is defined as

$$|\mathsf{Pr}[\mathcal{A}(g, g^x, g^y, e(g,g)^{x^2y}) = 1] - \mathsf{Pr}[\mathcal{A}(g, g^x, g^y, Q) = 1]|.$$

3.2 Our Construction

The anonymous and CCA secure PRE scheme can be constructed as follows:

setup(1^ℓ): Given the security parameter ℓ, it obtains two secondary security parameters ℓ_1, ℓ_2, and instantiates H_0, H_1, H_2, H_3, H_4, F_1, F_2 and the bilinear map (q, g, G, G_T, e) as in section 3.1, where q is a ℓ_1-bit prime and the message $\mathsf{m} \in \{0,1\}^{\ell_2}$. In addition, let $g_1 \overset{R}{\leftarrow} G$ and set

$$\mathsf{Param} = \{q, g, g_1, G, G_T, e\}.$$

KeyGen(i): For use i, this algorithm selects $a_i \overset{R}{\leftarrow} \mathbb{Z}_q$ and sets

$$\mathsf{sk}_i = a_i, \mathsf{pk}_i = g^{a_i}.$$

ReKeyGen$(\mathsf{sk}_i, \mathsf{pk}_j)$: This algorithm generates the re-encryption key by selecting $s, t, w \overset{R}{\leftarrow} \mathbb{Z}_q$ and setting $\mathsf{rk}_{i\to j} = (\mathsf{rk}_1, \mathsf{rk}_2, \mathsf{rk}_3, \mathsf{rk}_4, \mathsf{rk}_5, \mathsf{rk}_6)$ as

$$(H_0(\mathsf{pk}_i)^{-\mathsf{sk}_i} \cdot \mathsf{pk}_j^{H_4(s\cdot\mathsf{sk}_i)},\ g^{H_4(s\cdot\mathsf{sk}_i)} g_1^t,\ g^t,\ g^w, e(\mathsf{pk}_j, \mathsf{pk}_j)^{tw},\ e(\mathsf{pk}_j, g)^{tw})$$

Note that rk_5 cannot be altered due to unknown t and w. Thus, the re-encryption key rk cannot be manipulated (Manipulating $\mathsf{rk}_5, \mathsf{rk}_6$ will be detected in Dec when decrypting first level ciphertexts.)

Enc$(\mathsf{pk}_i, \mathsf{m})$: Given $\mathsf{m} \in \{0,1\}^{\ell_2}$ and pk_i, this algorithm generates a second level ciphertext by selecting $R \xleftarrow{R} G_T$, computing $r = H_1(\mathsf{m}, R)$, and setting $\mathsf{C} = (\mathsf{C}_1, \mathsf{C}_2, \mathsf{C}_3, \mathsf{C}_4, \mathsf{C}_5)$ as

$$(g^r, \quad R \cdot e(\mathsf{pk}_i, H_0(\mathsf{pk}_i))^r, \quad \mathsf{m} \bigoplus F_1(R), \quad g_1^r, \quad H_2(\mathsf{C}_1, \mathsf{C}_2, \mathsf{C}_3, \mathsf{C}_4)^r)$$

Here we apply the Fujisaki-Okamoto (FO) transformation [13] to generate C.

ReEnc$(\mathsf{rk}_{i\to j}, \mathsf{C})$: This algorithm first checks

$$e(g, \mathsf{C}_5) \stackrel{?}{=} e(\mathsf{C}_1, H_2(\mathsf{C}_1, \mathsf{C}_2, \mathsf{C}_3, \mathsf{C}_4)).$$

If the equation does not hold, it aborts; otherwise it re-encrypts $\mathsf{C} = (\mathsf{C}_1, \mathsf{C}_2, \mathsf{C}_3, \mathsf{C}_4, \mathsf{C}_5)$ to a first level ciphertext under pk_j as follows:

- Let $T_1 = \mathsf{C}_1 = g^r$, $T_2 = \mathsf{C}_2 \cdot e(\mathsf{C}_1, \mathsf{rk}_1) = R \cdot e(g^r, \mathsf{pk}_j^{H_4(s\cdot \mathsf{sk}_i)})$, $T_3 = \mathsf{C}_3 = \mathsf{m} \bigoplus F_1(R)$ and $T_4 = \mathsf{C}_4 = g_1^r$, and set $\Gamma = T_1||T_2||T_3||T_4||\mathsf{rk}_2||\mathsf{rk}_3||\mathsf{rk}_4$.
- Select $R' \xleftarrow{R} G_T$ and let $r' = H_3(\Gamma, R')$, and set a first level ciphertext $\mathsf{C}' = (\mathsf{C}'_1, \mathsf{C}'_2, \mathsf{C}'_3)$ as

$$(\mathsf{rk}_6^{r'}, R'\mathsf{rk}_5^{r'}, \Gamma \bigoplus F_2(R'))$$

The intermediate ciphertext (T_1, T_2, T_3) cannot be manipulated because r, m, R are correlated with each other and the relation among them will be verified in decryption. It is worth noting that we apply FO transformation again to generate C' in order to achieve the CCA security for the first level ciphertext.

Dec: It decrypts the ciphertext correspondingly:

- Given a second level ciphertext $\mathsf{C} = (\mathsf{C}_1, \mathsf{C}_2, \mathsf{C}_3, \mathsf{C}_4, \mathsf{C}_5)$, it decrypts with $\mathsf{sk}_i = a_i$:
  1. If $e(g, \mathsf{C}_5) \neq e(\mathsf{C}_1, H_2(\mathsf{C}_1, \mathsf{C}_2, \mathsf{C}_3, \mathsf{C}_4))$, then output $\perp$;
  2. Otherwise let $R = \mathsf{C}_2 / e(g^r, H_0(\mathsf{pk}_i))^{\mathsf{sk}_i}$, and have $\mathsf{m} = \mathsf{C}_3 \bigoplus F_1(R)$;
- Given a first level ciphertext $\mathsf{C}' = (\mathsf{C}'_1, \mathsf{C}'_2, \mathsf{C}'_3)$, it decrypts with $\mathsf{sk}_j = a_j$:
  1. Compute $R' = \mathsf{C}'_2 / \mathsf{C}'^{\mathsf{sk}_j}_1$, let $\Gamma = \mathsf{C}'_3 \bigoplus F_2(R')$, and parse $\Gamma = T_1||T_2||T_3||T_4||\mathsf{rk}_2||\mathsf{rk}_3||\mathsf{rk}_4$.
  2. If $e(\mathsf{rk}_3, \mathsf{rk}_4)^{\mathsf{sk}_j H_3(\Gamma, R')} \neq \mathsf{C}'_1$, then return $\perp$. This is to verify rk_5 and rk_6 without being manipulated.
  3. Let $R = T_2 (\frac{e(\mathsf{rk}_3, T_4)}{e(\mathsf{rk}_2, T_1)})^{\mathsf{sk}_j}$ so that $\mathsf{m} = T_3 \bigoplus F_1(R)$.
  4. Let $r = H_1(\mathsf{m}, R)$ and verify $T_1 \stackrel{?}{=} g^r$. If so, return m; $\perp$ otherwise.

