Non-Transferable Proxy Re-Encryption

Hui Guo, Zhenfeng Zhang and Jing Xu

Institute of Software, Chinese Academy of Sciences, Beijing, China.
{guohui,zfzhang,xujing}@tca.iscas.ac.cn

Abstract. Proxy re-encryption (PRE) allows a semi-trusted proxy to transform a ciphertext for Alice into a ciphertext of the same message for Bob. The traditional security notion of PRE focuses on preventing the proxy with the re-encryption key learning anything about the encrypted messages. However, such a basic security requirement is clearly not enough for many scenarios where the proxy can collude with Bob. A desirable security goal is therefore to prevent a malicious proxy colluding with Bob to re-delegate Alice's decryption right. In 2005, Ateniese, Fu, Green and Hohenberger first proposed this intriguing problem called non-transferability, in the sense that the only way for Bob to transfer Alice's decryption capability is to expose his own secret key. It captures the notion that Bob cannot collude with the proxy and transfer Alice's decryption right without compromising his own decryption capability. However, over the last decade, no solutions have achieved this property.

In this paper, we positively resolve this open problem. In particular, we give the first construction of non-transferable proxy re-encryption where the attacker is allowed to obtain one pair of keys consisting of Bob's secret key and the corresponding re-encryption key. Using indistinguishability obfuscation and k-unforgeable authentication as main tools, our scheme is provably secure in the standard model. The essential idea behind our approach is to allow Bob's secret key to be evoked in the process of decrypting Alice's ciphertext while hiding the fact that only Bob could decrypt it by the obfuscated program. In addition, we also show a negative result: a CPA secure proxy re-encryption scheme with "error-freeness" property cannot be non-transferable.

1 Introduction

In 1998, Blaze et al. [5] first proposed the primitive of proxy re-encryption (PRE) in order to solve the problem of delegating decryption rights. In a proxy re-encryption scheme, a semi-trusted proxy with re-encryption keys can transform a ciphertext intended for Alice (delegator) into another ciphertext of the same plaintext intended for Bob (delegatee). The proxy cannot, however, learn anything about the underlying plaintext. According to the direction of transformation, PRE can be categorized into bidirectional PRE, in which the proxy can transform ciphertexts from Alice to Bob and vice versa, and unidirectional PRE, in which the proxy cannot transform ciphertexts in the opposite direction. There is also another method to classify PRE schemes, namely, a scheme is multi-hop, if the ciphertext can be transformed from Alice to Bob and then to Charlie and so on, otherwise it is single-hop. In this paper, we concentrate on single-hop unidirectional PRE.

In the last decade, proxy re-encryption has attracted many researchers' attention and has many intriguing applications, such as email forwarding [7], distributed files systems [1][2], digital rights management [20] and cloud data sharing [22]. In all these cases, the proxy is supposed to be semi-trusted and uncontrolled, that is, the traditional security notions prevent the adversary who has the re-encryption key from learning anything about the underlying messages. However, such a basic security requirement is clearly not enough for some applications. For instance, in cloud service scenarios, Alice may be a data broker and she sells sensitive data encrypted and stored in the cloud. Clearly, Alice has the data ownership and she does not hope anyone including the cloud access the data without her authorization. If Bob asks for her data, he has to pay first. Then Alice generates a re-encryption key for Bob and sends it to the cloud. With the re-encryption key, the cloud could transform the ciphertext for Bob. A desirable security goal is therefore to prevent a malicious cloud from colluding with Bob to further re-delegate Alice's decryption rights to another unpaid

user Carol. Obviously, as soon as the cloud gets the re-encryption key, it could collude with Bob to decrypt Alice's ciphertext and forward the plaintext to Carol. However, this approach requires Bob to remain active and online. Again, Bob can always send his secret key to Carol, but in doing so, he assumes a security risk that is potentially injurious to himself.

In 2005, Ateniese et al. [1][2] first proposed this problem and introduced a notion of non-transferability. A proxy re-encryption scheme is said to be non-transferable if the proxy and a set of colluding delegatees cannot re-delegate decryption rights to other parties without compromising any malicious delegatee's decryption capability. This property can be seen as a tradeoff solution to protect Alice's "benefit" in delegating her decryption right: when a malicious delegatee Bob colludes with the proxy to transfer Alice's decryption capability to others, he must expose his own decryption capability as a pay. Informally speaking, non-transferability requires that the only way for Bob to transfer decryption capability of Alice is to expose his own secret key. More generally, if the proxy and Bob generate a useful decryption box L_a in decrypting Alice's ciphertext, then L_a is also useful in decrypting Bob's ciphertext.

Non-transferability of PRE scheme is an interesting problem in practice as illegal transfers of delegation may lead to financial loss and even worse, whereas the malicious user has very little risk of getting caught. Obviously, discouraging such behaviors seems much easier than preventing them, so one of the solutions is to trace the malicious proxy after its collusion with one or more delegatees. This means that the penalty can only be applied "after-the-fact", i.e., only after the unauthorized transfer has been finished. However, it is more desirable to have a better way to prevent collusion than to just discourage collusion. Non-transferability provides a proactive way of deterrence. In particular, if a malicious coalition transfers an unauthorized pirate decryption capability, any recipient will be able to obtain the decryption capability of the malicious delegatee. On the other hand, honest delegatees' decryption capabilities are guaranteed to remain hidden. How to construct a non-transferable PRE scheme is the main open problem left for PRE schemes.

Unfortunately, there are no solutions for this property since it was proposed. In fact, most known proposals for PRE allow the proxy colluding with Bob to directly generate new re-encryption keys from Alice to another user who has not been authorized by Alice, as pointed out in [12]. To achieve non-transferability, we need Alice's ciphertext has the property that only if a decryptor holds Bob's secret key (and the related re-encryption key), can it get the plaintext; otherwise, it will get nothing useful. In order to give a concrete construction, we resort to indistinguishability obfuscator ($i\mathcal{O}$) which is recently realized in [11] using candidate multilinear maps [10].

