9-2019

Puncturable proxy re-encryption supporting to group messaging service

Tran Viet Xuan PHUONG
University of Wollongong

Willy SUSILO
University of Wollongong

Jongkil KIM
University of Wollongong

Guomin YANG
Singapore Management University, gmyang@smu.edu.sg

Dongxi LIU
CSIRO

Puncturable Proxy Re-Encryption Supporting to Group Messaging Service

Tran Viet Xuan Phuong[1,2(✉)], Willy Susilo[1], Jongkil Kim[1], Guomin Yang[1], and Dongxi Liu[2]

[1] Institute of Cybersecurity and Cryptology School of Computing and Information Technology, University of Wollongong, Wollongong, Australia
{txuan,wsusilo,jongkil,gyang}@uow.edu.au
[2] Data61, CSIRO, Syndey, Australia
Dongxi.Liu@data61.csiro.au

Abstract. This work envisions a new encryption primitive for many-to-many paradigms such as group messaging systems. Previously, puncturable encryption (PE) was introduced to provide forward security for asynchronous messaging services. However, existing PE schemes were proposed only for one-to-one communication, and causes a significant overhead for a group messaging system. In fact, the group communication over PE can only be achieved by encrypting a message multiple times for each receiver by the sender's device, which is usually suitable to restricted resources such as mobile phones or sensor devices. Our new suggested scheme enables to re-encrypt ciphertexts of puncturable encryption by a message server (i.e., a proxy) so that computationally heavy operations are delegated to the server who has more powerful processors and a constant power source. We then proposed a new Puncturable Proxy Re-Encryption (PPRE) scheme. The scheme is inspired by unidirectional proxy re-encryption (UPRE), which achieves forward secrecy through fine-grained revocation of decryption capability by integrating the PE scheme. This paper first presents a forward secure PPRE in the group messaging service. Our scheme is IND-CCA secure under 3-weak Decision Bilinear Diffie-Hellman Inversion assumption.

Keywords: Puncturable encryption · Proxy Re-Encryption · Group messaging service · CCA security

1 Introduction

Green and Miers introduced Puncturable Encryption (PE) [15] to produce efficient forward-secure encryption for asynchronous communication with low overhead. Forward secrecy is a crucial trend on secure communication. For example, a new version of TLS v1.3 mandates forward secrecy for its key exchange. The protocols that do not support forward secrecy will be gradually deprecated in the near future. PE enables users to utilize forward secure asynchronous communication such as a messaging service.

K. Sako et al. (Eds.): ESORICS 2019, LNCS 11735, pp. 215–233, 2019.
https://doi.org/10.1007/978-3-030-29959-0_11

A group messaging service in real-world applications such as Snapchat and Whatsapp is essential since the communication between users is not always one-to-one. Many-to-many communication (e.g., a group messaging service) has a capability to boost customer convenience in the business/private conversation for a group of users. Therefore, supporting group messaging service makes a conversation more specific and focused. Forward secrecy and asynchronous properties that PE offers are still important in a group messaging service. However, messages in group communication are not always synchronized. Participants who are traveling or on-the-go will receive the messages with a significant delay. Furthermore, in the event that a user key in group communication is compromised; the confidentiality of past messages will fail.

In the existing work, the proposed PE schemes are constructed only for one-to-one communication between a sender and a receiver. How to use those PE schemes for many-to-many communication such as a group messaging service remains daunting. One of the most trivial ways is a participant of a group communication encrypts a message for all other participants in the communication one-by-one using PE, but this requires significant computation overhead to the sender. Particularly, if a message is sent from resource-constraint devices such as mobiles and sensor devices, this causes a substantial amount of battery consumption and delay as the number of participants grows.

To mitigate the delay time or support messaging for individuals who are away, we revisit the Proxy Re-Encryption (PRE) [2,3,17]. Suppose that Alice makes a group chatting room and invites multiple users, every time a new user joins in the group chatting, Alice computes the re-encryption key for this user by his public key and uploads to a messaging server, which is considered as a proxy. If anyone sends a message in this room, the message is encrypted only for Alice and send to the message server. The message server re-encrypts the encrypted message for all participants one-by-one using the re-encryption keys that Alice uploaded, then delivers it to the participants. Because the message server has more powerful processors and constant power source, it will reduce the delay caused by multiple encryptions. Moreover, each participant will encrypt the message only once as a general PE scheme; it prolongs the battery life of participant devices, significantly.

Contribution: Motivated by the aforementioned scenario, we investigate the fine-grained revocation of decryption capability only for specific messages while all other messages are decryptable in [15], then incorporate the unidirectional proxy re-encryption (UPRE) to firstly propose Puncturable Proxy Re-Encryption (PPRE). In a nutshell, PPRE scheme has both PE and UPRE scheme; however, it is not straightforward to deploy both schemes into the typical proxy re-encryption. At a high-level idea, each message is attached to a tag $t \in \{t_1, \ldots, t_d\}$, which can be time stamps or message identifiers. The ciphertext also includes the set of tags corresponding to the stamped messages. The delegator applies the puncture algorithm to puncture her secret key by tag $t \in \{t_1, \ldots, t_d\}$ if she wants to revoke the capability of decryption tag t. This addressing issue is achieved the forward secrecy in the messaging system.

Next to delegate the decryption right, the delegator sends the template of puncture key TK and the re-encryption key $\mathsf{R}_{B \leftarrow A}$ to the proxy server, then the proxy uses the $\mathsf{R}_{B \leftarrow A}$ to re-encrypt the ciphertext. The proxy will delegate TK and ciphertext to the delegatee. From the template key TK, the delegatee can derive his/her own puncture keys. The decryption of delegatee is input of the puncture key and his secret key. The proposed scheme first achieve IND-CCA security, which is a considerable security assumption to provide a more realistic adversarial model. We are inspired by the papers of [7,9,17] which present scheme to apply a strongly unforgeable one-time signature to bind ciphertext components altogether and offer a secure against chosen ciphertext attacks in the manner of [8].

Table 1. Performance comparison between our proposed scheme and [17] scheme.

Scheme	Level 1 - ciphertext	Level 2 - ciphertext	Level 1 dec	Level 2 dec	Attack
[17]	$2\ \mathsf{Sig} + 4\lvert\mathbb{G}\rvert + 1\lvert\mathbb{G}_T\rvert$	$2\ \mathsf{Sig} + 2\lvert\mathbb{G}\rvert + 1\lvert\mathbb{G}_T\rvert$	1p	1p	IND-CCA
PPRE	$2\ \mathsf{Sig} + (7+d)\lvert\mathbb{G}\rvert + 1\lvert\mathbb{G}_T\rvert$	$2\ \mathsf{Sig} + (2+d)\lvert\mathbb{G}\rvert + 1\lvert\mathbb{G}_T\rvert$	tp	tp	IND-CCA

We highlight a detailed computation of our Puncturable Proxy Re-Encryption and typical Unidirectional PRE of [17]. The schemes are compared in terms of the order of the underlying group, ciphertext size, decryption cost, and security assumption. We use d to denote the number of tags, and t the number of puncture tags.

Related Work. In the early concept presented in [18], the original recipient must be available for re-encrypting ciphertexts when needed, which is not always feasible. Later, [4] first proposed a proxy re-encryption, which establishes an actual notion with the elegance of the construction. This scheme is based on Elgamal, and it is constructed upon a group $\mathbb{G}$ of prime order p. [2,3] proposed the first unidirectional PRE scheme, based on bilinear pairings. These schemes are also the first to present the idea of multiple ciphertext space. Apart from [2,3,9] presents the first CCA-secure bidirectional scheme in the standard model, while the unidirectional case, [16,17] achieve the chosen ciphertext security in the standard model. [24] then proposed bidirectional schemes without pairings under CCA-secure in the random oracle model. [14] proposed a new fashion of PRE scheme as the first identity-based encryption proxy encryption (IBPRE) scheme. Using the identity-based, this scheme uses the identities of the delegator and delegatee as their public keys. Another interesting proposal proposed by [1] defines the notion of key privacy in the context of PRE, which prevents the proxy to derive the identities of both sender and receiver from a re-encryption key. As the new variances of IBPRE, [10] presented IB-PRE scheme built upon the reductions to the security of IBE [22]. [19] proposed Hybrid proxy re-encryption, which ciphertext encrypted by public key encryption scheme can be re-encrypted to ciphertexts under an identity-based encryption scheme. Recently, [20] introduces proxy re-encryption with the scenario of key rotation of data stored on the

cloud to reduce the rotating cost. Several works proposed as condition [23], type-based proxy re-encryption [21] to produce the diversity for PRE. In addition, PRE is appropriate to deploy in the cloud services. [13] proposed the variant of proxy re-encryption schemes in the dropbox, and [5] produced efficient and secure shared storage.

2 Preliminaries

2.1 Bilinear Map

Let $\mathbb{G}$ and $\mathbb{G}_{\mathbb{T}}$ be two multiplicative cyclic groups of same prime order p, and g a generator of $\mathbb{G}$. We define $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a bilinear map with the following properties: (1) Bilinearity : $e(u^a, v^b) = e(u^b, v^a) = e(u, v)^{ab}$ for all $u,v \in \mathbb{G}$ and $a,b \in \mathbb{Z}_p$. (2) Non-degeneracy : $e(g, g) \neq 1$. Notice that the map e is symmetric since $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$.

2.2 The 3-Weak Decision Bilinear Diffie-Hellman Inversion (3-WDBDHI)

The Decision 3-wDBDHI problem is the intractability of a variant of Decisional Bilinear Diffie-Hellman [6] assumption, which consider the indistinguishable computational of $e(g, g)^{b/a}$ from tuple of random elements $(g, g^a, g^{a^2}, g^{a^q}, g^b)$. There is a distinguisher $\mathsf{A}t, \epsilon$- breaks the assumption if it runs in polynomial t time and $|\Pr[\mathsf{A}(g, g^a, g^{a^2}, g^{a^3}, g^b, e(g, g)^{b/a}) = 1 | a, b \in_R \mathbb{Z}_p^*] - \Pr[\mathsf{A}(g, g^a, g^{a^2}, g^{a^3}, g^b, e(g, g)^z) = 1 | a, b, z \in_R \mathbb{Z}_p^*| \leq \epsilon(k)$. In the works of [12,17], the 3-wDBDHI problem is shown obviously that it is not easier than the (q-DBDHI) problem [6] for $q \leq 3$, which is to recognize $e(g, g)^{1/a}$ given $(g, g^a, \ldots, g^{a^q}) \in \mathbb{G}^{q+1}$.