Correctness of the PRE scheme can be verified. In the construction, we can see that we use FO transformation twice to assure CCA security for the first and second level ciphertexts. In addition, The PRE scheme is collusion-safe [6] even that user j and the cloud collude together. Intuitively, user j cannot derive $\mathsf{pk}_j^{H_4(s\cdot \mathsf{sk}_i)}$ from the re-encryption key $\mathsf{rk}_{i\to j}$, so that he is unable to infer $H_0(\mathsf{pk}_i)^{-\mathsf{sk}_i}$ and decrypt user i's second level ciphertexts.

3.3 Security Analysis

We show the theorem regarding CCA security for second level ciphertexts as follows. Similarly, we can prove our scheme achieves CCA security for first level ciphertexts and omit it here.

THEOREM 1. *Assume that the DBDH assumption holds, the* PRE *scheme is* CCA *secure in the random oracle model for second level ciphertexts.*

PROOF. The challenger $\mathcal{C}$ maintains random oracles as follows:

- Oracle $\mathcal{O}_{H_0}$: If $H_0(\mathsf{pk})$ has been queried before, it retrieves β from $\mathcal{O}_{H_0}$ according to pk; otherwise it selects $\beta \xleftarrow{R} \mathbb{Z}_q$, lets $H_0(\mathsf{pk}) = (g^y)^\beta$, and adds (pk, β) into $\mathcal{O}_{H_0}$. It returns $(g^y)^\beta$.
- Oracles $\mathcal{O}_{H_1}$, $\mathcal{O}_{H_2}$, $\mathcal{O}_{H_3}$,$\mathcal{O}_{H_4}$, $\mathcal{O}_{F_1}$ and $\mathcal{O}_{F_2}$ are modeled as random oracles and we skip their details.

Suppose the challenger $\mathcal{C}$ is given a DBDH instance of (g, g^x, g^y, g^z, Q), where $x, y, z \xleftarrow{R} \mathbb{Z}_p$, and $Q \xleftarrow{R} G_T$, and x, y, z are unknown. $\mathcal{C}$ simulates the CCA-security game with $\mathcal{A}$ by letting $g_1 = g^c$, where $c \xleftarrow{R} \mathbb{Z}_q$ is known to $\mathcal{C}$, and proceeds the game as follows.

Phase 1: $\mathcal{A}$ can challenge the oracles in polynomially may times and $\mathcal{C}$ responds as follows:

$\mathcal{O}_{\mathsf{KeyGen}}(i)$: $\mathcal{C}$ responds as follows:

- $i \in \mathbb{U}_h$: If user i has been queried before, it retrieves pk_i from $\mathcal{O}_h$; otherwise $\mathcal{C}$ selects $a_i \xleftarrow{R} \mathbb{Z}_q$, lets $\mathsf{pk}_i = (g^x)^{a_i}$, and adds (i, pk_i, a_i) into $\mathcal{O}_h$. It returns pk_i to $\mathcal{A}$, where it implicitly defines $\mathsf{sk}_i = x \cdot a_i$.
- $i \in \mathbb{U}_e$: If user i has been queried before, it returns $(\mathsf{pk}_i, \mathsf{sk}_i)$ from $\mathcal{O}_e$; otherwise $\mathcal{C}$ selects $b_i \xleftarrow{R} \mathbb{Z}_q$, lets $\mathsf{pk}_i = g^{b_i}, \mathsf{sk}_i = b_i$, adds $(i, \mathsf{pk}_i, \mathsf{sk}_i)$ into $\mathcal{O}_e$, and returns $(\mathsf{pk}_i, \mathsf{sk}_i)$ to $\mathcal{A}$.

$\mathcal{O}_{\mathsf{ReKeyGen}}(\mathsf{pk}_i, \mathsf{pk}_j)$: $\mathcal{C}$ generates $\mathsf{rk}_{i\to j} = (\mathsf{rk}_1, \mathsf{rk}_2, \mathsf{rk}_3, \mathsf{rk}_4, \mathsf{rk}_5, \mathsf{rk}_6)$ for a pair of users i and j by selecting $s, t, w \xleftarrow{R} \mathbb{Z}_q$, and :

- If $i \in \mathbb{U}_e$, let $\mathsf{rk}_1 = H_0(\mathsf{pk}_i)^{-\mathsf{sk}_i} \mathsf{pk}_j^{H_4(s\cdot \mathsf{sk}_i)}$, $\mathsf{rk}_2 = g^{H_4(s\cdot \mathsf{sk}_i)} g_1^t$, $\mathsf{rk}_3 = g^t, \mathsf{rk}_4 = g^w, \mathsf{rk}_5 = e(\mathsf{pk}_j, \mathsf{pk}_j)^{tw}$ and $\mathsf{rk}_6 = e(\mathsf{pk}_j, g)^{tw}$. $\mathcal{C}$ returns $\mathsf{rk}_{i\to j}$ to $\mathcal{A}$.
- If $i \in \mathbb{U}_h$ and $j \in \mathbb{U}_h$, let $\mathsf{rk}_1 = \mathsf{pk}_i^\gamma$, $\mathsf{rk}_2 = (g^y)^{a_i\beta_i/a_j} g^{a_i\gamma/a_j} g_1^t$, $\mathsf{rk}_3 = g^t$, $\mathsf{rk}_4 = g^w$, $\mathsf{rk}_5 = e(\mathsf{pk}_j, \mathsf{pk}_j)^{tw}$ and $\mathsf{rk}_6 = e(\mathsf{pk}_j, g)^{tw}$, where $\gamma \xleftarrow{R} \mathbb{Z}_p$, then $\mathcal{C}$ returns $\mathsf{rk}_{i\to j}$ to $\mathcal{A}$. Note that $\mathsf{rk}_{i\to j}$ is a valid re-encryption key. Let us implicitly define $H_4(s \cdot \mathsf{sk}_i) = (a_i\beta_i y + \gamma a_i)/a_j$ (note s, sk_i are unknown to $\mathcal{A}$) so that

$$\begin{aligned} \mathsf{rk}_1 &= \mathsf{pk}_i^\gamma = g^{-y\beta_i x a_i + y\beta_i x a_i + \gamma a_i x} \\ &= (g^y)^{-\beta_i a_i x} g^{a_j x \frac{1}{a_j}(y\beta_i a_i + \gamma a_i)} \\ &= H_0(\mathsf{pk}_i)^{-a_i x} \mathsf{pk}_j^{H_4(s\cdot \mathsf{sk}_i)} \end{aligned}$$

- Otherwise returns $\perp$.