Indistinguishability obfuscation is a notion of general program obfuscation, proposed by Barak et al. [3], the target of which is to make computer programs unintelligible while not changing their functionality. Indistinguishability obfuscation requires obfuscations of any two (equal-size) programs that have the same functionality are computationally indistinguishable. Since Garg et al. [11] introduced the first candidate indistinguishability obfuscation, indistinguishability obfuscation becomes a very important tool to achieve unprecedented results. In this paper, we use it to resolve the problem of non-transferability in PRE.

1.1 Our Contributions

In this paper, we resolve the above open question and propose the first non-transferable PRE scheme. Our construction is conceptually very simple, and it uses indistinguishability obfuscation and k-unforgeable authentication [6] as main tools. Roughly, the ciphertexts are obfuscated programs and re-encryption keys are generated by the specific authentication scheme [6]. The proposed PRE scheme is CPA secure at the 2nd level ciphertext and the 1st level ciphertext. We prove that our scheme satisfies non-transferability, in the sense that the attacker is allowed to obtain one pair of keys consisting of a re-encryption key and the related delegatee's secret key, but it cannot transfer the delegator's decryption right without exposing the delegatee's

secret key. It captures the most likely transference attack in reality: the proxy leaks the re-encryption keys carelessly and a malicious delegatee obtains the corresponding re-encryption key for himself.

Furthermore, we also demonstrate an interesting impossibility result in non-transferability. In particular we show that if a CPA secure PRE scheme is "error-free", then it cannot meet non-transferability. Informally, a PRE scheme is error-free if the re-encryption algorithm itself does not introduce any error to the re-encrypted ciphertext. This notion is formalized by Zhang et al. [23] and used to achieve CCA secure PRE schemes. As far as we know almost all existing PRE schemes have error-freeness except the traceable PRE scheme in [17]. This result rules out the possibility of constructing error-free and non-transferable PRE schemes.

1.2 Our Techniques

When the proxy colludes with Bob, the proxy can use the relevant re-encryption key to transform Alice's ciphertext to another ciphertext intended for Bob, so it's clear that they are able to generate a pirate decryption box in decrypting any ciphertext of Alice. To obtain the property of non-transferability, the crucial difficulty is how to "extract" the decryption capability of Bob from the pirate decryption box. Generally speaking, when the process of re-encryption and decryption can easily be "obfuscated" into a program without any information about Bob's private key, it is impossible to achieve non-transferability. In this paper, we show both positive and negative results for the goal of constructing non-transferable PRE schemes.

The Negative Result. Zhang et al. formalized a strong correctness notion called "error-freeness" [23], which requires the re-encryption algorithm itself does not introduce any error to the re-encrypted ciphertext. That is, the correctness of re-encryption is preserved for all ciphertexts of Alice, even including malformed ciphertexts. With error-freeness, given any ciphertext for Alice and the corresponding re-encrypted ciphertext for Bob, Alice and Bob will always obtain the same message. Although this property is strongly desired in constructing CCA secure PRE schemes, it does conflict with the property of non-transferability.

To see this, consider the generic attack where the proxy colludes with Bob and derives a decryption box L_a which is an obfuscation of the program: when given Alice's ciphertext, first re-encrypt it for Bob then decrypt the re-encrypted ciphertext using Bob's secret key. Due to error-freeness, L_a is indistinguishable from the following obfuscated program: when given Alice's ciphertext, decrypt it using Alice's secret key. In the second obfuscated program, there is no need of Bob's secret key. Thus, intuitively we think that the decryption box L_a hides all the private information of Bob and cannot be used to decrypt Bob's ciphertext. Furthermore, we also prove that it is impossible for a error-free PRE scheme to meet the non-transferability.

The Positive Result. To avoid the above impossibility result, our main goal is to give the first construction of non-transferable PRE.

In most previous works, although the colluding proxy and Bob cannot recover Alice's secret key, they can derive equivalent sub-keys that suffice to generate new re-encryption keys for unauthorized users, which means that these unauthorized users can decrypt all transformable ciphertexts intended for Alice. Obviously, such constructions cannot achieve non-transferability.

Motivated by this observation and the negative result, we propose a completely new approach to design non-transferable PRE schemes. In particular, a re-encryption key is an authentication which is generated by Alice to authenticate the identity of Bob, and both original ciphertexts and re-encrypted ciphertexts are the obfuscated programs which take as input the corresponding private key and output the plaintext. Since k-unforgeable authentication scheme can only authenticate the designated user, even though the proxy colludes with Bob, they can not recover any useful information about Alice's private key and generate any new re-encryption keys for other unauthorized users. Moreover, in order to avoid the mentioned negative result, we generate ciphertexts with "policies" such that for a malformed ciphertext and its re-encrypted ciphertext,

Alice and Bob will get different plaintexts, which makes the transferability attack infeasible. Particularly, we use obfuscated programs to hide the plaintext and the ciphertext's "policies". A detailed description of our techniques follows.

For the construction of the ciphertexts, basically, our solution is to make Alice's ciphertext has the property that only if an adversary holds Bob's secret key sk_b and the related re-encryption key $\mathsf{rk}_{a\to b}$, can it get the plaintext m; otherwise, it will get nothing useful. Actually, it is somewhat like point-function obfuscation [4], however, it is not sufficient for realizing re-encryption functionality. Recall that in order to achieve non-transferability, Bob's decryption capability should be extracted from any illegal decryption box L_a generated by the proxy and Bob. Particularly, our solution is to evoke L_a to recover Bob's ciphertext and the main difficulties are how to embed Bob's ciphertext into a fake ciphertext of Alice for invoking L_a. Therefore, the ciphertext should have somewhat "self-contained" property and the resulting fake ciphertext should be indistinguishable from the normal one. To overcome these obstacles, we adopt the punctured programs technique [19] for applying indistinguishability obfuscation. In particular, the original ciphertext is two obfuscated programs: the first one, which is for Alice to decrypt, takes as input Alice's secret key and outputs the plaintext m; the second one, which is for re-encryption, verifies the validity of a re-encryption key and the corresponding delegatee's secret key and then outputs the plaintext m.