2.3 One-Time Signatures

We apply the CHK method [8] to use one-time signatures, which consist of a triple of algorithms $\mathsf{Sig} = (\mathcal{G}, \mathcal{S}, \mathcal{V})$. The algorithm inputs of a security parameter λ, $\mathcal{G}$ generates a one-time key pair (ssk, svk) while, for any message M, $\mathcal{V}(\sigma, svk, M)$ outputs 1 whenever $\sigma = \mathcal{S}(ssk, M)$ and 0 otherwise. The strongly unforgetable one-time signatures are presented [8], which means that no PPT adversary can create a new signature for a previously signed message.

Definition 1. $\mathsf{Sig}= (\mathcal{G}, \mathcal{S}, \mathcal{V})$ is a strong one time signature if the probability is negligible for any PPT forger $\mathcal{F}$

$$Adv^{\mathsf{OTS}} = Pr[(ssk, svk) \leftarrow \mathcal{G}(\lambda); (M, St) \leftarrow \mathcal{F}(svk); \sigma \leftarrow \mathcal{S}(ssk, M); (M', \sigma') \leftarrow \mathcal{F}(M, \sigma, svk, St) : \mathcal{V}(\sigma', svk, M') = 1 \wedge (M', \sigma') \neq (M, \sigma)],$$

where St denotes $\mathcal{F}$'s state information across stages.

2.4 Lagrange Polynomial and Interpolation

Suppose that a polynomial of degree d is uniquely defined by a set of points $(x_0, y_0), (x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$. The Lagrange form of the polynomial allows the computation of a point x on the polynomial using only $d+1$ points as follows:

$$q(x) = L(x, xc, yc) = \sum_{j=0}^{d} (yc[j] \cdot \ell(x, j, xc)),$$

where, $xc = [x_0, \ldots, x_{d+1}]$ and $yc = [y_0, \ldots, y_{d+1}]$ and the Lagrange basis polynomial $\ell(\ldots)$ is:

$$\ell(x, j, xc) = \prod_{0 \le m, m \ne j \le d} \frac{x - xc[m]}{xc[j] - xc[m]}.$$

Applying the Lagrange polynomial form, a random degree d polynomial $q(x)$ is selected, which consists of sampling d random values $r_1 \ldots, r_d$ from $\mathbb{Z}_p$, setting points $(1, r_1), (2, r_2), \ldots, (d, r_d)$ and setting the final point to $(0, \beta)$ to guarantee $q(0) = \beta$. Lagrange interpolation can compute $V(x)$ without knowledge of the polynomial coefficients by only the public values $g^{q(0)}, \ldots, g^{q(d)}$ as: $V(x) = g^{q(x)} = g^{\sum_{j=0}^{d} y_j \ell(x,j,xc)} = \prod_{j=0}^{d} (g^{q(j)})^{\ell(x,j,xc)}$, where $\ell(x, j, xc)$ is already defined.

3 Model and Security Notions

3.1 Puncturable Proxy Re-Encryption

Puncturable Proxy Re-Encryption scheme has Global-setup, KeyGeneration, Re-KeyGen, Puncture, $\mathsf{Encryption}_1$ (is not re-encrypt-able), $\mathsf{Encryption}_2$, Re-Encryption, and $\mathsf{Decryption}_1$, $\mathsf{Decryption}_2$ algorithms defined in the following.

- $\mathsf{Global\text{-}setup}(1^k, d)$. On input a security parameter k, a maximum number of tags per ciphertext d, the algorithm outputs the public parameter param, and initial puncture key PSK_0.
- $\mathsf{Key\text{-}Generation}(\mathsf{param}, \mathsf{PSK}_0)$. On input the public parameter param and an initial puncture key PSK_0, the algorithms generates the public/secret key pair for a user A, then combines the secret key A and PSK_0 to produce new puncture key PSK'_0.
- $\mathsf{Puncture}(\mathsf{param}, \mathsf{TK}, \mathsf{PSK}_{i-1}, t)$. On input an existing key PSK_{i-1} as $\{\mathsf{PSK}'_0, \mathsf{PSK}_1, \ldots, \mathsf{PSK}_{i-1}\}$, and a tag t, the algorithm outputs PSK_i.
- $\mathsf{Re\text{-}KeyGen}(\mathsf{param}, \mathsf{sk}_\mathsf{A}, \mathsf{pk}_\mathsf{B})$. On input the public parameter param, secret key A, and public key B. A first generates a template puncture key TK by the param and sk_A. Then A then publicly delegates to user B a re-encryption key $\mathsf{R}_{\mathsf{B}\leftarrow\mathsf{A}}$, and encrypted form $\mathsf{Enc}_{\mathsf{pk}_\mathsf{B}}(\mathsf{TK})$.

- $\mathsf{Encryption}_1(\mathsf{param}, \mathsf{pk}_\mathsf{A}, M, t_1, \ldots, t_d)$. On input the param, public key of the user A, a message M, and a set of tags $(t_1, \ldots, t_d)$, the algorithm outputs the first level ciphertext CT_1 along with the tags $(t_1, \ldots, t_d)$.
- $\mathsf{Encryption}_2(\mathsf{param}, \mathsf{pk}_\mathsf{A}, M, t_1, \ldots, t_d)$. On input the param, public key of user A, a message M, and a set of tags $(t_1, \ldots, t_d)$, the algorithm outputs the second level ciphertext CT_2 along with the tags $t_1, \ldots, t_d$.
- $\mathsf{Re\text{-}Encryption}(\mathsf{CT}_2, \mathsf{R}_{\mathsf{B}\leftarrow\mathsf{A}})$. On input the second level ciphertext CT_2 along with the tags $t_1, \ldots, t_d$, a re-key $\mathsf{R}_{A\leftarrow B}$. The algorithm first checks the validity of CT_2. If CT_2 is well-formed, the algorithm computes from CT_2 by the re-encryption key $\mathsf{R}_{\mathsf{B}\leftarrow\mathsf{A}}$, and produce the ciphertext CT_1, along with the tags $t_1, \ldots, t_d$. Otherwise, CT_2 is declared 'invalid.'
- $\mathsf{Decryption}_1(\mathsf{param}, \mathsf{sk}_\mathsf{B}, \mathsf{CT}_1, t_1, \ldots, t_d)$. On the input param, the secret key of user B, punctured key PSK_i ,ciphertext CT_1 along with $(t_1, \ldots, t_d)$, the algorithm outputs message M or 'invalid.'
- $\mathsf{Decryption}_2(\mathsf{param}, \mathsf{sk}_\mathsf{A}, \mathsf{PSK}_i, \mathsf{CT}_2, t_1, \ldots, t_d)$. On the input param, the secret key of user A, punctured key PSK_i, second ciphertext CT_2 along with $(t_1, \ldots, t_d)$, the decryption outputs message as M or 'invalid.'

Correctness. For any common public parameters param, for any message $m \in \{0,1\}^*$ and any couple of public/secret key pair $(\mathsf{pk}_\mathsf{A}, \mathsf{sk}_\mathsf{A}), (\mathsf{pk}_\mathsf{B}, \mathsf{sk}_\mathsf{B})$ these algorithms should satisfy the following conditions:

$$\begin{aligned}
&\mathsf{Decryption}_1(\mathsf{param}, \mathsf{sk}_\mathsf{A}, \mathsf{PSK}_i, \mathsf{CT}_1) = M;\\
&\mathsf{Decryption}_2(\mathsf{param}, \mathsf{sk}_\mathsf{A}, \mathsf{PSK}_i, \mathsf{CT}_2) = M;\\
&\mathsf{Decryption}_1(\mathsf{param}, \mathsf{sk}_\mathsf{B}, \mathsf{Re\text{-}Encryption}(\mathsf{CT}_2, \mathsf{R}_{\mathsf{B}\leftarrow\mathsf{A}}),\\
&\mathsf{Re\text{-}KeyGen}(\mathsf{param}, \mathsf{sk}_\mathsf{A}, \mathsf{pk}_\mathsf{B}), \mathsf{PSK}_i) = M.
\end{aligned}$$

We use the standard security notions of proxy re-encryption schemes [2,3,11,17], which initializes empty lists of corrupted users CU and honest users HU. In addition, we define two empty sets P, C, a counter n, a targeted user x^*, and a set of tags $t_1^*, \ldots, t_d^*$. Then, A Puncturable Proxy Re-Encryption scheme is replayable chosen-ciphertext attack (RCCA) secure at level 2 ciphertexts for any PPT adversary $\mathcal{A}$ if the probability

$$\begin{aligned}
&\Pr[\mathsf{param} \leftarrow \mathsf{Global\text{-}setup}(1^k, d); (\mathsf{pk}_{x^*}, \mathsf{sk}_{x^*}) \leftarrow \mathsf{Key\text{-}Generation}_{x^*}(\mathsf{param});\\
&\{(\mathsf{pk}_x, \mathsf{sk}_x) \leftarrow \mathsf{Key\text{-}Generation}_{\mathsf{HU}}(\mathsf{param})\}; \{(\mathsf{pk}_y, \mathsf{sk}_y) \leftarrow \mathsf{Key\text{-}Generation}_{\mathsf{CU}}(\mathsf{param})\};\\
&\{\mathsf{R}_{x\leftarrow x^*} \leftarrow \mathsf{Re\text{-}KeyGen}(\mathsf{sk}_{x^*}, \mathsf{pk}_x)\}, \{\mathsf{R}_{x^*\leftarrow x} \leftarrow \mathsf{Re\text{-}KeyGen}(\mathsf{sk}_x, \mathsf{pk}_{x^*})\};\\
&\{\mathsf{R}_{x'\leftarrow x} \leftarrow \mathsf{Re\text{-}KeyGen}(\mathsf{sk}_x, \mathsf{pk}'_x)\}, \{\mathsf{R}_{x\leftarrow y} \leftarrow \mathsf{Re\text{-}KeyGen}(\mathsf{sk}_y, \mathsf{pk}_x)\};\\
&\{n++, \mathsf{PSK}_n = \mathsf{Puncture}_{x^*}(\mathsf{param}, \mathsf{TK}, \mathsf{PSK}'_{n-1}, t), P \leftarrow t\};\\
&\{n++, \mathsf{PSK}_n = \mathsf{Puncture}_{\mathsf{HU}}(\mathsf{param}, \mathsf{TK}, \mathsf{PSK}'_{n-1}, t), P \leftarrow t\};\\
&\{n++, \mathsf{PSK}_n = \mathsf{Puncture}_{\mathsf{CU}}(\mathsf{param}, \mathsf{TK}, \mathsf{PSK}'_{n-1}, t), P \leftarrow t\}; \mathbf{Corrupt()};\\
&(m_1, m_0, St) \leftarrow \mathcal{A}^{\mathcal{O}_{1-dec}, \mathcal{O}_{reenc}}(\mathsf{pk}_{x^*}, \{(\mathsf{pk}_x, \mathsf{sk}_x)\},\\
&\{\mathsf{R}_{x\leftarrow x^*}\}, \{\mathsf{R}_{x\leftarrow x^*}\}, \{\mathsf{R}_{x'\leftarrow x}\}, \{\mathsf{R}_{x\leftarrow y}\}, (t_1, \ldots, t_d), (t_1^*, \ldots, t_d^*));\\
&\mu \overset{R}{\leftarrow} \{0,1\}, \mathsf{CT}_2^* = \mathsf{Encryption}_2(m_{\mu^*}, \mathsf{pk}_{x^*}, (t_1^*, \ldots, t_d^*));\\
&\mu' \leftarrow \mathcal{A}^{\mathcal{O}_{1-dec}, \mathcal{O}_{reenc}}(\mathsf{CT}_2^*, St)) : \mu' = \mu] - \frac{1}{2} < \epsilon(k),
\end{aligned}$$

with St is the state information, $\{x'\}$ are honest users.