$\mathcal{O}_{\mathsf{ReEnc}}(\mathsf{pk}_i, \mathsf{pk}_j, \mathsf{C})$: Given a second level ciphertext $\mathsf{C} = (\mathsf{C}_1, \mathsf{C}_2, \mathsf{C}_3, \mathsf{C}_4, \mathsf{C}_5)$, $\mathcal{C}$ does:

- If $e(g, \mathsf{C}_5) \neq e(\mathsf{C}_1, H_3(\mathsf{C}_1, \mathsf{C}_2, \mathsf{C}_3, \mathsf{C}_4))$ or $i \in \mathbb{U}_h$ and $j \in \mathbb{U}_e$, output $\perp$.
- Query $\mathcal{O}_{\mathsf{ReKeyGen}}(\mathsf{pk}_i, \mathsf{pk}_j)$ to obtain $\mathsf{rk}_{i\to j} = (\mathsf{rk}_1, \mathsf{rk}_2, \mathsf{rk}_3, \mathsf{rk}_4, \mathsf{rk}_5, \mathsf{rk}_6)$.
- Execute ReEnc by taking as input C and $\mathsf{rk}_{i\to j}$, and output $\mathsf{C}' = (\mathsf{C}'_1, \mathsf{C}'_2, \mathsf{C}'_3)$.

$\mathcal{O}_{\mathsf{Dec}}(\mathsf{pk}_i, C)$: To decrypt C for user i, $\mathcal{C}$ does:

- If $i \in \mathbb{U}_e$, run algorithm Dec by taking as input the secret key b_i and the ciphertext C, and return the output.
- If C is a second level ciphertext, i.e., $\mathsf{C} = (\mathsf{C}_1, \mathsf{C}_2, \mathsf{C}_3, \mathsf{C}_4, \mathsf{C}_5)$, verify $e(g, \mathsf{C}_5) \stackrel{?}{=} e(\mathsf{C}_1, H_3(\mathsf{C}_1, \mathsf{C}_2, \mathsf{C}_3, \mathsf{C}_4))$. If not hold, then output $\perp$; otherwise search $\mathcal{O}_{H_1}$ for the tuple (m, R, r) where $r = H_1(\mathsf{m}, R)$ and the oracle $\mathcal{O}_{F_1}$, such that $\mathsf{C}_1 = g^r$ and $\mathsf{C}_3 = \mathsf{m} \bigoplus F_1(R)$. If such tuple exists, then return m; otherwise return $\perp$.
- If C is a first level ciphertext, i.e., $\mathsf{C}' = (\mathsf{C}'_1, \mathsf{C}'_2, \mathsf{C}'_3)$, search $\mathcal{O}_{H_3}$ for the tuple (Γ, R', r'), where $r' = H_3(\Gamma, R')$ and $\mathcal{O}_{F_2}$, such that $\mathsf{C}'_3 = \Gamma \bigoplus F_2(R')$. Then, parse Γ as $T_1, T_2, T_3, T_4, \mathsf{rk}_2, \mathsf{rk}_3, \mathsf{rk}_4$. Search $\mathcal{O}_{H_1}$ for the tuple (m, R, r), where $r = H_1(\mathsf{m}, R)$ and the oracle $\mathcal{O}_{F_1}$, such that $T_1 = g^r$ and $T_3 = \mathsf{m} \bigoplus F_1(R)$. If such tuple exists, return m, else return $\perp$.

Challenge: Once $\mathcal{A}$ decides that Phase 1 is finished, it outputs a public key pk^* and two messages $\mathsf{m}_0, \mathsf{m}_1$ of the same length, and sends them to $\mathcal{C}$. $\mathcal{C}$ returns $(\mathsf{C}^*_1, \mathsf{C}^*_2, \mathsf{C}^*_3, \mathsf{C}^*_4, \mathsf{C}^*_5)$ as follows: select $R^* \stackrel{R}{\leftarrow} G_T$ and let $\mathsf{C}^*_1 = g^z$, $\mathsf{C}^*_2 = R^* Q^{a^*\beta^*}$, $\mathsf{C}^*_3 = \mathsf{m}_\lambda \bigoplus F_1(R^*)$, $\mathsf{C}^*_4 = g_1^z = (g^z)^c$ and $\mathsf{C}^*_5 = (g^z)^{\theta^*}$, where $\theta^* \stackrel{R}{\leftarrow} \mathbb{Z}_q$.

If $Q = e(g,g)^{xyz}$, $(\mathsf{C}^*_1, \mathsf{C}^*_2, \mathsf{C}^*_3, \mathsf{C}^*_4, \mathsf{C}^*_5)$ is a valid ciphertext for m_λ under pk^*. To see it, we implicitly let $H_3(\mathsf{C}^*_1, \mathsf{C}^*_2, \mathsf{C}^*_3, \mathsf{C}^*_4) = g^{\theta^*}$, and have

$$
\begin{aligned}
\mathsf{C}^*_1 &= g^z, \\
\mathsf{C}^*_2 &= R^* Q^{a^*\cdot\beta^*} = R^* e((g^x)^{a^*}, (g^y)^{\beta^*})^z = R^* e(\mathsf{pk}^*, H_0(\mathsf{pk}^*))^z, \\
\mathsf{C}^*_3 &= \mathsf{m}_\lambda \bigoplus F_1(R^*), \\
\mathsf{C}^*_4 &= (g^z)^c = g_1^z, \\
\mathsf{C}^*_5 &= (g^z)^{\theta^*} = (g^{\theta^*})^z = H_3(\mathsf{C}^*_1, \mathsf{C}^*_2, \mathsf{C}^*_3, \mathsf{C}^*_4)^z,
\end{aligned}
$$

where $H_0(\mathsf{pk}^*) = (g^y)^{\beta^*}$, $\mathsf{pk}^* = (g^x)^{a^*}$.

Phase 2: $\mathcal{C}$ and $\mathcal{A}$ proceed the same as in Phase 1 while complying with constraints defined in the CCA-security game.

Guess: Finally, $\mathcal{A}$ returns a guess $\lambda' \in \{0, 1\}$. If $\lambda' = \lambda$, $\mathcal{C}$ outputs $Q = e(g,g)^{xyz}$; otherwise, $\mathcal{C}$ returns $Q \neq e(g,g)^{xyz}$.