For the construction of the user's keys, basically, our solution is to make the re-encryption keys unforgeable. Otherwise, an adversary could transfer Alice's decryption right by forging a re-encryption key [12]. Furthermore, in order to use the security of indistinguishability obfuscation, the main difficulties are how to ensure the re-encryption keys to be information-theoretically unforgeable. Besides, the re-encryption keys should have a public verifiable algorithm for an encryptor to generate a ciphertext. To overcome the above difficulties, we adopt the specific authentication scheme in [6]. In particular, we regard an authentication, which is generated by Alice to authenticate Bob, as a re-encryption key.

Technique of Proof. For non-transferability, our primary challenge is to extract Bob's decryption capability from the illegal decryption box L_a of Alice, which is generated by the collusion attack of proxy and Bob.

Our solution is to embed Bob's ciphertext in a fake ciphertext of Alice such that the fake ciphertext is polynomial indistinguishable from a normal ciphertext of the same encrypted message. With the fake ciphertext of Alice, we could run the illegal decryption box L_a and get the underlying message. Unfortunately, though we use indistinguishability obfuscation to build the ciphertext, we could not directly obtain the indistinguishability of the fake ciphertext and the normal ciphertext. That is because, the programs being obfuscated in the fake ciphertext and the normal ciphertext have different functionalities. In particular, the normal ciphertext takes as input the secret key of any delegatee and the corresponding re-encryption key; while the fake ciphertext only takes as input Bob's secret key and the corresponding re-encryption key.

In order to show the indistinguishability of the fake ciphertext and the normal ciphertext by using the security of indistinguishability obfuscation, we build a hybrid experiment in which the normal ciphertext could not be evoked by pairs of keys $(\mathsf{rk}_{a\to b'}, \mathsf{sk}_{b'})$ of any delegatee other than Bob. The key point to make this switch is that there does not exist such a key pair $(\mathsf{rk}_{a\to b'}, \mathsf{sk}_{b'})$ leading to different outputs. In more detail, we introduce a fake key generation mode, which is also used in [6]. The scheme has an indistinguishable fake key generation mode, where there does not exist any secret keys of honest users (including Alice) or any re-encryption keys from Alice to corrupted users except the one from Alice to Bob (recall that Bob is a corrupted user). Under these restrictions, the fake ciphertext and the normal ciphertext would have the same inputs/outputs and they are indistinguishable by indistinguishability obfuscation's security. This also indicates that if a decryption box L_a is useful in decrypting Alice's ciphertext, it must run with $(\mathsf{rk}_{a\to b}, \mathsf{sk}_b)$, thus Bob's decryption capability could be extracted from it.

1.3 Related Works

Since the notion of non-transferability was introduced in 2005, there have been many attempts to prevent the transferability of PRE. In 2008, Libert and Vergnaud [17] proposed traceable proxy re-encryption, where malicious proxies that reveal their re-encryption key to third parties can be identified by the delegator. Although it is one possible approach to the non-transferable PRE, it still cannot prevent colluding proxies and delegatees from re-delegating the decryption rights. It is more desirable to have a better way to prevent collusion, not just discourage collusion. In 2011, Wang et al. [21] proposed an identity-based proxy re-encryption scheme to prevent collusion attacks. As the re-encryption key is generated using the master key of the PKG (a fully trusted private key generator), the proxy and the delegatees cannot further delegate the decryption right to others without the help of the PKG. However, this solution is undesirable since PKG in their scheme can decrypt both original and re-encrypted ciphertexts. Furthermore, PKG can transfer any re-encryption key for other users, that is, the transferable problem is still not solved. Later, Hayashi et al. [13] also introduced a relaxed notion of non-transferability and proposed two concrete constructions. Unfortunately, Isshiki et al. [15] pointed out that Hayashi et al.'s scheme is vulnerable to the forgeability attack of re-encryption keys. Moreover, the security assumption employed in their proofs can be solved efficiently. Later, Guo et al. [12] proposed an efficient PRE scheme with unforgeable re-encryption keys. However, they pointed out their security model could not capture all attacks of transference of decryption rights. Besides, there are several re-encryption constructions [8] [9] [14] achieving VBB obfuscation-based security. However, due to collusion attacks, they still cannot meet non-transferability.

2 Preliminary

In this section, we describe some primitives that will be used in our construction. Let λ be the security parameter.

2.1 Indistinguishability Obfuscation

We recall the definition of indistinguishability obfuscator ($i\mathcal{O}$) recently realized in [11] using candidate multilinear maps [10]. Note that our scheme is constructed from a generic $i\mathcal{O}$, a weaker complexity assumption than those needed for multilinear maps may eventually be sufficient for our scheme.

Definition 1 (Indistinguishability Obfuscator). An indistinguishability obfuscator $i\mathcal{O}$ for a circuit class $\mathbb{C}_\lambda$ is a PPT uniform algorithm satisfying the following conditions:

- $i\mathcal{O}(\lambda, C)$ preserves the functionality of C. That is, for any $C \in \mathbb{C}_\lambda$, if we compute $C' = i\mathcal{O}(\lambda, C)$, then $C'(x) = C(x)$ for all inputs x.
- For any λ and any two circuits $C_0, C_1 \in \mathbb{C}_\lambda$ with the same functionality, the circuits $i\mathcal{O}(\lambda, C_0)$ and $i\mathcal{O}(\lambda, C_1)$ are indistinguishable. More precisely, for all pairs of PPT adversaries (Samp, D) there exists a negligible function α such that, if

$$\Pr[(C_0, C_1, \tau) \leftarrow \text{Samp}(\lambda) : \forall x, C_0(x) = C_1(x)] > 1 - \alpha(\lambda)$$

  then

$$|\Pr[\text{D}(\tau, i\mathcal{O}(\lambda, C_0)) = 1] - \Pr[\text{D}(\tau, i\mathcal{O}(\lambda, C_1)) = 1]| < \alpha(\lambda).$$

  In this paper, we will make use of such indistinguishability obfuscators for all polynomial-size circuits:

Definition 2 (Indistinguishability Obfuscator for P/*poly*). A uniform PPT machine $i\mathcal{O}$ is called an indistinguishability obfuscator for P/*poly* if the following holds: Let $\mathbb{C}_\lambda$ be the class of circuits of size at most λ. Then $i\mathcal{O}$ is an indistinguishability obfuscator for the class $\{\mathbb{C}_\lambda\}$.