- **Corrupt()**is invoked in the first time; the adversary issues this query. Then, the challenger returns the most recent punctured key PSK_n to the adversary, and sets $C \leftarrow P$. All subsequent queries return $\perp$.
- $\mathcal{O}$(reenc): Responding a re-encryption query from user pk_x to user pk_y, PSK, and tags $(t_1, \ldots, t_d)$ for a second level ciphertext CT_2, this oracle returns 'invalid' if CT_2 is not encrypted by $\mathsf{pk}_x, (t_1, \ldots, t_d)$. It returns $\perp$ if $\mathsf{pk}_y \in \mathsf{CU}$ and $(\mathsf{pk}_{x^*}, \mathsf{CT}_2^*, (t_1^*, \ldots, t_d^*)) = (\mathsf{pk}_x, \mathsf{CT}_2, (t_1, \ldots, t_d))$. Otherwise, $\mathsf{CT}_1 = \mathsf{Re\text{-}Encryption}(\mathsf{CT}_2, \mathsf{sk}_x, \mathsf{pk}_y)$ is returned to $\mathcal{A}$.
- $\mathcal{O}$(1-dec): Given $\mathsf{pk}_{x'}, \mathsf{CT}_1, (t_1, \ldots, t_d)$, this oracle returns 'invalid' if CT_1 is not belongs to $\mathsf{pk}_{x'}$ and $(t_1, \ldots, t_d)$. If the condition in 'guess' stage occurs similarly in the 'queries' stage, $\mathcal{B}$ outputs $\perp$. If $(\mathsf{pk}_{x'}, \mathsf{CT}_1, (t_1, \ldots, t_d))$ is Derivative of challenge pair $(\mathsf{pk}_{x^*}, \mathsf{CT}_1^*, (t_1^*, \ldots, t_d^*))$ as CT_1 is the first level ciphertext and $\mathsf{pk}_{x'} = \mathsf{pk}_{x^*}$ or $x' \in \mathsf{HU}$, it returns $\perp$. If $\mathsf{Decryption}_1(\mathsf{param}, \mathsf{sk}_{x'}, \mathsf{CT}_1, \mathsf{PSK}_i, t_1, \ldots, t_d) \in \{m_0, m_1\}$, it returns $\perp$. Otherwise, $m = \mathsf{Decryption}_1(\mathsf{param}, \mathsf{sk}_{x'}, \mathsf{PSK}_i, \mathsf{CT}_1,$.

A Puncturable Proxy Re-Encryption scheme is also replayable chosen-ciphertext attack (RCCA) *secure at level* 1 *ciphertexts.* In fact, the adversary is guaranteed to access to re-encryption keys. Since first level ciphertexts cannot be re-encrypted, the attackers is not equipped to obtain the honest-to-corrupt re-encryption keys. The $\mathcal{O}_{reenc}$ oracle is unusable since all re-encryption keys are available to $\mathcal{A}$, $\mathcal{O}_{2-dec}$ is also unnecessary. Finally, Derivative of the challenge ciphertext is simply defined as encryptions of either m_0 or m_1 with the target public key pk_{x^*}.

4 Puncturable Proxy Re-Encryption Under Chosen Ciphertext Attack

The main construction of Puncturable Proxy Re-Encryption (PPRE) applies the inherent Unidirectional Proxy Re-Encryption (UPRE) [3], where the second ciphertext is $((g^a)^s, M \cdot e(g,g)^s)$; g^a is public key of Alice. Then the proxy re-encrypts the second ciphertext into the first ciphertext as $(e((g^a)^s, g^{b/a}), M \cdot e(g,g)^s) = (e(g,g)^{bs}, M \cdot e(g,g)^s)$, which $g^{b/a}$ is the re-encryption key between Alice and Bob.

Hence, [16,17] employs the CHK transform [8] to product the re-encrypted ciphertext by the following fashion. The proxy replaces g^{as} by a randomized pair $(g^{b/ar}, g^{ars})$, for a blinding random $r \in_R \mathbb{Z}_p$. All components in second ciphertext remain in $\mathbb{G}$. Bob can eventually decrypt the message $M \cdot e(g,g)^s / (e(g,g)^{bs})^{1/b}$. Firstly, we are inspired [17]'s method incorporating the Puncturable Encryption (PE) and URPE schemes. The global setup algorithm initially shares a master secret key α as α_1, α_2, which are used as the master secret keys of PE, UPRE schemes respectively. In order to recover α_1, α_2 in decryption process, we produce a delegation key $\mathsf{DK} = g^{\alpha_2 + r_2}$ and puncture key $\mathsf{PSK} = g^{\alpha_1 + r_1 - r_2}$ as the mode of [15]. The second ciphertext is generated to $((\mathsf{DK}^a)^s, M \cdot e(g,g)^{(\alpha_1+\alpha_2)s}, F(t)^s)$, in which $F(t)$ is the arbitrary formula to compute the tags in Puncturable Encryption. Secondly, the component of ciphertext includes the $F(t)^s$, then the first

ciphertext should have $F(t)^{rs}$. Consequently, the proxy should replace DK^{as} by a randomized pair $(\mathsf{DK}^{b/ak}, g^{arsk})$ for "*double blinding randoms*" $r, k \in_R \mathbb{Z}_p$. By this way, $e(\mathsf{DK}^{b/ak}, g^{arsk}) = e(\mathsf{DK}, g)^{brs}$ can be cancel out with the exponent's components including rs. In this manner, Bob can recover the $e(g,g)^{(\alpha_1+\alpha_2)s}$ to read message M by computing the α_1, α_2 in term of puncture key, re-encryption form, respectively. In addition, we produce $(\mathsf{ct}_{12}, \mathsf{ct}'_{12}, \mathsf{ct}''_{12}) = (g^r, g^{ark}, g^{ak})$ in order to check whether the safety of ciphertext is, meanwhile the verifying step is required to achieve the IND-CCA security. We will elaborate PPRE scheme in the next description.

4.1 Description

We elaborate the Global-setup, Key-Generation, Puncture, Re-KeyGen, $\mathsf{Encryption}_1$ (is not re-encryptable), $\mathsf{Encryption}_2$, Re-Encryption, $\mathsf{Decryption}_1$, and $\mathsf{Decryption}_2$ algorithms defined in the following.

- ▶ $\mathsf{Global\text{-}setup}(1^k, d)$. On input a security parameter k, a maximum number of tags per ciphertext d, the algorithm firstly chooses a group $\mathbb{G}$ of prime order p, a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, a generator g, w, v, a hash function $H : \{0,1\}^* \to \mathbb{Z}_p$, and a strongly unforgeable one-time signature scheme $\mathsf{Sig} = (\mathcal{G}, \mathcal{S}, \mathcal{V})$. Then the algorithm randomly selects exponents $r_1, r_2, \alpha_1, \alpha_2 \in \mathbb{Z}_p$. It samples polynomial $q(x)$ of degree d. From $i = 1$ to d, it computes $q(i)$, subjects to the constraint that $q(0) = 1$.
  Secondly, the algorithms defines $V(x) = g^{q(x)}$, and let t_0 be a distinguished tag not used during normal operation. Next, it computes the initial puncture key using the master key α_1, and distinguished tag t_0: $\mathsf{PSK}_0 = (\mathsf{PSK}_{01}, \mathsf{PSK}_{02}, \mathsf{PSK}_{03}, \mathsf{PSK}_{04}) = (g^{\alpha_1+r_1-r_2}, V(H(t_0))^{r_1}, g^{r_1}, t_0)$. Using the master key α_2, the algorithm generates the global key for user key generation: $\mathsf{DK} = g^{\alpha_2+r_2}$. Finally, it outputs the public parameter: $\mathsf{param} = (g, Y = e(g, g^{\alpha_1+\alpha_2}), \mathsf{Sig}, \mathsf{DK}, g^{q(1)}, \ldots, g^{q(d)}, t_0)$, and initial puncture key PSK_0.
- ▶ $\mathsf{Key\text{-}Generation}(\mathsf{param}, \mathsf{PSK}_0)$. To generate the public/secret key pair for a user A, the algorithm randomly picks $a \in_R \mathbb{Z}_p$. Then, it sets the public key, the secret key, and new initial puncture key from SK_0 to be: $\mathsf{pk}_\mathsf{A} = \mathsf{DK}^a, \mathsf{sk}_\mathsf{A} = a, \mathsf{PSK}'_0 = (\mathsf{PSK}'_{01}, \mathsf{PSK}'_{02}, \mathsf{PSK}'_{03}, \mathsf{PSK}'_{04}) = ((g^{\alpha_1+r_1-r_2})^{1/a}, (V(H(t_0))^{r_1})^{1/a}, g^{r_1}, t_0)$.
- ▶ $\mathsf{Re\text{-}KeyGen}(\mathsf{param}, \mathsf{DK}, \mathsf{PSK}_i)$. A user A delegates to B as follows:
  - B chooses and stores a random value $u \in \mathbb{Z}_p$, then publishes (g^u, DK^u).
  - A creates $(\mathsf{R}_{\mathsf{B}\leftarrow\mathsf{A}} = \mathsf{DK}^{u/a})$.
  - A create $\mathsf{TK} = \{g^{1/a}, g^{q(1)/a}, \ldots, g^{q(d)/a}\}$.
  - A uses public key B to encrypt A's puncture keys as $\mathsf{Enc}_{\mathsf{pk}_\mathsf{B}}(\mathsf{PSK}_i)$.
  - A then delegates $(\mathsf{R}_{\mathsf{B}\leftarrow\mathsf{A}}, \mathsf{Enc}_{pk_\mathsf{B}}(\mathsf{TK}))$ to B.