This completes the simulation. In Phase 2, if $Q = e(g,g)^{xyz}$, then the ciphertext $(\mathsf{C}^*_1, \mathsf{C}^*_2, \mathsf{C}^*_3, \mathsf{C}^*_4, \mathsf{C}^*_5)$ is a valid ciphertext of m_λ, so the probability of $\mathcal{A}$ outputting $\lambda' = \lambda$ is $\frac{1}{2} + \mu$; if Q is an element randomly that is selected from G_T and independent from z, then the probability of $\mathcal{A}$ outputting $\lambda' = \lambda$ is $\frac{1}{2}$, because the ciphertext has no information about m_λ. Therefore, the probability of $\mathcal{C}$ correctly guessing $Q \stackrel{?}{=} e(g,g)^{xyz}$ with the instance of (g, g^x, g^y, g^z, Q) is $\frac{1}{2}(\frac{1}{2} + \mu + \frac{1}{2}) = \frac{1}{2} + \frac{\mu}{2}$.

□

Theorem 2 states that our scheme achieves our anonymity notion for re-encryption keys. Correspondingly, we can show that our scheme also achieves the anonymity for second level ciphertexts and first level ciphertexts, which are skipped here.

THEOREM 2. *Assume that SDBDH assumption holds, our scheme achieves the anonymity for re-encryption keys in Definition 3 in the random oracle model.*

The challenger $\mathcal{C}$ maintains random oracles as follows:

- Oracle $\mathcal{O}_{H_0}$: if $H_0(\mathsf{pk})$ has been queried, it retrieves $H_0(\mathsf{pk})$ from $\mathcal{O}_{H_0}$; otherwise select $\beta \stackrel{R}{\leftarrow} \mathbb{Z}_q$, let $H_0(\mathsf{pk}) = g^\beta$, and add (pk, β) into $\mathcal{O}_{H_0}$. It returns g^β.
- Oracles $\mathcal{O}_{H_1}$, $\mathcal{O}_{H_2}$, $\mathcal{O}_{H_3}$, $\mathcal{O}_{H_4}$, $\mathcal{O}_{F_1}$ and $\mathcal{O}_{F_2}$ are modeled as random oracles and we skip their details here.

The proof strategy is that $\mathcal{C}$ randomly selects an uncorrupted user J from the set of uncorrupted users and lets J be the targeted user in the Challenge phase. That is, if $j^* = J$ in the Challenge phase, the challenger will proceed the $\mathsf{anonymous}$ game; otherwise abort.

Proof of Theorem 2 Suppose $\mathcal{A}$ breaks the $\mathsf{anonymous}$ game with the probability $\frac{1}{2} + \mu$, then we can construct the challenger $\mathcal{C}$ solving $Q \stackrel{?}{=} e(g,g)^{x^2 y}$ for the instance (g, e, g^x, g^y, Q) with probability $\frac{1}{2} + \frac{\mu}{2n}$ at least, where n is the number of users in the PRE scheme, i.e., $|\mathbb{U}_h| + |\mathbb{U}_e| = n$.

Recall that $\mathcal{C}$ maintains a key list storing re-encryption keys for (i, j): $(i \to j, \{(\mathsf{rk}_{i\to j}, query)\})$. Assume $\mathcal{C}$ selects user J before proceeding the game. Given an instance (g, e, g^x, g^y, Q), $\mathcal{C}$ selects $c \stackrel{R}{\leftarrow} \mathbb{Z}_q$, lets $g_1 = g^c$, and proceeds the $\mathsf{anonymous}$ game to determine $Q \stackrel{?}{=} e(g,g)^{x^2 y}$ as follows.

Phase 1: $\mathcal{A}$ queries oracles below in polynomially many time:

$\mathcal{O}_{\mathsf{KeyGen}}(i)$: $\mathcal{C}$ responds as follows:

- $i \in \mathbb{U}_h$: If user i has been queried before, $\mathcal{C}$ retrieves pk_i from $\mathcal{O}_h$; otherwise (i) $i = J$, $\mathcal{C}$ selects $a_i \stackrel{R}{\leftarrow} \mathbb{Z}_q$, lets $\mathsf{pk}_i = (g^x)^{a_i}$, and adds (i, a_i, pk_i) into $\mathcal{O}_h$, (ii) $i \neq J$, $\mathcal{C}$ selects $a_i \stackrel{R}{\leftarrow} \mathbb{Z}_q$, lets $\mathsf{pk}_i = g^{a_i}$, and adds (i, a_i, pk_i) into $\mathcal{O}_h$. $\mathcal{C}$ returns pk_i to $\mathcal{A}$. Note that we explicitly define $\mathsf{sk}_i = x \cdot a_i$ when $i = J$.
- $i \in \mathbb{U}_e$: If the query has been queried before, then $\mathcal{C}$ returns $(\mathsf{pk}_i, \mathsf{sk}_i)$ from $\mathcal{O}_e$; else $\mathcal{C}$ selects $a_i \stackrel{R}{\leftarrow} \mathbb{Z}_q$, lets $\mathsf{pk}_i = g^{a_i}, \mathsf{sk}_i = a_i$, and adds $(i, \mathsf{pk}_i, \mathsf{sk}_i)$ into $\mathcal{O}_e$. It returns $(\mathsf{pk}_i, \mathsf{sk}_i)$ to $\mathcal{A}$.

$\mathcal{O}_{\mathsf{ReKeyGen}}(\mathsf{pk}_i, \mathsf{pk}_j)$: Given $\mathsf{pk}_i, \mathsf{pk}_j$, $\mathcal{C}$ obtains $(i \to j, \{(\mathsf{rk}_{i\to j}, query)\})$ and

- If $\{(\mathsf{rk}_{i\to j}, query)\}$ is $\mathtt{null}$ or all $\mathsf{rk}_{i\to j}$ have been queried before (i.e., all indicators $query = 1$), then
  1. If $i = J$, it generates $\mathsf{rk}_{J\to j}$ by selecting $s, w, v \stackrel{R}{\leftarrow} \mathbb{Z}_q$ and letting: $\mathsf{rk}_1 = (g^x)^{-a_i \beta_i} \mathsf{pk}_j^\gamma, \mathsf{rk}_2 = g^\gamma g_1^t, \mathsf{rk}_3 = g^t, \mathsf{rk}_4 = g^w, \mathsf{rk}_5 = e(\mathsf{pk}_j, \mathsf{pk}_j)^{tw}, \mathsf{rk}_6 = e(\mathsf{pk}_j, g)^{tv}$, where $\gamma \stackrel{R}{\leftarrow} \mathbb{Z}_q$. We implicitly define $H_4(s \cdot \mathsf{sk}_i) = \gamma$, and we can see $\mathsf{rk}_{i\to j}$ is a valid re-encryption key.