2.2 k-Unforgeable Authentication Scheme

We recall the definition of k-unforgeable authentication recently realized in [6] using k-cover-free sets [16] and PRG. An authentication scheme over an identity space $\mathcal{I}$ consists of the following three algorithms:

- $\mathsf{AuthGen}(\lambda)$: The algorithm takes as input a security parameter λ, and outputs a secret authentication key ask and a public verification key avk.
- $\mathsf{AuthProve}(\mathsf{ask}, \mathsf{id})$: The algorithm takes as input a secret authentication key ask and a user's identity id, and outputs $\mathsf{auth}_{\mathsf{id}}$.
- $\mathsf{Verify}(\mathsf{avk}, \mathsf{id}, \mathsf{auth}_{\mathsf{id}})$: The algorithm takes as input a public verification key avk, a user's identity id, and an authentication value $\mathsf{auth}_{\mathsf{id}}$, and outputs 1 if $\mathsf{auth}_{\mathsf{id}}$ is valid or 0 otherwise.

Correctness. For any identity $\mathsf{id} \in \mathcal{I}$, the following condition holds:

$$\Pr[(\mathsf{avk}, \mathsf{ask}) \leftarrow \mathsf{AuthGen}(\lambda); \mathsf{auth}_{\mathsf{id}} \leftarrow \mathsf{AuthProve}(\mathsf{ask}, \mathsf{id}) : \mathsf{Verify}(\mathsf{avk}, \mathsf{id}, \mathsf{auth}_{\mathsf{id}}) = 1] = 1.$$

Definition 3 (k-Unforgeability of Authentication Scheme). An authentication scheme is said to be k-unforgeable, if for any PPT adversary $\mathcal{A}$ and any set $\mathcal{T}$ of at most k identities, the following condition holds:

$$\Pr\left[\begin{array}{r}(\mathsf{avk}, \mathsf{ask}) \leftarrow \mathsf{AuthGen}(\lambda); \\ \mathsf{auth}_{\mathsf{id}_j} \leftarrow \mathsf{AuthProve}(\mathsf{ask}, \mathsf{id}_j), \forall \mathsf{id}_j \in \mathcal{T}; \\ (\mathsf{id}^*, \mathsf{auth}_{\mathsf{id}^*}) \leftarrow \mathcal{A}(\mathsf{avk}, \{\mathsf{id}_j, \mathsf{auth}_{\mathsf{id}_j}\}_{\mathsf{id}_j \in \mathcal{T}}) : \\ \mathsf{id}^* \notin \mathcal{T} \cap \mathsf{Verify}(\mathsf{avk}, \mathsf{id}^*, \mathsf{auth}_{\mathsf{id}^*}) = 1\end{array}\right] \leq \mathsf{negl}(\lambda),$$

where k is polynomially related to the security parameter.

Indistinguishable Fake Setup. The k-unforgeable authentication scheme constructed in [6] is accompanied with a PPT indistinguishable fake algorithm of setup. The algorithm $\mathsf{FakeAuthGen}(\lambda, \mathcal{T})$ takes as input a security parameter λ and a set $\mathcal{T}$ including k identities, and outputs a key pair $(\mathsf{ask}, \mathsf{avk})$ such that it is only possible to authenticate the identities in $\mathcal{T}$ and there does not exist any authentication for users outside of $\mathcal{T}$. Moreover, the real setup and fake setup should be indistinguishable to the adversary. Formally, the security properties of algorithm $\mathsf{FakeAuthGen}$ are described as following:

- No authentication for $\mathsf{id} \notin \mathcal{T}$:

$$\Pr\left[\begin{array}{r}(\mathsf{avk}, \mathsf{ask}) \leftarrow \mathsf{FakeAuthGen}(\lambda, \mathcal{T}) : \\ \forall \mathsf{id} \notin \mathcal{T}, \forall \mathsf{auth}_{\mathsf{id}} \in \{0,1\}^*, \mathsf{Verify}(\mathsf{avk}, \mathsf{id}, \mathsf{auth}_{\mathsf{id}}) = 0\end{array}\right] > 1 - \mathsf{negl}(\lambda).$$

- Indistinguishability: For any PPT adversary $\mathcal{A}$, and any set $\mathcal{T}$ of at most k identities, the following holds:

$$\begin{aligned}&\Pr\left[\begin{array}{r}(\mathsf{ask}, \mathsf{avk}) \leftarrow \mathsf{AuthGen}(\lambda); \\ \mathsf{auth}_{\mathsf{id}_j} \leftarrow \mathsf{AuthProve}(\mathsf{ask}, \mathsf{id}_j), \forall \mathsf{id}_j \in \mathcal{T} : \\ \mathcal{A}(\mathsf{avk}, \{\mathsf{auth}_{\mathsf{id}_j}\}_{\mathsf{id}_j \in \mathcal{T}}) = 1\end{array}\right] \\ \approx &\Pr\left[\begin{array}{r}(\mathsf{ask}, \mathsf{avk}) \leftarrow \mathsf{FakeAuthGen}(\lambda, \mathcal{T}); \\ \mathsf{auth}_{\mathsf{id}_j} \leftarrow \mathsf{AuthProve}(\mathsf{ask}, \mathsf{id}_j), \forall \mathsf{id}_j \in \mathcal{T} : \\ \mathcal{A}(\mathsf{avk}, \{\mathsf{auth}_{\mathsf{id}_j}\}_{\mathsf{id}_j \in \mathcal{T}}) = 1\end{array}\right].\end{aligned}$$

3 Proxy Re-Encryption

In this section, we first recall the definitions of PRE. Then, we introduce the non-transferability definition against collusion attacks.