- $\mathsf{Puncture}(\mathsf{param}, \mathsf{TK}, \mathsf{PSK}'_i, t)$. On input an existing key PSK_{i-1} as $\{\mathsf{PSK}_0, \mathsf{PSK}_1, \ldots, \mathsf{PSK}_{i-1}\}$, the algorithm chooses λ', r', r_t randomly from $\mathbb{Z}_p$, and computes:

$$\mathsf{PSK}'_0 = (\mathsf{PSK}'_{01} \cdot (g^{1/a})^{r'-\lambda'}, \mathsf{PSK}'_{02} \cdot (V(H(t_0))^{1/a})^{r'}, \mathsf{PSK}'_{03} \cdot g^{r'}, t_0)$$
$$\mathsf{PSK}_i = ((g^{1/a})^{\lambda'+r_t}, (V(H(t))^{1/a})^{r_t}, g^{r_t}, t).$$

  Then it outputs: $\mathsf{PSK}_i = (\mathsf{PSK}'_0, \mathsf{PSK}_1, \ldots, \mathsf{PSK}_{i-1}, \mathsf{PSK}_i)$.
- $\mathsf{Encryption}_1(\mathsf{param}, \mathsf{pk}_\mathsf{A}, M, t_1, \ldots, t_d)$. On input the param, public key of user A, a message M, and a set of tags $t_1, \ldots, t_d \in \{0,1\}^* \backslash \{t_0\}$, the algorithm first randomly chooses s, r, k in $\mathbb{Z}_p$. Secondly, the algorithm selects a one-time signature key pair (ssk, svk) randomly from $\mathcal{G}(\lambda)$. It outputs:

$$\begin{aligned}\mathsf{CT}_1 &= (\mathsf{ct}_{10}, \mathsf{ct}_{11}, \mathsf{ct}_{12}, \mathsf{ct}'_{12}, \mathsf{ct}''_{12}, \mathsf{ct}'''_{12}, \mathsf{ct}_{13}, \mathsf{ct}_{14}, \mathsf{ct}_{15_i}, \mathsf{ct}_{16}, \sigma)\\ &= (svk, M \cdot Y^{rs}, g^{ars}, g^r, g^{ark}, g^{ak}, \mathsf{DK}^{1/k}, g^{akrs}, \{V(H(t_i))^{rs}\}_{i\in\{1,\ldots,d\}},\\ &\quad (u^{svk} \cdot v)^{rs}, \mathcal{S}(ssk, (\mathsf{ct}_{11}, \mathsf{ct}_{15_i}, \mathsf{ct}_{16})))\end{aligned}$$

  along with the tags $t_1, \ldots, t_d$. In such a way, the ciphertext can be decrypted by only the user A.
- $\mathsf{Encryption}_2(\mathsf{param}, \mathsf{pk}_\mathsf{A}, M, t_1, \ldots, t_d)$. On input the param, public key of user A, a message M, and a set of tags $t_1, \ldots, t_d \in \{0,1\}^* \backslash \{t_0\}$, the algorithm first randomly chooses s in $\mathbb{Z}_p$. Secondly, the algorithm selects a one-time signature key pair (ssk, svk) randomly from $\mathcal{G}(\lambda)$. It outputs:

$$\begin{aligned}\mathsf{CT}_2 &= (\mathsf{ct}_{20}, \mathsf{ct}_{21}, \mathsf{ct}_{22}, \mathsf{ct}_{23_i}, ct_{24}, \sigma)\\ &= (svk, M \cdot Y^s, g^{as}, \{V(H(t_i))^s\}_{i\in\{1,\ldots,d\}}, (u^{svk} \cdot v)^s, \mathcal{S}(ssk, (\mathsf{ct}_{21}, \mathsf{ct}_{23_i}, \mathsf{ct}_{24_i}))),\end{aligned}$$

  along with the tags $t_1, \ldots, t_d$. In such a way, the ciphertext can be decrypted by user A and her delegatees.
- $\mathsf{Re\text{-}Encryption}(\mathsf{CT}_2, \mathsf{R}_{\mathsf{B}\leftarrow\mathsf{A}})$. On input the second level ciphertext CT_2, a re-key $\mathsf{R}_{A\leftarrow B}$, and a set of tags $t_1, \ldots, t_d \in \{0,1\}^* \backslash \{t_0\}$. The algorithm first checks the validity of CT_2 by verifying the following conditions:

$$e(\mathsf{ct}_{22}, u^{\mathsf{ct}_{20}} \cdot w) \stackrel{?}{=} e(g^a, \mathsf{ct}_{24}), \tag{1}$$
$$\mathcal{V}(\mathsf{ct}_{20}, \sigma, (\mathsf{ct}_{21}, \mathsf{ct}_{23_i}, \mathsf{ct}_{24})) \stackrel{?}{=} 1. \tag{2}$$

  If CT_2 is well-formed, the algorithm chooses r, k randomly from $\mathbb{Z}_p$, then computes from CT_2 :

$$\begin{aligned}\mathsf{CT}_1 &= (\mathsf{ct}_{10}, \mathsf{ct}_{11}, \mathsf{ct}_{12}, \mathsf{ct}'_{12}, \mathsf{ct}''_{12}, \mathsf{ct}'''_{12}, \mathsf{ct}_{13}, \mathsf{ct}_{14}, \mathsf{ct}_{15_i}, \mathsf{ct}_{16}, \sigma)\\ &= (svk, M \cdot Y^{rs}, g^{ars}, g^r, g^{ark}, g^{ak}, (\mathsf{DK}^{u/a})^{1/k}, g^{akrs}, \{V(H(t_i))^{rs}\}_{i\in\{1,\ldots,d\}},\\ &\quad (u^{svk} \cdot v)^{rs}, \mathcal{S}(ssk, (\mathsf{ct}_{11}, \mathsf{ct}_{15_i}, \mathsf{ct}_{16}))),\end{aligned}$$

  along with the tags $t_1, \ldots, t_d$. Otherwise, CT_2 is declared '$\mathsf{invalid}$'.
- $\mathsf{Decryption}_1(\mathsf{param}, \mathsf{sk}_\mathsf{B}, \mathsf{Enc}_{\mathsf{pk}_\mathsf{B}}(\mathsf{PSK}_i), \mathsf{CT}_1, t_1, \ldots, t_d)$. On the input param, the secret key of user B, encrypted form $\mathsf{Enc}_{\mathsf{pk}_\mathsf{B}}(\mathsf{PSK}_i)$, re-encrypted ciphertext

CT_1 along with $\{t_1, \ldots, t_d\}$, the algorithm first checks the validity of CT_1 by verifying the following conditions:

$$e(\mathsf{ct}''_{12}, \mathsf{ct}_{14}) \stackrel{?}{=} e(\mathsf{DK}^u, \mathsf{ct}'_{12}), \tag{3}$$

$$e(\mathsf{ct}_{13}, u^{\mathsf{ct}_{10}} \cdot w) \stackrel{?}{=} e(\mathsf{ct}''_{12}, \mathsf{ct}_{16}), \tag{4}$$

$$\mathcal{V}(\mathsf{ct}_{10}, \sigma, (\mathsf{ct}_{11}, \mathsf{ct}_{15_i}, \mathsf{ct}_{16})) \stackrel{?}{=} 1. \tag{5}$$

If (3)-(5) hold, then for $j = 0, \ldots, i$, the punctured key PSK_i is parsed as $(\mathsf{PSK}_{i1}, \mathsf{PSK}_{i2}, \mathsf{PSK}_{i3}, \mathsf{PSK}_{i4})$. Next, it computes a set of coefficients $w_1, \ldots, w_d, w^*$ such that: $w^* \cdot q(H(\mathsf{PSK}_{i4})) + \sum_{k=1}^{d} (w_k \cdot q(H(t_k))) = q(0) = 1$ Finally, it computes:

$$\begin{aligned}
A &= \prod_{j=0}^{i} \frac{e(\mathsf{PSK}_{j1}, \mathsf{ct}_{12})}{e(\mathsf{PSK}_{j3}, \prod_{k=1}^{d} \mathsf{ct}_{15,k}^{w_k}) \cdot e(\mathsf{PSK}_{j2}, \mathsf{ct}_{12})^{w^*}} \\
&= \frac{e((g^{\alpha_1 + r_1 - r_2 + r' - \lambda'})^{1/a}, g^{ars})}{e(g^{r_1 + r'}, \prod_{k=1}^{d} V(H(t_k))^{rsw_k}) \cdot e((V(H(t_0))^{r'+r_0})^{1/a}, g^{ars})^{w^*}} \\
&\quad \cdots \frac{e(g^{\lambda' + r_t}, g^{rs})}{e(g^{r_t}, \prod_{k=1}^{d} V(H(t_k))^{w_k}) \cdot e(V(H(t))^{r_t}, g^{rs})^{w^*}} \\
&= \frac{e(g,g)^{(\alpha_1 + r_1 - r_2 + r' - \lambda')rs}}{e(g,g)^{rs(r_1 + r')}} \cdots \frac{e(g,g)^{(r_t + \lambda')rs}}{e(g,g)^{r_t sr}} = e(g,g)^{(\alpha_1 - r_2)rs}. \\
B &= e(\mathsf{ct}_{13}, \mathsf{ct}_{14}) = e(g^{(\alpha_2 + r_2)u/ak}, g^{arks}) = e(g,g)^{(\alpha_2 + r_2)rus},
\end{aligned}$$

and outputs message as: $M = \frac{\mathsf{ct}_{11}}{A \cdot B^{1/u}}$.