2. Otherwise let $\mathsf{rk}_{i\to j} = (\mathsf{rk}_1, \mathsf{rk}_2, \mathsf{rk}_3, \mathsf{rk}_4, \mathsf{rk}_5, \mathsf{rk}_6)$ as

$$(H_0(\mathsf{pk}_i)^{-\mathsf{sk}_i} \cdot \mathsf{pk}_j^{H_4(s\cdot\mathsf{sk}_i)},\ g^{H_4(s\cdot\mathsf{sk}_i)} g_1^t,\ \ g^t,\ \ g^w,\ e(\mathsf{pk}_j, \mathsf{pk}_j)^{tw},\ \ e(\mathsf{pk}_j, g)^{tw})$$

It adds $(\mathsf{rk}_{i\to j}, query = 1)$ to $\{(\mathsf{rk}_{i\to j}, query)\}$ and returns $\mathsf{rk}_{i\to j}$;

- Else, it chooses $\mathsf{rk}_{i\to j}$ randomly from $\{(\mathsf{rk}_{i\to j}, query)\}$ where $query = 0$, sets $query = 1$ correspondingly, and returns $\mathsf{rk}_{i\to j}$.

$\mathcal{O}_{\mathsf{ReEnc}}(\mathsf{pk}_i, \mathsf{pk}_j, \mathsf{C})$: Given a second level ciphertext $\mathsf{C} = (\mathsf{C}_1, \mathsf{C}_2, \mathsf{C}_3, \mathsf{C}_4, \mathsf{C}_5)$, $\mathcal{C}$ proceeds as follows:

- If $e(g, \mathsf{C}_5) \neq e(\mathsf{C}_1, H_3(\mathsf{C}_1, \mathsf{C}_2, \mathsf{C}_3, \mathsf{C}_4))$, $\mathcal{C}$ outputs $\perp$.
- If the set $\{i \to j, (\mathsf{rk}_{i\to j}, query)\}$ is `null`, $\mathcal{C}$ generates a re-encryption key $\mathsf{rk}_{i\to}$ as in $\mathcal{O}_{\mathsf{ReKeyGen}}$, and adds $(\mathsf{rk}_{i\to}, query = 0)$ to the key list; otherwise, $\mathcal{C}$ selects $\mathsf{rk}_{i\to j}$ randomly from the set $\{i \to j, (\mathsf{rk}_{i\to j}, query)\}$.
- $\mathcal{C}$ applies the algorithm ReEnc by taking input C and $\mathsf{rk}_{i\to j}$, and outputs $\mathsf{C}' = (\mathsf{C}'_1, \mathsf{C}'_2, \mathsf{C}'_3)$.

$\mathcal{O}_{\mathsf{Dec}}(\mathsf{pk}_i, \mathsf{C})$: Given pk_i and C, $\mathcal{C}$ does:

- If $i \neq J$, $\mathcal{C}$ applies the algorithm Dec by taking input the private key a_i and the ciphertext C, and returns the output.
- If $i = J$ and C is a second level ciphertext, i.e., $\mathsf{C} = (\mathsf{C}_1, \mathsf{C}_2, \mathsf{C}_3, \mathsf{C}_4, \mathsf{C}_5)$, $\mathcal{C}$ verifies $e(g, \mathsf{C}_5) \stackrel{?}{=} e(\mathsf{C}_1, H_3(\mathsf{C}_1, \mathsf{C}_2, \mathsf{C}_3, \mathsf{C}_4))$. If it does not hold, $\mathcal{C}$ outputs $\perp$; else $\mathcal{C}$ searches $\mathcal{O}_{H_1}$ for the tuple (m, R, r), where $r = H_1(\mathsf{m}, R)$ and $\mathcal{O}_{F_1}$, such that $\mathsf{C}_1 = g^r$ and $\mathsf{C}_3 = \mathsf{m} \bigoplus F_1(R)$. If such tuple exists, $\mathcal{C}$ returns m; else returns $\perp$.
- If $i = J$ and C is a first level ciphertext, i.e., $\mathsf{C}' = (\mathsf{C}'_1, \mathsf{C}'_2, \mathsf{C}'_3)$, $\mathcal{C}$ searches $\mathcal{O}_{H_3}$ for the tuple (Γ, R', r'), where $r' = H_3(\Gamma, R')$, and $\mathcal{O}_{F_2}$, such that $\mathsf{C}'_3 = \Gamma \bigoplus F_2(R')$. Then, $\mathcal{C}$ parses Γ as $T_1, T_2, T_3, T_4, \mathsf{rk}_2, \mathsf{rk}_3, \mathsf{rk}_4$, and searches $\mathcal{O}_{H_1}$ for the tuple (m, R, r), where $r = H_1(\mathsf{m}, R)$, and $\mathcal{O}_{F_1}$, such that $T_1 = g^r$ and $T_3 = \mathsf{m} \bigoplus F_1(R)$. If such tuple exists, $\mathcal{C}$ returns m; else returns $\perp$.

Challenge: Once $\mathcal{A}$ decides that Phase 1 is finished, it outputs (i^*, j^*), where $i^*, j^* \in \mathbb{U}_h$. If $j^* \neq J$, $\mathcal{C}$ aborts; otherwise $\mathcal{C}$ has $\mathsf{rk}^* = (\mathsf{rk}_1^*, \mathsf{rk}_2^*, \mathsf{rk}_3^*, \mathsf{rk}_4^*, \mathsf{rk}_5^*, \mathsf{rk}_6^*)$:

$$(H_0(\mathsf{pk}_i)^{-\mathsf{sk}_i} \cdot \mathsf{pk}_J^{H_4(s\cdot\mathsf{sk}_i)},\ g^{H_4(s\cdot\mathsf{sk}_i)} g^{yc},\ \ g^y,\ \ g^w,\ Q^{a_J^2 w},\ \ e(\mathsf{pk}_J, g^y)^w)$$

$\mathcal{C}$ adds $(\mathsf{rk}^*, query = 1)$ to the set of $\{(\mathsf{rk}^*, query)\}$. $\mathcal{C}$ selects $\lambda \stackrel{R}{\leftarrow} \{0, 1\}$, and delivers rk^* to $\mathcal{A}$ if $\lambda = 0$, or a random key from re-encryption key space if $\lambda = 1$.