Let M denote the plaintext space. A single-hop unidirectional PRE scheme is a tuple of the following algorithms [1]:

- $\mathsf{KeyGen}(\lambda)$: The algorithm takes as input a security parameter λ, and outputs the user's public-secret key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$: The algorithm takes as input a secret key sk_i of user i (i.e., delegator) and a public key pk_j of user j (i.e., delegatee), and outputs a re-encryption key $\mathsf{rk}_{i\to j}$.
- $\mathsf{Enc}_1(\mathsf{pk}_j, m)$: The algorithm takes as input a public key pk_j and a plaintext $m \in \mathsf{M}$, and outputs a first level ciphertext that cannot be re-encrypted for another user.
- $\mathsf{Enc}_2(\mathsf{pk}_i, m)$: The algorithm takes as input a public key pk_i and a plaintext $m \in \mathsf{M}$, and outputs a second level ciphertext that can be re-encrypted into a first level one (for a possibly different receiver) using the suitable re-encryption key.
- $\mathsf{ReEnc}(\mathsf{rk}_{i\to j}, C_i)$: The algorithm takes as input a re-encryption key $\mathsf{rk}_{i\to j}$ and a second level ciphertext C_i, and outputs a first level ciphertext C'_j re-encrypted for user j.
- $\mathsf{Dec}_1(\mathsf{sk}_j, C'_j)$: The algorithm takes as input a secret key sk_j and a first level ciphertext C'_j, and outputs a plaintext $m \in \mathsf{M}$ or an error symbol $\perp$.
- $\mathsf{Dec}_2(\mathsf{sk}_i, C_i)$: The algorithm takes as input a secret key sk_i and a second level ciphertext C_i, and outputs a plaintext $m \in \mathsf{M}$ or an error symbol $\perp$.

Correctness. For any key pairs $(\mathsf{pk}_i, \mathsf{sk}_i), (\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{KeyGen}(\lambda)$, any re-encryption key $\mathsf{rk}_{i\to j} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$ and any message $m \in \mathsf{M}$, the following conditions hold:

$$\mathsf{Dec}_1(\mathsf{sk}_j, \mathsf{Enc}_1(\mathsf{pk}_j, m)) = m; \;\; \mathsf{Dec}_2(\mathsf{sk}_i, \mathsf{Enc}_2(\mathsf{pk}_i, m)) = m;$$

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

3.1 CPA Security

The security of second level ciphertexts for single-hop unidirectional PRE schemes [1] is defined as follows.
Definition 4 (CPA Security of Second Level Ciphertext). A single-hop unidirectional PRE scheme is chosen plaintext secure at level 2 if

$$\begin{aligned} |\Pr[\{(\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{KeyGen}(\lambda)\}_{j\in\mathbb{C}\cup\mathbb{H}}; (\mathsf{pk}_{i^*}, \mathsf{sk}_{i^*}) \leftarrow \mathsf{KeyGen}(\lambda); \\ \{\mathsf{rk}_{i^*\to j} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_{i^*}, \mathsf{pk}_j)\}_{j\in\mathbb{H}}; \{\mathsf{rk}_{i\to j} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)\}_{i\in\mathbb{H}, j\in\mathbb{C}\cup\mathbb{H}\cup\{i^*\}}; \\ (m_0, m_1, st) \leftarrow \mathcal{A}(\mathsf{pk}_{i^*}, \{\mathsf{pk}_j, \mathsf{sk}_j\}_{j\in\mathbb{C}}, \{\mathsf{pk}_j\}_{j\in\mathbb{H}}, \{\mathsf{rk}_{i^*\to j}\}_{j\in\mathbb{H}}, \{\mathsf{rk}_{i\to j}\}_{i\in\mathbb{H}, j\in\mathbb{C}\cup\mathbb{H}\cup\{i^*\}}); \\ b \leftarrow \{0,1\}; C^* \leftarrow \mathsf{Enc}_2(\mathsf{pk}_{i^*}, m_b); b' \leftarrow \mathcal{A}(C^*, st) : b' = b] - 1/2| \leq \mathsf{negl}(\lambda) \end{aligned}$$

for any PPT adversary $\mathcal{A}$, where the target user's key pair $(\mathsf{pk}_{j^*}, \mathsf{sk}_{j^*})$ and the challenge ciphertext C^* is generated by the challenger, st is the state information maintained by $\mathcal{A}$, and $\mathbb{C}$ and $\mathbb{H}$ denote the set of corrupted users and the set of honest users, respectively.

Remark 1. In the above security definition, we do not provide $\mathcal{A}$ with any re-encrypted ciphertext for the following two reasons: (1) $\mathcal{A}$ can generate re-encrypted ciphertexts from user i^* to user j ($j \in \mathbb{H}$), and re-encrypted ciphertexts from user i to user j ($i \in \mathbb{H}, j \in \mathbb{C} \cup \mathbb{H} \cup \{i^*\}$), by using $\{\mathsf{rk}_{i^* \to j}\}_{j \in \mathbb{H}}$ and $\{\mathsf{rk}_{i \to j}\}_{i \in \mathbb{H}, j \in \mathbb{C} \cup \mathbb{H} \cup \{i^*\}}$. (2) $\mathcal{A}$ cannot obtain re-encrypted ciphertexts from user i^* to user j ($j \in \mathbb{C}$), since such ciphertexts implicitly provide the second level decryption capability of user i^*, which we do not consider in CPA model.