► $\mathsf{Decryption}_2(\mathsf{param}, \mathsf{sk}_\mathsf{A}, \mathsf{PSK}_i, \mathsf{CT}_2, t_1, \ldots, t_d)$. On the input param, the secret key of user A, puncture key PSK_i, ciphertext CT_2 along with $\{t_1, \ldots, t_d\}, t$, the decryption algorithm first computes: for $j = 0, \ldots, i$, the punctured key PSK_i is parsed as $(\mathsf{PSK}_{i1}, \mathsf{PSK}_{i2}, \mathsf{PSK}_{i3}, \mathsf{PSK}_{i4})$. Next, it computes a set of coefficients $w_1, \ldots, w_d, w^*$ such that $w^* \cdot q(H(\mathsf{PSK}_{i4})) + \sum_{k=1}^{d} (w_k \cdot q(H(t_k))) = q(0) = 1$. Finally, it computes:

$$A = \prod_{j=0}^{i} \frac{e(\mathsf{PSK}_{j1}, \mathsf{ct}_{22})}{e(\mathsf{PSK}_{j3}, \prod_{k=1}^{d} \mathsf{ct}_{23,k}^{w_k}) \cdot e(\mathsf{PSK}_{j2}, \mathsf{ct}_{22})^{w^*}}$$

$$= \frac{e((g^{\alpha_1 + r_1 - r_2 + r' - \lambda'})^{1/a}, g^{as})}{e(g^{r_1 + r'}, \prod_{k=1}^{d} V(H(t_k))^{sw_k}) \cdot e((V(H(t_0))^{r' + r_0})^{1/a}, g^{as})^{w^*}}$$

$$\cdots \frac{e(g^{\lambda' + r_t}, g^s)}{e(g^{r_t}, \prod_{k=1}^{d} V(H(t_k))^{w_k}) \cdot e(V(H(t))^{r_t}, g^s)^{w^*}}$$

$$= \frac{e(g,g)^{(\alpha_1 + r_1 - r_2 + r' - \lambda')s}}{e(g,g)^{s(r_1 + r')}} \cdots \frac{e(g,g)^{(r_t + \lambda')rs}}{e(g,g)^{r_t s}} = e(g,g)^{\alpha_1 s} e(g,g)^{-r_2 s}.$$

$$B = e(\mathsf{ct}_{22}, \mathsf{DK}) = e(g^{as}, g^{\alpha_2 + r_2}),$$

and outputs message as: $M = \frac{\mathsf{ct}_{11}}{A \cdot B^{1/a}}$.

4.2 Security

Theorem 1. *Assuming the strong unforgebility of the one-time signature, our Puncturable Proxy Re-Encryption scheme is RCCA−secure at level 2 under the* $3 - wDBDHI$ *assumption.*

Proof. Let $(g, A_{-1} = g^{1/a}, A_1 = g^a, A_2 = g^{a^2}, B = g^b, T)$ be modified $3 - wDBDHI$ instance. We build an algorithm $\mathcal{B}$ deciding if $T = (g,g)^{b/a^2}$ out of a successful RCCA adversary $\mathcal{A}$.

We define an event F_{OTS} and bound its probability to occur. Let $\mathsf{CT}_2^* = (\mathsf{ct}_{20}^*, \mathsf{ct}_{21}^*, \mathsf{ct}_{22}^*, \mathsf{ct}_{23_i}^*, \mathsf{ct}_{24}^*, \sigma^*)$ be the challenge ciphertext received by $\mathcal{A}$, and the set $(t_1^*, \ldots, t_d^*)$ be the target set initially output by $\mathcal{A}$. At some points in the process, F_{OTS} is the even that $\mathcal{A}$ issues a decryption query for a first level ciphertext $\mathsf{CT}_1 = (svk^*, \mathsf{ct}_{11}, \mathsf{ct}_{12}, \mathsf{ct}'_{12}, \mathsf{ct}''_{12}, \mathsf{ct}'''_{12}, \mathsf{ct}_{13}, \mathsf{ct}_{14}, \mathsf{ct}_{15_i}, \mathsf{ct}_{16}, \sigma)$ or a re-encryption query $\mathsf{RC}^* = (svk^*, \mathsf{rc}_1, \mathsf{rc}_2, \mathsf{rc}_{3_i}, \mathsf{rc}_4, \sigma)$ where $(\mathsf{ct}_{11}, \mathsf{ct}_{15_i}, \mathsf{ct}_{16}, \sigma) \neq (\mathsf{ct}_{21}^*, \mathsf{ct}_{23_i}^*, ct_{24}^*, \sigma^*)$ but $\mathcal{V}(\sigma, svk, (\mathsf{ct}_{11}, \mathsf{ct}_{15_i}, \mathsf{ct}_{16})) = 1$ (resp. $\mathcal{V}(\sigma, svk, (\mathsf{ct}_{11}, \mathsf{ct}_{15_i}, \mathsf{ct}_{16})) = 1$). In the *queries* stage, $\mathcal{A}$ has simply no information on svk^*. Therefore, the probability of a pre-challenge occurrence of F_{OTS} does not exceed $q_0 \cdot \delta$ if q_0 is the overall number of oracle queries and δ denotes the maximal probability. In the guess stage, F_{OTS} is enhanced to an algorithm breaking the strong unforgeability of the one-time signatures. Therefore, the probability $Pr[F_{\mathsf{OTS}}] \leq q_0/p + Adv^{\mathsf{OTS}}$, where $q_0/p + Adv^{\mathsf{OTS}}$ must be negligible by assumption.

Global Setup Phase. $\mathcal{B}$ generates a one-time signature key pair $(ssk^*, svk^*) \leftarrow \mathcal{G}(\lambda)$ and provides $\mathcal{A}$ with public parameters including $w = A_1^{\beta_1(\alpha_1 + \alpha_2)}$ and $v = A_1^{(-\beta_1 svk^*)((\alpha_1 + \alpha_2))} \cdot A_2^{\beta_2(\alpha_1 + \alpha_2)}$ for random $\beta_1, \beta_2, \alpha_1, \alpha_2$ in $\mathbb{Z}_p$. Observe that w and v define a hash function $F(svk) = w^{svk} \cdot v = A_1^{\alpha_1(svk - svk^*)} \cdot A_2^{\alpha_2}$,

and computes $g^{\alpha_1}, g^{\alpha_2}$. $\mathcal{B}$ chooses $d+1$ points $\theta_0, \theta_1, \ldots, \theta_d$ uniformly at random from $\mathbb{Z}_p$, in which θ_0 is a distinguished value not used normal simulation. Then it implicitly sets $q(0) = 1$, while $q(t_i) = \theta_i$, then $V(H(t_i)) = A_2^{q(t_i)} = g^{a^2\theta_i}$. $\mathcal{B}$ continuously initializes two empty sets P, C and a counter $\tau = 0$.

$\mathcal{B}$ generates the initial puncture key as $\mathsf{PSK}_0 = (\mathsf{PSK}_{01}, \mathsf{PSK}_{02}, \mathsf{PSK}_{03}, \mathsf{PSK}_{04}) = (A_1^{\alpha_1+r_1+r_2}, V(H(t_0))^{r_1}, g^{r_1}, t_0)$, and the global key for user key generation $\mathsf{DK} = g^{\alpha_2+r_2}$, with $r_1, r_2, \in_R \mathbb{Z}_p$.

Phase 1. $\mathcal{A}$ can repeatedly issue any of the following queries: Hereafter, we call HU the set of honest parties, including user x^* that is assigned the target public key pk_{x^*}, and CU the set of corrupt parties. Throughout the game, $\mathcal{A}$'s environment is simulated as follows:

– Key-Generation: public keys of honest users $x \in \mathsf{HU}\backslash x^*$ are defined as $\mathsf{pk}_x = A_1^x = g^{ax}$ for a randomly chosen x in $\mathbb{Z}_p$, also implicitly sets $\mathsf{sk}_x = x$. In addition, user x will generate the PSK'_0:

$$\mathsf{PSK}'_0 = (\mathsf{PSK}'_{01}, \mathsf{PSK}'_{02}, \mathsf{PSK}'_{03}, \mathsf{PSK}'_{04}) = (A_1^{(\alpha_1+r_1-r_2)1/x}, A_1^{\theta_0 r_1 1/x}, g^{r_1}, t_0).$$

The target user's public key is set as $A_2^{x^*} = g^{x^* a^2}$, also implicitly sets $\mathsf{sk}_{x^*} = ax^*$ with $x^* \in_R \mathbb{Z}_p$. User x^* generates the key PSK'_0

$$\mathsf{PSK}'_0 = (\mathsf{PSK}'_{01}, \mathsf{PSK}'_{02}, \mathsf{PSK}'_{03}, \mathsf{PSK}'_{04}) = (A_1^{(\alpha_1+r_1-r_2)1/x^*}, A_1^{\theta_0 r_1 1/x^*}, g^{r_1}, t_0).$$

The key pair of a corrupted user $x \in \mathsf{CU}$ is set as (g^x, x), for a random $x \in_R \mathbb{Z}_p$. The key PSK'_0 of corrupted user is generated

$$\mathsf{PSK}'_0 = (\mathsf{PSK}'_{01}, \mathsf{PSK}'_{02}, \mathsf{PSK}'_{03}, \mathsf{PSK}'_{04}) = (A_1^{(\alpha_1+r_1-r_2)1/x}, A_1^{\theta_0 r_1 1/x}, g^{r_1}, t_0).$$

So that all pairs of keys can be given to $\mathcal{A}$.
To generate re-encryption keys from player x to player y, $\mathcal{B}$ has to distinguish several situations:

- If $x \in \mathsf{HU}\backslash\{x^*\}$ and $y = x^*$, $\mathcal{B}$ returns $\mathsf{R}_{x\leftarrow x^*} = (g^{\alpha_2+r_2})^{x^* \cdot a^2/(ax)} = A_1^{(\alpha_2+r_2)x^*/x}$, and $\mathsf{TK} = \{g^{1/x}, g^{a^2\theta_i/x}\}$, which is a valid re-encryption key.
- If $x = x^*$ and $y \in \mathsf{HU}\backslash\{x^*\}$, $\mathcal{B}$ responds with $\mathsf{R}_{x^*\leftarrow y} = (g^{\alpha_2+r_2})^{ax/x^* a^2} = A_{-1}^{(\alpha_2+r_2)x/x^*}$. and $\mathsf{TK} = \{g^{1/x^*}, g^{a^2\theta_i/x^*}\}$, that also has the correct distribution.
- If $x, y \in \mathsf{HU}\backslash\{x^*\}$, $\mathcal{B}$ returns $\mathsf{R}_{x\leftarrow y} = (g^{\alpha_2+r_2})^{(ay)/(ax)} = g^{(\alpha_2+r_2)y/x}$, and $\mathsf{TK} = \{g^{1/x}, g^{a^2\theta_i/x}\}$,.
- If $x \in \mathsf{HU}\backslash\{x^*\}$ and $y \in \mathsf{CU}$, $\mathcal{B}$ outputs $\mathsf{R}_{x\leftarrow y} = (g^{\alpha_2+r_2})^{y/(ax)} = A_{-1}^{(\alpha_2+r_2)y/x}$, and $\mathsf{TK} = \{g^{1/x}, g^{a^2\theta_i/x}\}$, which is also computable.
- Finally, $\mathcal{B}$ uses public key y to encrypt $\mathsf{Enc}_{\mathsf{pk}_y}(\mathsf{TK})$ x's puncture key.