Note that if $Q = e(g, g)^{x^2 y}$, rk^* is a valid re-encryption key for (i^*, j^*) since $\mathsf{rk}_5^* = e(g, g)^{x^2 y w a_J^2} = e(g^{a_J x}, g^{a_J x})^{yw} = e(\mathsf{pk}_J, \mathsf{pk}_J)^{yw}$.

Phase 2: $\mathcal{C}$ proceeds the game as that in the Phase 1 by following the constraints specified in the $\mathsf{anonymous}$ game.

Guess: Finally, $\mathcal{A}$ returns a guess $\lambda' \in \{0, 1\}$. If $\lambda' = \lambda$, $\mathcal{C}$ outputs $Q = e(g, g)^{x^2 y}$; $\mathcal{C}$ returns $Q \neq e(g, g)^{x^2 y}$.

This completes the simulation. Note $\mathsf{rk}_1^*, \mathsf{rk}_2^*, \mathsf{rk}_3^*$ are distributed uniformly and independently because of independent and random pk_j, $H_4(s\mathsf{sk}_i)$ and c. If $Q = e(g, g)^{x^2 y}$ (meaning $\mathsf{rk}_{i^*\to j^*}$ is a valid re-encryption key of (i^*, j^*)), $\mathcal{A}$ will output $\lambda' = \lambda$ with the probability of $\frac{1}{2} + \mu$; if Q is a random element in G_T independent from $e(g, g)^{x^2 y}$, $\mathcal{A}$ will output $\lambda' = \lambda$ with the probability of $\frac{1}{2}$ since the challenged re-encryption key reveals no information about (i^*, j^*). Hence, the probability of $\mathcal{C}$'s correctly guessing $Q \stackrel{?}{=} e(g, g)^{x^2 y}$ with the instance of (g, e, g^x, g^y, Q) is $\frac{1}{2}(\frac{1}{2} + \frac{\mu}{|\mathbb{U}_h|} + \frac{1}{2}) \geq \frac{1}{2} + \frac{\mu}{2n}$, because the probability of $\mathcal{C}$ proceeding the game is $\frac{1}{|\mathbb{U}_h|}$. Therefore, if $\mathcal{A}$ can break the $\mathsf{anonymous}$ game with a probability of $\frac{1}{2} + \mu$, we can solve the SDBDH problem at least with a probability of $\frac{1}{2} + \frac{\mu}{2n}$.

4. PERFORMANCE EVALUATION

We implement the proposed PRE scheme co-operating with Amazon Web Services (AWS) [1], an Infrastructure-as-a-Service (IaaS) cloud platform. In particularly, we deploy the proxy re-encryption as a service in Amazon EC2, provide simple API for application integration, and adopt Amazon S3 as storage server for ciphertexts. As a reference application, we develop a secure email forwarding application based on it and demonstrate its efficiency and feasibility.

4.1 Implementation

We implement the PRE in C language based on the Pairing-based Cryptography (PBC) library [4]. We instantiate the random oracles with OpenSSL/SHA256 and the converting function `element_from_hash` of PBC, and let each data block with length of 64 Bytes, e.g. $\ell_2 = 512$. The bilinear map is instantiated with two different parameters:

- `a.param`: The group order is 160 bits long, and the order of the base field is 512 bits long. It offers a level of security that is equivalent to 1024-bit DLOG. The operations is more efficient compared with other kinds of parameters [4].
- `e.param`: The group order is 160 bits long, and the order of the base field is 1024 bits long. It also offers a level of security equivalent to 1024-bit DLOG, but the specified pairing is not as efficient as that of `a.param`.

4.2 Asymptotic Complexity Comparison

Table 1 describes the asymptotic complexity for the proposed PRE scheme, and summarizes the security properties with the three anonymous schemes in the state of the art. We can see that the scheme [5] achieves CPA security and the scheme [19] achieves CCA security, both in the random oracle model. However, both schemes cannot achieve the anonymity as we defined. While the scheme [20] is proved to achieve the anonymity property as their defined, we find that their scheme does satisfy the anonymity notion we defined in the present paper. Unfortunately, [15] proved that the scheme [20] cannot attain CCA security as they claimed due to the attack [15]. In contrast, the proposed scheme in this paper not only achieves CCA security in the random oracle model, but also attains the desired anonymity property.

4.3 Efficiency of PRE Operations

To evaluate the efficiency of algorithms in the proposed PRE scheme, we run experiments on a machine with Linux

	Scheme [5]	Scheme [19]	Scheme [20]	Scheme in this Paper
Property				
Security Model	RO	RO		RO
Semantic Security	CPA	CCA	[CCA]*	CCA
Satisfying our anonymity	No	No	Yes	Yes
Bilinear Map?	Yes	No	Yes	Yes
Assumption	DBDH, Decision Linear	DDH, CCA-secury SKE	5-EDBDH, DDH CCA-secure SKE Strongly unforgeable SIG	DBDH
Complexity				
ReKeyGen	1P +4E	2E+ 1Enc	11E	2P +8E
Enc	3E	4E+ 1Enc	1P+ 5E+ 1ENC+ 1SIGN	1P +4E
ReEnc	2P+2E	2E	5P + 6E + 1VER	3P +2E
Dec (2nd level ciphertext)	1P + 1E	3E + 1DEC	5P+ 2E+ 1DEC +1VER	3P +1E
Dec (1st level ciphertext)	1E	2E+ 1DEC	7E+ 1 DEC	3P +4E

Table 1: Comparison between prior schemes and ours, where SKE denotes a one time symmetric encryption scheme, SIG denotes a one time signature scheme, P denotes a pairing operation, E denotes an exponentiation operation, ENC denotes an encryption operation of SKE, DEC denotes an decryption operation of SKE, SIGN denotes a signing operation of SIG, and VER denotes a verification operation of SIG. * The scheme [20] is not CCA secure shown in [15] (CT-RSA'13).

OS, Intel Duo CPU E7500 2.93GHz and 2GB RAM. We average the execution time by running each experiment 20 times and show them in Table 4.3. We can see that ReKeyGen is most costly compared with other algorithms. Fortunately, this operation is executed infrequently. We observe that the performance of the PRE scheme is quite acceptable in practice with the bilinear map specified by `a.param`, since the operations only require around 10-14 milliseconds except the re-encryption key generation.

Operation	a.param (ms)	e.param (ms)
ReKeyGen	20.17	56.89
Enc	10.69	30.22
ReEnc	11.86	35.69
Dec(2nd level ciphertext)	10.79	32.47
Dec(1st level ciphertext)	12.37	39.49

Table 2: Efficiency of operations of the proposed PRE scheme.