Then we recall the security of first level ciphertexts for single-hop unidirectional PRE schemes. Since first level ciphertexts cannot be re-encrypted, the adversary $\mathcal{A}$ is granted access to all re-encryption keys in this definition.

Definition 5 (CPA Security of First Level Ciphertext). A single-hop unidirectional PRE scheme is chosen plaintext secure at level 1 if

$$\begin{aligned} |\Pr[\{(\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{KeyGen}(\lambda)\}_{j \in \mathbb{C} \cup \mathbb{H}}; (\mathsf{pk}_{j^*}, \mathsf{sk}_{j^*}) \leftarrow \mathsf{KeyGen}(\lambda); \\ \{\mathsf{rk}_{j \to j'} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_j, \mathsf{pk}_{j'})\}_{j, j' \in \mathbb{C} \cup \mathbb{H} \cup \{j^*\}}; \\ (m_0, m_1, st) \leftarrow \mathcal{A}(\mathsf{pk}_{j^*}, \{\mathsf{pk}_j, \mathsf{sk}_j\}_{j \in \mathbb{C}}, \{\mathsf{pk}_j\}_{j \in \mathbb{H}}, \{\mathsf{rk}_{j \to j'}\}_{j, j' \in \mathbb{C} \cup \mathbb{H} \cup \{j^*\}}); \\ b \leftarrow \{0,1\}; C^* \leftarrow \mathsf{Enc}_1(\mathsf{pk}_{j^*}, m_b); b' \leftarrow \mathcal{A}(C^*, st) : b' = b] - 1/2| \leq \mathsf{negl}(\lambda) \end{aligned}$$

for any PPT adversary $\mathcal{A}$, where the target user's key pair $(\mathsf{pk}_{j^*}, \mathsf{sk}_{j^*})$ and the challenge ciphertext C^* is generated by the challenger, st is the state information maintained by $\mathcal{A}$, and $\mathbb{C}$ and $\mathbb{H}$ denote the set of corrupted users and the set of honest users, respectively.

A PRE scheme is said to be CPA secure, if it is CPA secure at both level 1 and level 2.

3.2 Non-Transferability

Intuitively, non-transferability requires that the only way for a delegatee to transfer delegator's decryption capability is to expose his own second level decryption capability. To formalize this intuition, we first introduce a notion of ϵ-useful second level decryption box, which is viewed as a PPT algorithm that takes as input a 2nd level ciphertext and outputs a message or $\perp$. Such a decryption box does not need to be perfect, namely, we only require it to be able to decrypt with non-negligible success probability ϵ.

Definition 6 (ϵ-Useful Second Level Decryption Box). For non-negligible probability value ϵ, a PPT algorithm $L_{i,\epsilon}$ is an ϵ-useful second level decryption box for user i, if $\Pr[m \leftarrow \mathsf{M}, C \leftarrow \mathsf{Enc}_2(\mathsf{pk}_i, m), m' \leftarrow L_{i,\epsilon}(C) : m = m'] \geq \epsilon$, where M is the plaintext space.

Definition 7 (Non-Transferability). A single-hop unidirectional PRE scheme is non-transferable against collusion attacks if for any PPT adversary $\mathcal{A}$, there exists a PPT algorithm I such that

$$\begin{aligned} \Pr[\{(\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{KeyGen}(\lambda)\}_{j \in \mathbb{C} \cup \mathbb{H}}; (\mathsf{pk}_{i^*}, \mathsf{sk}_{i^*}) \leftarrow \mathsf{KeyGen}(\lambda); \\ \{\mathsf{rk}_{i^* \to j} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_{i^*}, \mathsf{pk}_j)\}_{j \in \mathbb{D}}; \\ L_{i^*,\epsilon} \leftarrow \mathcal{A}(\mathsf{pk}_{i^*}, \{\mathsf{pk}_j, \mathsf{sk}_j\}_{j \in \mathbb{C}}, \{\mathsf{pk}_j\}_{j \in \mathbb{H}}, \{\mathsf{rk}_{i^* \to j}\}_{j \in \mathbb{D}}) : \\ L_{j^*,\epsilon} \leftarrow I(L_{i^*,\epsilon}, \mathsf{pk}_{i^*}, \{\mathsf{pk}_j\}_{j \in \mathbb{C} \cap \mathbb{D}}), j^* \in \mathbb{C} \cap \mathbb{D}] > 1 - \mathsf{negl}(\lambda) \end{aligned}$$

where $\mathcal{A}$ outputs an ϵ-useful decryption box $L_{i^*,\epsilon}$ for user i^*, I outputs an ϵ-useful decryption box $L_{j^*,\epsilon}$ for user j^*, $\mathbb{C}$ denotes the set of corrupted users, $\mathbb{H}$ denotes the set of uncorrupted users, and $\mathbb{D} \subset \mathbb{C} \cup \mathbb{H}$ denotes the set of delegatees whose corresponding re-encryption keys are corrupted.

If $|\mathbb{C} \cap \mathbb{D}| = 1$, the scheme is said to be non-transferable against one-pair-of-keys collusion attacks.

Remark 2. Non-transferability is considered as an advanced security requirement. In other words, it is meaningless for a PRE scheme to be non-transferable, unless the PRE scheme is CPA secure.

Remark 3. In our proposed scheme, the adversary $\mathcal{A}$ is allowed to obtain one pair of keys $(\mathsf{rk}_{i^*\to j}, \mathsf{sk}_j)$ in collusion attacks, e.g. $|\mathbb{C} \cap \mathbb{D}| = 1$. Nevertheless, it captures the most common attack in reality: the proxy leaks the re-encryption keys carelessly, and a malicious delegatee obtains the corresponding re-encryption key for himself.