– Puncture: $\mathcal{B}$ increments n, and computes: $\mathsf{PSK}_n = \mathsf{Puncture}(\mathsf{param}, \mathsf{PSK}'_{n-1}, \mathsf{TK}, t)$, and adds t to set P, we consider:

Corrupt() query and $\{t_1^*, \ldots, t_d^*\} \cap C = \emptyset$. $\mathcal{B}$ now chooses randomly $r', r_t, \lambda \in \mathbb{Z}_p$. Thus, it outputs the following:

$$\mathsf{PSK}_0'' = ((\mathsf{PSK}_{01}' \cdot A_1^{r'-\lambda'})^{1/x}, \mathsf{PSK}_{02}' \cdot (A_1^{\theta_0 r'})^{1/x}, \mathsf{PSK}_{03}' \cdot g^{r'}, t_0),$$
$$\mathsf{PSK}_i = ((A_1^{\lambda'+r_t})^{1/x}, (V(H(t))^{r_t})^{1/x}, g^{r_t}, t) = ((A_1)^{(\lambda'+r_t)1/x}, A_1^{\theta_t r_t 1/x}, g^{r_t}, t)$$

Corrupt() is invoked at the first time; the adversary issues this query. Then, the challenger returns the most recent punctured key PSK_n to the adversary and sets $C \leftarrow P$. All subsequent queries return $\perp$.

– Re-Encryption queries: Responding to a re-encryption query from user x to user y for a second level ciphertext $\mathsf{CT}_2 = (\mathsf{ct}_{20}, \mathsf{ct}_{21}, \mathsf{ct}_{22}, \mathsf{ct}_{23_i}, ct_{24}, \sigma)$, $\mathcal{B}$ returns 'invalid' if the following testing is not bypassed (1)–(2)
  - If $x \neq x^*$ or if $x = x^*$ and $y \in \mathsf{HU} \backslash \{x^*\}, \mathcal{B}$ simply re-encrypts using the re-encryption key which is available in either case.
  - If $x = x^*$, and $y \in \mathsf{CU}$,
    * If $\mathsf{ct}_{20} = svk^*$, $\mathcal{B}$ encounters an occurrence of F_{OTS} and halts. Indeed, re-encryptions of the challenge ciphertext towards corrupt users are disallowed in the 'guess' stage. Therefore, $(\mathsf{ct}_{21}, \mathsf{ct}_{23_i}, \mathsf{ct}_{24}, \sigma) \neq (\mathsf{ct}_{21}^*, \mathsf{ct}_{23_i}^*, \mathsf{ct}_{24}^*, \sigma^*)$ since we would have $\mathsf{CT}_2 \neq \mathsf{CT}_2^*$ and $x \neq x^*$ if $(\mathsf{ct}_{21}, \mathsf{ct}_{23_i}, \mathsf{ct}_{24}, \sigma) \neq (\mathsf{ct}_{21}^*, \mathsf{ct}_{23_i}^*, \mathsf{ct}_{24}^*, \sigma^*)$.
    * With the case $\mathsf{ct}_{20} \neq svk^*$, $x = x^*$ and $y \in \mathsf{CU}$. Given $\mathsf{ct}_{22}^{1/x^*} = A_2^s$, from $\mathsf{ct}_{16} = \mathsf{ct}_{24} = F(svk)^s = (A_1^{\beta(svk-svk^*).A_2^{\beta_2}})^s$, $\mathcal{B}$ can compute: $A_1^s = g^{as} = (\frac{\mathsf{ct}_{24}}{\mathsf{ct}_{22}^{\beta_2/x^*}})^{\frac{1}{\beta_1(svk-svk^*)}}$.
    * Knowing g^{as} and user y's private key, $\mathcal{B}$ picks $r, k \in_R \mathbb{Z}_p$ to compute: $\mathsf{ct}_{12} = A_1^{rs} = g^{ars}, \mathsf{ct}_{12}' = g^r, \mathsf{ct}_{12}'' = g^{ark}, \mathsf{ct}_{12}''' = g^{ak}, \mathsf{ct}_{13} = (A_{-1})^{(\alpha_2+r_2)y/x^*k} = (g^{y/x^*})^{(\alpha_2+r_2)/k} \mathsf{ct}_{14} = A_1^{rsk} = g^{ars}, \mathsf{ct}_{15_i} = A_2^{\theta_i sr}$, and return $\mathsf{CT}_1 = (\mathsf{ct}_{10}, \mathsf{ct}_{11}, \mathsf{ct}_{12}, \mathsf{ct}_{12}', \mathsf{ct}_{12}'', \mathsf{ct}_{12}''', \mathsf{ct}_{13}, \mathsf{ct}_{14}, \mathsf{ct}_{15_i}, \mathsf{ct}_{16}, \sigma)$ which has the proper distribution.

– First level decryption queries: $\mathcal{A}$ may ask the decryption of a first level ciphertext $\mathsf{CT}_1 = (\mathsf{ct}_{10}, \mathsf{ct}_{11}, \mathsf{ct}_{12}, \mathsf{ct}_{12}', \mathsf{ct}_{12}'', \mathsf{ct}_{12}''', \mathsf{ct}_{13}, \mathsf{ct}_{14}, \mathsf{ct}_{15_i}, \mathsf{ct}_{16}, \sigma)$ under the public key g^x. For such a request, $\mathcal{B}$ returns 'invalid' if (3)–(5) do not hold. We assume $y \in \mathsf{HU}$ since $\mathcal{B}$ can decrypt using the known private key, then $\mathcal{B}$ can decrypt $\mathsf{Dec}_{\mathsf{sk}_y}(\mathsf{Enc}_{\mathsf{pk}_y}(\mathsf{PSK}_i))$ to receive the PSK_i. In the next step, let us first assume that $\mathsf{ct}_{10} = \mathsf{ct}_{10}^* = svk^*$. If $(\mathsf{ct}_{11}, \mathsf{ct}_{15_i}, \mathsf{ct}_{16}, \sigma) \neq (\mathsf{ct}_{11}^*, \mathsf{ct}_{15_i}^*, \mathsf{ct}_{16}^*, \sigma)$, $\mathcal{B}$ is presented with occurrence of F_{OTS} and halts. If $(\mathsf{ct}_{11}, \mathsf{ct}_{15_i}, \mathsf{ct}_{16}, \sigma) = (\mathsf{ct}_{11}^*, \mathsf{ct}_{15_i}^*, \mathsf{ct}_{16}^*, \sigma)$, $\mathcal{B}$ outputs $\perp$ which deem CT_1 as a derivative of the challenge pair of CT^*, x^*. Additionally, we reduce the computation of $e(\mathsf{ct}_{13}, \mathsf{ct}_{14}) = e(\mathsf{DK}_y, g)^{rs}$ to simulate conveniently in the next step. Lets $\mathsf{ct}_{10} \neq svk^*$, we assume that $y = x^*$, then we $\mathsf{pk}_y = g^{a^2 x^*}$ since $\mathcal{B}$ can decrypt using the known private key y. The validity of the ciphertext guarantees : $e(\mathsf{ct}_{13}, \mathsf{ct}_{14}) = e(\mathsf{DK}, g)^{a^2 yrs}, \mathsf{ct}_{16} = F(svk)^{rs} = g^{\beta_1 ars(svk-svk^*)(\alpha_1+\alpha_2)} \cdot g^{a^2 r\beta_2(\alpha_1+\alpha_2)}$.

$$A = \prod_{j=0}^{i} \frac{e(\mathsf{PSK}_{j1}, \mathsf{ct}_{12})^{x^*}}{e(\mathsf{PSK}_{j3}, \prod_{k=1}^{d} \mathsf{ct}_{15,k}^{w_k}) \cdot e(\mathsf{PSK}_{j2}, \mathsf{ct}_{12})^{x^* w^*}}$$

$$= \frac{e((A_1^{\alpha_1 + r_1 - r_2 + r' - \lambda'})^{1/x^*}, g^{ars})^{x^*}}{e(g^{r_1 + r'}, \prod_{k=1}^{d} (A_2^{\theta_0})^{rsw_k}) \cdot e(A_1^{\theta_0 r_1 1/x}, g^{ars})^{x^* w^*}}$$

$$\cdots \frac{e(g^{\lambda' + r_t}, g^{ars})^{x^*}}{e(g^{r_t}, \prod_{k=1}^{d} V(H(t_k))^{w_k}) \cdot e(A_1^{\theta_t r_1 1/x}, g^{ars})^{x^* w^*}} = e(g,g)^{a^2(\alpha_1 - r_2)rs}.$$

Next, $\mathcal{B}$ computes:

$$\gamma = e(g,g)^{ars(\alpha_1+\alpha_2)} = \left(\frac{e(\mathsf{ct}_{16}, g)}{A^{\beta_2} \cdot (\mathsf{ct}_{13}, \mathsf{ct}_{14})^{\beta_2/y(\alpha_2 + r_2)}} \right)^{\frac{1}{\beta_1(svk - svk^*)}}.$$

$\mathcal{B}$ continually computes: $e(\mathsf{ct}_{16}, A_{-1}) = e(\mathsf{ct}_{16}, g^{1/a}) = e(g,g)^{\beta_1 rs(svk - svk^*)(\alpha_1+\alpha_2)} \cdot e(g,g)^{ars\beta_2(\alpha_1+\alpha_2)}$. γ uncovers: $e(g,g)^{rs(\alpha_1+\alpha_2)} = \left(\frac{e(\mathsf{ct}_{16}, A_{-1})}{\gamma^{\beta_2/x^*}} \right)^{\frac{1}{\beta_1(svk - svk^*)}}$, and the plaintext $m = \mathsf{ct}_{11}/e(g,g)^{rs(\alpha_1+\alpha_2)}$.

In the next phases, $\mathcal{B}$ must check that m differs from messages m_0, m_1 involved in the challenge query. If $m \in \{m_0, m_1\}$. $\mathcal{B}$ returns $\perp$ according to the RCCA-security rules.