4.4 Implementation of Proxy Service

To investigate the utilization of our PRE, we implement the proxy service on top of two AWS services: EC2 and S3, where EC2 provides a virtualized environment to launch virtual machine instances with optional computing resources (CPU frequency/core/RAM) and S3 is a cloud-based storage system to store data in buckets and provides management interface, i.e., writing, reading and deletion.

We implement the proxy service as a PHP web server running on an EC2 virtual machine where the bilinear map is specified by `a.param`. The proxy service provides a simple HTTP API ReEnc(`rk`, `s3-url2`, `s3-url1`), where `rk` is a re-encryption key, `s3-url2` is the URL of a second level ciphertext object in S3, and `s3-url1` is the URL of a first level ciphertext object (after re-encrypting `s3-url2`) in S3. When the re-encryption has been done, a successful HTTP response is sent out to notify that the operation is executed successfully and the first level ciphertext has been stored to the location specified by `s3-url1`. Note that the proxy service itself does not manage any content. If necessary, the proxy service should be granted permissions to read/write data objects in S3. In addition, we can easily extend our proxy service to support other cloud storage services such as SQL (e.g., RDS) and NoSQL data (e.g., SimpleDB).

4.5 Secure Email Forwarding with Anonymous Proxy Service

As a case study, we have developed a secure email forwarding application with our proxy service in AWS.

System overview: As shown in Figure 1, our application consists of three parts: (i) a web-based email client (email sender/receiver here) provides regular browser interface for users to login and send/read emails. To keep the email confidential, it employs our PRE scheme to perform encryption and decryption, together with necessary functions, i.e., downloading ciphertext from S3. (ii)A email server, which is to manage incoming emails and handle them, and (iii) the proxy service deployed in EC2.

A secure email forwarding typically takes the following steps. First, Alice sends to Bob the email in encryption form (encrypting email body, and optional attachment) via the email client (step 1). Upon receiving it, the email server will upload the encrypted email body to `s3-url2` (step 2), and invoke the proxy service via the API ReEnc(`rk, s3-url2, s3-url1`) with a re-encryption key so that the re-encrypted email can be decrypted by Carol (step 3). The proxy service will parse `rk`, fetch second level ciphertexts specified by `s3-url2` from S3, perform the re-encryption, write re-encrypted ciphertexts to the location specified by `s3-url1`, and return `HTTP OK` to the email server (step 4). Finally the email server will send to Carol the final email stored in the location specified by `s3-url1` (step 5), so that Carol can decrypt and read it with its email client (step 6).

We use open source Hastymail [3] as the email client, which is written in PHP. We extend Hastymail with three important plugins: encryption, decryption, and key management. The encryption plugin encrypts the email that will be sent out and the decryption plugin decrypts the encrypted emails (before or after re-encrypted) that are downloaded from S3. On the other hand, we use Apache JAMES [2] as the email server, which is responsible to upload the incom-

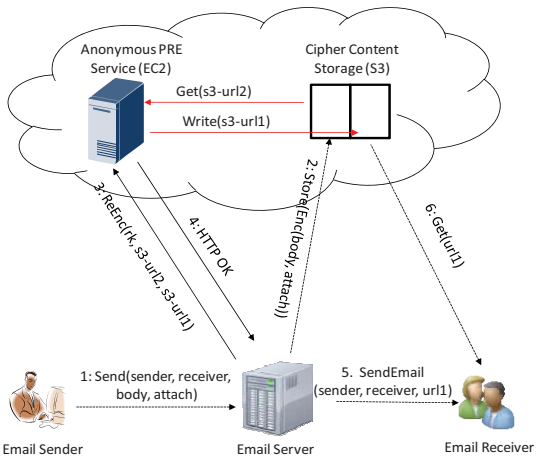

Figure 1: System overview of cloud-based email forwarding service with anonymous PRE.

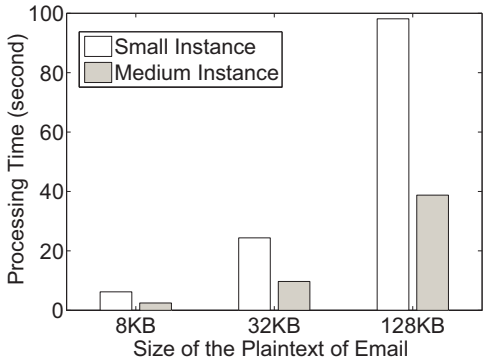

Figure 2: Average processing time for re-encryption in the cloud (one request).

ing email objects to S3, invoke the proxy service, and send (or broadcast) forwarding emails to appropriate receivers.

Experiment evaluation: We evaluate the system performance with two different EC2 instances:

- Small Instance: One core CPU of 1 ECU and 1.7GB memory;
- Medium Instance: Two core CPU of 5 ECUs and 1.7GB memory,

where an ECU provides the equivalent CPU capacity of 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor. In particular, we are interested in the execution time on the re-encryption operation by the proxy service.

We model the scenario that the proxy service receives 1,000 re-encryption requests simultaneously from the email server. The sizes of plaintexts for emails are 8KB, 32KB, and 128KB respectively, which are close to the size of emails in reality. we measure the execution time by the proxy service, including downloading second level ciphertexts from S3, performing re-encryption operations, and uploading first level ciphertexts to S3. Note here we do not encrypt plaintext with hybrid encryption methodology [14]: Encrypt the symmetric key with the PRE, and then encrypt plaintext with symmetric encryption. Instead we divide plaintexts into multiple blocks and encrypt each block individually.

As shown in Figure 2, we can see that the performance of the proxy service is acceptable. Taking the Medium Instance as an example, an email of 8KB can be re-encrypted around 2.3 second, and an email of 128KB around 40 seconds. Second, the computation capability has a positive influence on the re-encryption operations: the operation in the Medium Instance costs less time than that in the Small Instance.

Security properties: We use our anonymous PRE scheme to reduce the trust of cloud for email forwarding, which is ideal for large volume email sending, e.g., group email distribution. As an incoming email is encrypted with a user's (or an organization's) public key, the cloud based proxy service cannot access the encrypted content without its private key, which is stored in an isolated domain (Hastymail in our case). Furthermore, since users' private keys are maintained locally and re-encryption keys are generated out of cloud, the cloud cannot infer any identities information due to the anonymous property in our PRE scheme. Therefore, the identities of the re-encryption participants are kept private.

5. RELATED WORK

The cryptographic primitive of Proxy Re-Encryption (PRE) was formalized in [8] and has become an important building block for constructing cryptographic solutions to many application problems [6, 10, 11, 23, 24]. Various features of PRE have been investigated: e.g. [5] initiated the anonymity property, [9, 17] constructed CCA security PRE, [12] proposed efficient PRE construction and [21] enriched the PRE with conditional re-encryption feature.