From another point of view, if $\mathcal{A}$ obtains more than one pair of keys, non-transferability might be hardly satisfied. Without loss of generality, $\mathcal{A}$ obtains $(\mathsf{rk}_{i^*\to j_1}, \mathsf{sk}_{j_1})$ and $(\mathsf{rk}_{i^*\to j_2}, \mathsf{sk}_{j_2})$, and there might be a decryption box $L'_{i^*,\epsilon}$ with defending policy as follows. Taking a ciphertext as input, $L'_{i^*,\epsilon}$ re-encrypts it to C_{j_1} using $\mathsf{rk}_{i^*\to j_1}$ and decrypts C_{j_1} using sk_{j_1}; it repeats the re-encryption and the decryption by using $(\mathsf{rk}_{i^*\to j_2}, \mathsf{sk}_{j_2})$, and it rejects if the two executions have different results. Intuitively, such $L'_{i^*,\epsilon}$ might hardly be utilized in a black-box manner to decrypt either user j_1's or user j_2's ciphertexts.

Remark 4. In previous model [13], Bob and the proxy derive a re-encryption box R which can re-encrypt Alice's ciphertext to another user Carol. In fact, a re-encryption box of [13] and a decryption box $L_{i^*,\epsilon}$ defined in this paper could be mutually transformed. If $L_{i^*,\epsilon}$ is leaked, we can use it to decrypt Alice's ciphertext and encrypt the message to Carol. If R is leaked, we can use it to re-encrypt Alice's ciphertext and decrypt it with Carol's secret key (i.e. Carol use R to extract Bob's decryption capability). The decryption box adopted in this paper is more simple and essential.

Remark 5. For convenience, we only consider the second level decryption capabilities of the malicious delegatees. Obviously, for delegatees, the second level ciphertext is more important while the first level ciphertext might be transformed ciphertext from other users.

4 Construction

In this section we present our construction of a non-transferable proxy re-encryption scheme.

Intuitively, to ensure the delegatee can decrypt the re-encrypted ciphertext using his own private key, the existing constructions require the delegator's private key "included" in the re-encryption key. Specifically, using the re-encryption key and the delegatee's private key, even though the delegator's private key itself cannot be disclosed, the adversary can obtain the partial information about the delegator's private key such as it's one-way transformation, using which the adversary can generate a new re-encryption key. Therefore, the scheme cannot be non-transferable. On the other hand, in almost all the existing constructions, the correctness of re-encryption is preserved for all ciphertexts of Alice. Thus, with the re-encryption key $\mathsf{rk}_{a\to b}$ and the delegatee's private key sk_b, the adversary can obfuscate the re-encryption algorithm $\mathsf{ReEnc}(\mathsf{rk}_{a\to b}, C_a)$ and decryption algorithm $\mathsf{Dec_1}(\mathsf{sk}_b, C'_b)$ into a program L_a which reveals no information about Bob's secret key sk_b. More details can be found in Section 6. We believe that such attacks are the principle reasons that the existing constructions cannot provide non-transferability.

To overcome the above difficulties, we present a new methodology by applying the tools of indistinguishability obfuscation and k-unforgeable authentication. In our construction, a re-encryption key is an authentication which is generated by the delegator to authenticate the identity of a delegatee, and the ciphertexts (original and re-encrypted) are both the obfuscated programs which take as input the corresponding private key and output the plaintext. Since k-unforgeable authentication scheme can only authenticate the designated delegatee, even though the proxy colludes with a delegatee, they can not recover any useful information about the delegator's private key and generate any new re-encryption keys for other delegatees. Moreover, we use obfuscated program to generate ciphertexts with "policies" such that for a malformed ciphertext and its re-encrypted ciphertext, Alice and Bob will get different plaintext messages. The difference

in the resulting messages makes the latter attack infeasible. In particular, the obfuscated program hides the plaintext and the ciphertext's "policies". This gives an intuition why our scheme achieves non-transferability.

Let $\mathsf{Auth} = (\mathsf{AuthGen}, \mathsf{AuthProve}, \mathsf{Verify})$ be the k-unforgeable authentication scheme over an identity space $\mathcal{I} = \{0,1\}^{2\lambda}$ of [6] [1], $i\mathcal{O}$ be a program indistinguishability obfuscator, and PRG be a pseudo-random generator that maps $\{0,1\}^{\lambda}$ to $\{0,1\}^{2\lambda}$. The message space $\mathcal{M} = \{0,1\}^{\lambda}$. The description of our single-hop unidirectional PRE scheme is given below.

- $\mathsf{KeyGen}(\lambda)$: Given the security parameter λ, the user i randomly chooses s_i and computes $t_i = \mathsf{PRG}(s_i)$. It also generates $(\mathsf{avk}_i, \mathsf{ask}_i) \leftarrow \mathsf{AuthGen}(\lambda)$. Return the key pair $(\mathsf{pk}_i = (t_i, \mathsf{avk}_i), \mathsf{sk}_i = (s_i, \mathsf{ask}_i))$.
- $\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$: Given a secret key sk_i and a public key pk_j, compute $\mathsf{auth}_j \leftarrow \mathsf{AuthProve}(\mathsf{ask}_i, t_j)$ and set $\mathsf{rk}_{i\to j} = \mathsf{auth}_j$. Return the re-encryption key $\mathsf{rk}_{i\to j}$.
- $\mathsf{Enc}_1(\mathsf{pk}_j, m)$: Given a public key pk_j, a message $m \in \mathcal{M}$, build the program Cipher-I in Figure 1. Return the ciphertext $C'_j = i\mathcal{O}(\text{Cipher-I})$. Note that C'_j is an obfuscated program, whose size is padded to be the maximum of itself and those of Figure 4 and Figure 5.

Cipher-I

Constants: m, t_j.
Input: s_j.
1. If $\mathsf{PRG}(s_j) = t_j$, output m and exit.
2. Output "$\perp$".

Fig. 1. The program Cipher-I

- $\mathsf{Enc}_2(\mathsf{pk}_i, m)$: Given a public key pk_i, a message $m \in \mathcal{M}$, build the programs Cipher-II-L and Cipher-II-R in Figure 2, and create $c_l = i\mathcal{O}(\text{Cipher-II-L})$ and $c_r = i\mathcal{O}(\text{Cipher-II-R})$. Return the ciphertext $C_i = (c_l, c_r)$. Note that c_l and c_r are two obfuscated programs, whose sizes are padded to the appropriate length.