Challenge. $\mathcal{A}$ chooses messages m_0, m_1. At this stage, $\mathcal{B}$ flips a coin $\mu^* \in_R \{0,1\}$, and generates the challenge ciphertext ct_2^* as:

$$\mathsf{ct}_{20}^* = svk^*, \mathsf{ct}_{21}^* = m_{\mu^*} \cdot T^{\alpha_1+\alpha_2}, \mathsf{ct}_{22}^* = B^{x^*}, \mathsf{ct}_{23_i}^* = B^{\theta_i}, \mathsf{ct}_{24}^* = B^{\beta_2},$$

and $\sigma^* = \mathcal{S}(ssk^*, (\mathsf{ct}_{21}^*, \mathsf{ct}_{23}^*, \mathsf{ct}_{24_i}^*))$. With $\mathsf{pk}_x = g^{x^* a^2}$, $B = g^b$, and the random exponent $s = b/a^2$.

Phase 2. It is identical to Phase 1 with the following restrictions: (1) Corrupt() returns $\perp$ if $\{t_1^*, \ldots, t_d^*\} \cap P = \emptyset$; (2) Re-Encryption queries if (1)–(2) is bypassed and $\mathsf{CT}_2 \neq \mathsf{CT}_2^* \wedge x \neq x^*$. (3) $\mathsf{Decrypt}_1(\mathsf{param}, \mathsf{sk}_x, \mathsf{PSK}_i, \mathsf{CT}_1, t_1, \ldots, t_d)$ is queried.

Guess. CT_2^* is a valid encryption of m_{μ^*} if $T = e(g,g)^{b/a^2}$. In contrast, if T is random in $\mathbb{G}_T$, CT_2^* perfectly hides m_{μ^*} and $\mathcal{A}$ cannot guess μ^* with better probability than 1/2. When $\mathcal{A}$ eventually outputs her result $\mu' \in \{0,1\}$, $\mathcal{B}$ decides $T = e(g,g)^{b/a^2}$ if $\mu' = \mu$ and that T is randomly chosen. □

Theorem 2. *Assuming the strong unforgebility of the one-time signature, Puncturable Proxy Re-Encryption scheme is RCCA−secure at level 1 under the* $3 - wDBDHI$ *assumption.*

Proof. The proof of Theorem 2 will be provided in Appendix A. □

5 Conclusion

We present a Puncturable Proxy Re-Encryption Scheme supporting forward secrecy for asynchronous communication. Particularly, the proposed scheme is well-suited to many-to-many communication such as a group messaging service since a participant securely delegates computational demand operations to communicate with multiple parties to a proxy (i.e. a message server). Therefore, it allows many participants to exchange messages efficiently in group communication. One opening problem is the transformation of these schemes to obtain adaptive security. We leave it as our future work.

A Proof of Theorem 2

Let $(g, A_{-1} = g^{1/a}, A_1 = g^a, A_2 = g^{a^2}, B = g^b, T)$ be modified $3 - wDBDHI$ instance. We build an algorithm $\mathcal{B}$ deciding if $T = (g, g)^{b/a^2}$ out of a successful RCCA adversary $\mathcal{A}$.

In this proof, our simulator $\mathcal{B}$ simply halts and outputs a random bit if F_{OTS} ever occurs. Let $\mathsf{CT}_1^* = (svk^*, \mathsf{ct}_{11}^*, \mathsf{ct}_{12}^*, \mathsf{ct}_{13}^*, \mathsf{ct}_{14}^*, \mathsf{ct}_{15_i}^*, \mathsf{ct}_{16}^*, \sigma^*)$ denotes the challenge ciphertext at the first level received by $\mathcal{A}$, and the set $(t_1^*, \ldots, t_d^*)$ be the target set initially output by $\mathcal{A}$.

Global setup phase. $\mathcal{B}$ generates a one-time signature key pair $(ssk^*, svk^*) \leftarrow \mathcal{G}(\lambda)$ and provides $\mathcal{A}$ with public parameters including $w = A_1^{\beta_1}$ and $v = A_1^{-\beta_1 svk^*} \cdot A_2^{\beta_2}$ for random β_1, β_2 in $\mathbb{Z}_p$. Observe that w and v define a hash function $F(svk) = w^{svk} \cdot v = A_1^{\alpha_1(svk - svk^*)} \cdot A_2^{\alpha_2}$.
$\mathcal{B}$ also selects randomly $\alpha_1, \alpha_2 \in \mathbb{Z}_p$, and computes $g^{\alpha_1}, g^{\alpha_2}$. $\mathcal{B}$ chooses $d+1$ points $\theta_0, \theta_1, \ldots, \theta_d$ uniformly at random from $\mathbb{Z}_p$, in which θ_0 be a distinguished value not used normal simulation. Then, $\mathcal{B}$ implicitly sets $q(0) = 1$, while $q(t_i) = \theta_{t_i}$, then $V(H(t_i)) = gA_1^{q(t_i)} = g^{a\theta_{t_i}}$. $\mathcal{B}$ continuously initializes two empty sets P, C and a counter $\tau = 0$.
$\mathcal{B}$ generates the initial puncture key as $\mathsf{PSK}_0 = (\mathsf{PSK}_{01}, \mathsf{PSK}_{02}, \mathsf{PSK}_{03}, \mathsf{PSK}_{04}) = (A_1^{\alpha_1 + r_1 + r_2}, V(H(t_0))^{r_1}, g^{r_1}, t_0)$, and the global key for user key generation $\mathsf{DK} = g^{\alpha_2 + r_2}$, with $r_1, r_2, \in_R \mathbb{Z}_p$.

Phase 1. $\mathcal{A}$ can repeatedly issue any of the following queries: we call HU the set of honest parties, including user $x*$ that is assigned the target public key pk_{x^*}, and CU the set of corrupt parties. Throughout the game, $\mathcal{A}$'s environment is simulated as follows:

- Key-Generation: public keys of honest users $x \in \mathsf{HU} \backslash x^*$ and corrupt users $x \in \mathsf{CU}$ are defined as $\mathsf{pk}_x = g^x$ for a randomly chosen x in $\mathbb{Z}_p$. In addition, user x will generate the PSK_0':

$$\mathsf{PSK}_0' = (\mathsf{PSK}_{01}', \mathsf{PSK}_{02}', \mathsf{PSK}_{03}', \mathsf{PSK}_{04}') = (A_1^{(\alpha_1 + r_1 - r_2)1/x}, A_1^{\theta_0 r_1 1/x}, g^{r_1}, t_0).$$

  The target user's public key is set as $A_1^{x^*} = g^a$.

$$\mathsf{PSK}_0' = (\mathsf{PSK}_{01}', \mathsf{PSK}_{02}', \mathsf{PSK}_{03}', \mathsf{PSK}_{04}') = (g^{(\alpha_1 + r_1 - r_2)}, g^{\theta_0 r_1}, g^{r_1}, t_0).$$

For corrupt users $i \in \mathsf{CU}$, public key and secret key are both disclosed To generate re-encryption keys from player x to player y, all re-encryption keys are computed:

- If $x, y \neq x^*, \mathsf{R}_{x \leftarrow y} = g^{(\alpha_2 + r_2)y/x}$
- If $y \neq x^*, \mathsf{R}_{x^* \leftarrow y} = A_{-1}^{(\alpha_2 + r_2)y}$ and $\mathsf{R}_{y \leftarrow x^*} = A_1^{(\alpha_2 + r_2)1/y}$.

– Puncture: . $\mathcal{B}$ increments n, and computes: $\mathsf{PSK}_n = \mathsf{Puncture}(\mathsf{param}, \mathsf{PSK}'_{n-1}, \mathsf{TK}, t)$, and adds t to set P, we consider: **Corrupt()** is queried and $\{t_1^*, \ldots, t_d^*\} \cap C = \emptyset$. $\mathcal{B}$ now chooses randomly $r', r_t, \lambda \in \mathbb{Z}_p$. Thus it outputs the following:

$$\begin{aligned}
\mathsf{PSK}''_0 &= (\mathsf{PSK}'_{01} \cdot (A_1^{r'-\lambda'})^{1/x}, \mathsf{PSK}'_{02} \cdot (V(H(t_0))^{r'})^{1/x}, \mathsf{PSK}'_{03} \cdot g^{r'}, t_0) \\
&= (\mathsf{PSK}'_{01} \cdot (A_1^{r'-\lambda'})^{1/x}, \mathsf{PSK}'_{02} \cdot (A_1^{\theta_0 r'})^{1/x}, \mathsf{PSK}'_{03} \cdot g^{r'}, t_0), \\
\mathsf{PSK}_i &= ((A_1^{\lambda'+r_t})^{1/x}, (A_1^{\theta_t r_t})^{1/x}, g^{r_t}, t).
\end{aligned}$$

Corrupt() is called at the first time; the adversary issues this query. Then the challenger returns the most recent punctured key PSK_n to the adversary and sets $C \leftarrow P$. All subsequent queries return $\perp$.

– First level decryption queries: $\mathcal{A}$ may ask the decryption of a first level ciphertext $\mathsf{CT}_1 = (\mathsf{ct}_{10}, \mathsf{ct}_{11}, \mathsf{ct}_{12}, \mathsf{ct}'_{12}, \mathsf{ct}''_{12}, \mathsf{ct}'''_{12}, , \mathsf{ct}_{13}, \mathsf{ct}_{14}, \mathsf{ct}_{15_i}, \mathsf{ct}_{16}, \sigma)$ under the public key g^x. For such a request, $\mathcal{B}$ returns 'invalid' if (3)−(5) do not hold. We assume $y \in \mathsf{HU}$ since $\mathcal{B}$ can decrypt using the known private key, then $\mathcal{B}$ can decrypt $\mathsf{Dec}_{\mathsf{sk}_y}(\mathsf{Enc}_{\mathsf{pk}_y}(\mathsf{PSK}_i))$ to receive the PSK_i. In the next step, let us first assume that $\mathsf{ct}_{10} = \mathsf{ct}^*_{10} = svk^*$. If $(\mathsf{ct}_{11}, \mathsf{ct}_{15_i}, \mathsf{ct}_{16}, \sigma) \neq (\mathsf{ct}^*_{11}, \mathsf{ct}^*_{15_i}, \mathsf{ct}^*_{16}, \sigma)$, $\mathcal{B}$ is presented with occurence of F_{OTS} and halts. If $(\mathsf{ct}_{11}, \mathsf{ct}_{15_i}, \mathsf{ct}_{16}, \sigma) = (\mathsf{ct}^*_{11}, \mathsf{ct}^*_{15_i}, \mathsf{ct}^*_{16}, \sigma)$, $\mathcal{B}$ outputs $\perp$ which deem CT_1 as a derivative of the challenge pair of CT^*, x^*. We have to compute:

$$\begin{aligned}
&\mathsf{ct}_{12} = A_1^{rs} = g^{ars}, \mathsf{ct}'_{12} = g^r, \mathsf{ct}''_{12} = g^{ark}, \mathsf{ct}'''_{12} = g^{ak}, \mathsf{ct}_{13} = (A_{-1})^{(\alpha_2+r_2)y/k} \\
&= (g^{1/y})^{(\alpha_2+r_2)/k}, \mathsf{ct}_{14} = A_1 rsk = g^{ars}, \mathsf{ct}_{15_i} = A_2^{\theta_i sr},
\end{aligned}$$

for unknown exponents $r, k \in_R \mathbb{Z}_p$. We reduce the computation of $e(\mathsf{ct}_{13}, \mathsf{ct}_{14})$ equals to $e(\mathsf{DK}_y, g)^{rs}$ to simulate conveniently in the next step. Lets $\mathsf{ct}_{10} \neq svk^*$, we assume that $y = x^*$, then we $\mathsf{pk}_y = g^a$ since $\mathcal{B}$ can decrypt using the known private key y. The validity of the ciphertext guarantees

$$\begin{aligned}
e(\mathsf{ct}_{13}, \mathsf{ct}_{14}) &= e(\mathsf{DK}, g)^{ars}, \\
\mathsf{ct}_{16} = F(svk)^{rs} &= g^{\beta_1 ars(svk - svk^*)(\alpha_1+\alpha_2)} \cdot g^{a^2 r \beta_2 (\alpha_1+\alpha_2)}.
\end{aligned}$$

Then,

$$A = \prod_{j=0}^{i} \frac{e(\mathsf{PSK}_{j1}, \mathsf{ct}_{12})^{x^*}}{e(\mathsf{PSK}_{j3}, \prod_{k=1}^{d} \mathsf{ct}_{15,k}^{w_k}) \cdot e(\mathsf{PSK}_{j2}, \mathsf{ct}_{12})^{x^* w^*}}$$

$$= \frac{e((g^{\alpha_1 + r_1 - r_2 + r' - \lambda'})^{1/x^*}, g^{ars})^{x^*}}{e(g^{r_1 + r'}, \prod_{k=1}^{d} (A_1^{\theta_0})^{rsw_k}) \cdot e(g^{\theta_0 r_1 1/x}, g^{ars})^{x^* w^*}}$$

$$\cdots \frac{e(g^{\lambda' + r_t}, g^{ars})^{x^*}}{e(g^{r_t}, \prod_{k=1}^{d} V(H(t_k))^{w_k}) \cdot e(g^{\theta_t r_1 1/x}, g^{ars})^{x^* w^*}} = e(g,g)^{a(\alpha_1 - r_2)rs}.$$

$\mathcal{B}$ computes: $e(g,g)^{rs(\alpha_1+\alpha_2)} = \left(\frac{e(\mathsf{ct}_{16}, A_{-1})}{A^{\beta_2 \cdot (\mathsf{ct}_{13}, \mathsf{ct}_{14})^{\beta_2 / y(\alpha_2 + r_2)}}} \right)^{\frac{1}{\beta_1 (svk - svk^*)}}$, and recovers the plaintext $m = \mathsf{ct}_{11} / e(g,g)^{rs(\alpha_1+\alpha_2)}$.

- If $e(\mathsf{ct}_{13,\mathsf{ct}14}) = e(\mathsf{ct}^*_{13,\mathsf{ct}^*_1 4})$, $\mathcal{B}$ returns $\perp$ meaning that CT_1 is simply a re-randomization of the challenge ciphertext.
- We require $(\mathsf{ct}_{11}, \mathsf{ct}_{15_i}, \mathsf{ct}_{16}, \sigma) \neq (\mathsf{ct}^*_{11}, \mathsf{ct}^*_{15_i}, \mathsf{ct}^*_{16}, \sigma)$, which is an occurence of F_{OTS} and implies $\mathcal{B}$'s termination.

In the next phases, $\mathcal{B}$ must check that m differs from messages m_0, m_1 involved in the challenge query. If $m \in \{m_0, m_1\}$. $\mathcal{B}$ returns $\perp$ according to the RCCA-security rules.

Challenge. $\mathcal{A}$ chooses messages m_0, m_1. At this stage, $\mathcal{B}$ flips a coin $\mu^* \in_R \{0,1\}$, and generates the challenge ciphertext ct^*_1 as:

$$\mathsf{ct}^*_{10} = svk^*, \mathsf{ct}^*_{11} = m_{\mu^*} \cdot T^{\alpha_1 + \alpha_2}, \mathsf{ct}^*_{12} = B^{\gamma x^*}, \mathsf{ct}'^*_{12} = A_1^{\gamma}, \mathsf{ct}''^*_{12} = A_2^{\gamma k}, \mathsf{ct}'''^*_{12} = A_1^{k},$$
$$\mathsf{ct}^*_{13} = A_{-1}^{k(\alpha_2 + r_2)}, \mathsf{ct}^*_{14} = B^{k\gamma}, \mathsf{ct}^*_{15_i} = B^{\theta_i \gamma}, \mathsf{ct}^*_{16} = B^{\beta_2},$$

and $\sigma^* = \mathcal{S}(ssk, (\mathsf{ct}^*_{11}, \mathsf{ct}^*_{14_i}, \mathsf{ct}^*_{15}, \mathsf{ct}^*_{16}))$. With $\mathsf{pk}_x = g^{x^* a}$, $B = g^b$, and $r = a\gamma, k, s = b/a^2$ with the random numbers $\gamma, k \in \mathbb{Z}_p$.

Phase 2. This phase is identical to Phase 1 with following restrictions: (1) Corrupt() returns $\perp$ if $\{t^*_1, \ldots, t^*_d\} \cap P = \emptyset$. (2) $\mathsf{Decrypt}_1(param, \mathsf{sk}_\mathsf{A}, \mathsf{PSK}_i, \mathsf{CT}_1, t_1, \ldots, t_d)$ returns $\perp$ if $(\mathsf{CT}_1, t_1, \ldots, t_d) \neq (\mathsf{CT}^*_1, t^*_1, \ldots, t^*_d)$.

Guess. CT^*_1 is a valid encryption of m_{μ^*} if $T = e(g,g)^{b/a^2}$. In contrast, if T is random in $\mathbb{G}_T$, CT^*_1 perfectly hides m_{μ^*} and $\mathcal{A}$ cannot guess μ^* with better probability than 1/2. When $\mathcal{A}$ eventually outputs her result $\mu' \in \{0,1\}$, $\mathcal{B}$ decides $T = e(g,g)^{b/a^2}$ if $\mu' = \mu$ and that T is randomly chosen.

References

1. Ateniese, G., Benson, K., Hohenberger, S.: Key-private proxy re-encryption. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 279–294. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00862-7_19
2. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Trans. Inf. Syst. Secur. **9**, 1–30 (2006)
3. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. In: NDSS (2015)
4. Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. Adv. Cryptol. - EUROCRYPT **1403**, 127–144 (1998)
5. Blazy, O., Bultel, X., Lafourcade, P.: Two secure anonymous proxy-based data storages. In: Proceedings of the 13th ICETE. pp. 251–258 (2016)
6. Boneh, D., Boyen, X.: Efficient selective identity-based encryption without random oracles. J. Cryptol. **24**, 659–693 (2011)
7. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-39200-9_16
8. Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_13
9. Canetti, R., Hohenberger, S.: Chosen-ciphertext secure proxy re-encryption. In: Proceedings of the 14th ACM CCS (2007)
10. Chu, C.-K., Tzeng, W.-G.: Identity-based proxy re-encryption without random oracles. In: Garay, J.A., Lenstra, A.K., Mambo, M., Peralta, R. (eds.) ISC 2007. LNCS, vol. 4779, pp. 189–202. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75496-1_13
11. Derler, D., Krenn, S., Lorünser, T., Ramacher, S., Slamanig, D., Striecks, C.: Revisiting proxy re-encryption: forward secrecy, improved security, and applications. In: Abdalla, M., Dahab, R. (eds.) PKC 2018. LNCS, vol. 10769, pp. 219–250. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76578-5_8
12. Dodis, Y., Yampolskiy, A.: A verifiable random function with short proofs and keys. In: Proceedings of the 8th PKC. pp. 416–431 (2005)
13. Ge, C., Susilo, W., Fang, L., Wang, J., Shi, Y.: A cca-secure key-policy attribute-based proxy re-encryption in the adaptive corruption model for dropbox data sharing system. Des. Codes Crypt. **86**(11), 2587–2603 (2018)
14. Green, M., Ateniese, G.: Identity-based proxy re-encryption. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 288–306. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72738-5_19
15. Green, M.D., Miers, I.: Forward secure asynchronous messaging from puncturable encryption. In: Proceedings of the 2015 IEEE S and P, pp. 305–320. IEEE Computer Society (2015)
16. Libert, B., Vergnaud, D.: Unidirectional chosen-ciphertext secure proxy re-encryption. IEEE Trans. Inf. Theor. **57**(3), 1786–1802 (2011)
17. Libert, B., Vergnaud, D.: Unidirectional chosen-ciphertext secure proxy re-encryption. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 360–379. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78440-1_21

18. Mambo, M., Okamoto, E.: Proxy cryptosystems: delegation of the power to decrypt ciphertexts. IEICE Trans. Fundam. **80**–**A**, 54–63 (1997)
19. Matsuo, T.: Proxy re-encryption systems for identity-based encryption. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 247–267. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73489-5_13
20. Myers, S., Shull, A.: Efficient hybrid proxy re-encryption for practical revocation and key rotation. Cryptology ePrint Archive, Report 2017/833 (2017). https://eprint.iacr.org/2017/833
21. Tang, Q.: Type-based proxy re-encryption and its construction. In: Proceedings of the 9th INDOCRYPT. pp. 130–144. Berlin, Heidelberg (2008)
22. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_7
23. Weng, J., Deng, R.H., Ding, X., Chu, C.K., Lai, J.: Conditional proxy re-encryption secure against chosen-ciphertext attack. In: Proceedings of the 4th ASIACCS. pp. 322–332 (2009)
24. Weng, J., Deng, R.H., Liu, S., Chen, K.: Chosen-ciphertext secure bidirectional proxy re-encryption schemes without pairings. Inf. Sci. **180**(24), 5077–5089 (2010)