The anonymity (key privacy) of PRE was first initiated in [5], where they assumed that only one re-encryption key existing for each pair of users. In addition, their proposed PRE scheme achieved security against chosen-plaintext attacks based on the DBDH assumption, and left constructing anonymous CCA-secure PRE as an open problem. Another anonymity notion came from the work [19], where allowed multiple re-encryption keys for each pair of users while lacking access to the re-encryption oracleïijN̆ which weakened the adversary's capability as explained in Section 1. Their proposed scheme [19] achieved their defined anonymity and enjoyed the security against chosen-ciphertext attack in the random oracle. Another work [20] claimed attaining the security against chosen-ciphertext attack and the anonymity property simultaneously, but later was proved not CCA secure.

In order to achieve chosen-ciphertext security, the work [9] first formalized the notion of security against chosen-ciphertext attacks for PRE and presented an efficient construction satisfying that. The subsequent work on achieving CCA security can be found in [12, 17, 22, 16, 18]. To achieve both anonymity and CCA security in PRE, the construction is quite difficult, since the CPA-to-CCA translations, such as the Fujisaki-Okamoto transformation cannot be applied directly [5, 20].

6. CONCLUSION

We consider the identity privacy problem for proxy re-encryption, which is useful when cloud users outsource re-encryption operations to public clouds. We propose a new

anonymity notion by comprehensively considering the limitations of existing anonymity notions. We then present an efficient construction, which achieves the anonymity and security against chosen-ciphertext attack in random oracle simultaneously. We implement and deploy our proposed scheme in Amazon Web Services, based on which we develop the prototype of secure email forwarding with anonymous re-encryption. Our evaluation shows that the performance is acceptable for many web-based applications. Our future work includes designing anonymous and CCA secure PRE in the standard model.

7. REFERENCES

[1] Amazon web service. `http://aws.amazon.com/`.

[2] Apache james. `http://projects.apache.org/projects/james.html`.

[3] Hastymail. `http://www.hastymail.org/`.

[4] Pairing-based cryptography library. `http://crypto.stanford.edu/pbc/`.

[5] G. Ateniese, K. Benson, and S. Hohenberger. Key-private proxy re-encryption. In *Proceedings of the The Cryptographers' Track at the RSA Conference 2009 on Topics in Cryptology*, CT-RSA '09, pages 279–294, Berlin, Heidelberg, 2009. Springer-Verlag.

[6] G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. volume 9, pages 1–30, New York, NY, USA, February 2006. ACM.

[7] M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-privacy in public-key encryption. In *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT '01, pages 566–582, London, UK, UK, 2001. Springer-Verlag.

[8] M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. In *In EUROCRYPT*, pages 127–144. Springer-Verlag, 1998.

[9] R. Canetti and S. Hohenberger. Chosen-ciphertext secure proxy re-encryption. In *Proceedings of the 14th ACM conference on Computer and communications security*, CCS '07, pages 185–194, New York, NY, USA, 2007. ACM.

[10] Y.-R. Chen, C.-K. Chu, W.-G. Tzeng, and J. Zhou. Cloudhka: A cryptographic approach for hierarchical access control in cloud computing. In *Proceedings of the 11th International Conference on Applied Cryptography and Network Security*, ACNS'13, pages 37–52, Berlin, Heidelberg, 2013. Springer-Verlag.

[11] Y.-R. Chen, J. D. Tygar, and W.-G. Tzeng. Secure group key management using uni-directional proxy re-encryption schemes. In *INFOCOM*, pages 1952–1960, 2011.

[12] R. H. Deng, J. Weng, S. Liu, and K. Chen. Chosen-ciphertext secure proxy re-encryption without pairings. In *Proceedings of the 7th International Conference on Cryptology and Network Security*, CANS '08, pages 1–17, Berlin, Heidelberg, 2008. Springer-Verlag.

[13] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '99, pages 537–554, London, UK, 1999. Springer-Verlag.

[14] D. Hofheinz and E. Kiltz. Secure hybrid encryption from weakened key encapsulation. In *CRYPTO*, pages 553–571, 2007.

[15] T. Isshiki, M. H. Nguyen, and K. Tanaka. Proxy re-encryption in a stronger security model extended from ct-rsa2012. In *CT-RSA*, pages 277–292, 2013.

[16] X. Jia, J. Shao, J. Jing, and P. Liu. Cca-secure type-based proxy re-encryption with invisible proxy. In *CIT*, pages 1299–1305, 2010.

[17] B. Libert and D. Vergnaud. Unidirectional chosen-ciphertext secure proxy re-encryption. In *Proceedings of the Practice and theory in public key cryptography, 11th international conference on Public key cryptography*, PKC'08, pages 360–379, Berlin, Heidelberg, 2008. Springer-Verlag.

[18] T. Matsuda, R. Nishimaki, and K. Tanaka. Cca proxy re-encryption without bilinear maps in the standard model. In *Proceedings of the 13th international conference on Practice and Theory in Public Key Cryptography*, PKC'10, pages 261–278, Berlin, Heidelberg, 2010. Springer-Verlag.

[19] J. Shao, P. Liu, G. Wei, and Y. Ling. Anonymous proxy re-encryption. volume 5, pages 439–449. John Wiley & Sons, Ltd, 2012.

[20] J. Shao, P. Liu, and Y. Zhou. Achieving key privacy without losing cca security in proxy re-encryption. In *Journal of Systems and Software*, 2011.

[21] J. Weng, R. H. Deng, X. Ding, C.-K. Chu, and J. Lai. Conditional proxy re-encryption secure against chosen-ciphertext attack. In *ASIACCS*, pages 322–332, 2009.

[22] J. Weng, Y. Yang, Q. Tang, R. H. Deng, and F. Bao. Efficient conditional proxy re-encryption with chosen-ciphertext security. In *Proceedings of the 12th International Conference on Information Security*, ISC '09, pages 151–166, Berlin, Heidelberg, 2009. Springer-Verlag.

[23] H. Xiong, X. Zhang, W. Zhu, and D. Yao. Cloudseal: End-to-end content protection in cloud-based storage and delivery services. In *SecureComm*, pages 491–500, 2011.

[24] K. Yang, X. Jia, and K. Ren. Attribute-based fine-grained access control with efficient revocation in cloud storage systems. In *ASIACCS*, pages 523–528, 2013.

[25] F. Zhang, R. Safavi-Naini, and W. Susilo. An efficient signature scheme from bilinear pairings and its applications. In *Public Key Cryptography*, pages 277–290, 2004.