Cipher-II-L

Constants: m, t_i.
Input: s_i.
1. If $\mathsf{PRG}(s_i) = t_i$, output m and exit.
2. Output "$\perp$".

Cipher-II-R

Constants: m, avk_i.
Input: $(\mathsf{rk}_{i\to j}, s_j, t_j)$.
1. If $\mathsf{Verify}(\mathsf{avk}_i, t_j, \mathsf{rk}_{i\to j}) = 1$ and $\mathsf{PRG}(s_j) = t_j$ output m and exit.
2. Output "$\perp$".

Fig. 2. The programs Cipher-II-L and Cipher-II-R

- $\mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{pk}_j, (c_l, c_r))$: Given a re-encryption key $\mathsf{rk}_{i\to j}$, a public key pk_j and a ciphertext $C_i = (c_l, c_r)$, build the program Re-Cipher in Figure 3. Return the ciphertext $C'_j = i\mathcal{O}(\text{Re-Cipher})$. Note that C'_j is an obfuscated program, whose size is padded to the appropriate length.

[1] For any $\mathsf{id} \in \{0,1\}^{2\lambda}$, it can be interpreted as an element in $\mathbb{F}_q^{d+1}$, where $q = k\lambda, d = q - 1/k$, as required in [6]. In our scheme, k is the bound of the number of corrupted re-encryption keys for a delegator.

Re-Cipher

Constants: $\mathsf{rk}_{i\to j}, t_j, c_r$.
Input: s_j.
1. Evaluate c_r with $(\mathsf{rk}_{i\to j}, s_j, t_j)$ as input.

Fig. 3. The program Re-Cipher

– $\mathsf{Dec}_1(\mathsf{sk}_j, C'_j)$: Given a secret key sk_j and a first level ciphertext C'_j, run the program C'_j on sk_j to obtain the plaintext m or $\perp$.

– $\mathsf{Dec}_2(\mathsf{sk}_i, C_i)$: Given a secret key sk_i and a second level ciphertext $C_i = (c_l, c_r)$, run the program c_l on sk_i to obtain the plaintext m or $\perp$.

Correctness. A second level ciphertext C_i under pk_i has a form (c_l, c_r) where c_l and c_r are two obfuscated programs. Given a re-encryption key $\mathsf{rk}_{i\to j}$, C_i can be transformed into a first level ciphertext C'_j under pk_j, where C'_j is also an obfuscated program. Also, the decryption algorithm runs the obfuscated program on inputting a secret key for either level ciphertext, the correctness is obvious.

5 Security

In this section, we prove the CPA security and non-transferability of the proposed construction.

5.1 CPA Security

First, we consider the CPA security.

Theorem 1. *If* PRG *is a secure pseudo-random generator,* Auth *a k-unforgeable authentication scheme, and $i\mathcal{O}$ a secure indistinguishability obfuscator, then our PRE scheme is CPA secure at the first and second level ciphertext.*

The theorem is obtained by combining the following Lemma 1 and Lemma 2.

Lemma 1. *If* PRG *is a secure pseudo-random generator and $i\mathcal{O}$ a secure indistinguishability obfuscator, then our PRE scheme is CPA secure at the first level ciphertext.*

Experiments	Description	Reduction to
Hyb_0	the real experiment	–
Hyb_1	t_{j^*} is randomly chosen	security of PRG
Hyb_2	the challenge ciphertext C^* is an obfuscation of the program Target-Cipher-I* in Figure 5	$i\mathcal{O}$ security

Table 1. Description of the experiments in the proof of Lemma 1

Proof. We describe a proof as a sequence of hybrid experiments where the first hybrid experiment corresponds to the original CPA security experiment of the 1st level ciphertext. We prove that the attacker $\mathcal{A}$'s advantage must be negligibly close between each successive hybrid experiment and that the attacker has negligible advantage in the final experiment. Table 1 roughly describes the hybrid experiments.

• **Hyb**$_0$ The first hybrid is the real experiment.

1. The challenger generates the keys $\{\mathsf{pk}_j, \mathsf{sk}_j\}_{j\in\mathbb{C}\cup\mathbb{H}}, \{\mathsf{rk}_{j\to j'}\}_{j,j'\in\mathbb{C}\cup\mathbb{H}\cup\{j^*\}}$ and pk_{j^*} as follows.
   (a) Pick $(\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{KeyGen}(\lambda), \forall j \in \mathbb{C}\cup\mathbb{H}$.
   (b) Pick s_{j^*} at random and compute $t_{j^*} = \mathsf{PRG}(s_{j^*})$. Pick $(\mathsf{avk}_{j^*}, \mathsf{ask}_{j^*}) \leftarrow \mathsf{AuthGen}(\lambda)$. Set $\mathsf{pk}_{j^*} = (t_{j^*}, \mathsf{avk}_{j^*})$.
   (c) For all $j, j' \in \mathbb{C}\cup\mathbb{H}\cup\{j^*\}$, generate $\mathsf{rk}_{j\to j'} \leftarrow \mathsf{AuthProve}(\mathsf{ask}_j, t_{j'})$.
   It returns $\{\mathsf{pk}_j, \mathsf{sk}_j\}_{j\in\mathbb{C}}, \{\mathsf{pk}_j\}_{j\in\mathbb{H}}, \{\mathsf{rk}_{j\to j'}\}_{j,j'\in\mathbb{C}\cup\mathbb{H}\cup\{j^*\}}$ and pk_{j^*}.
2. The attacker obtains $\{\mathsf{pk}_j, \mathsf{sk}_j\}_{j\in\mathbb{C}}, \{\mathsf{pk}_j\}_{j\in\mathbb{H}}, \{\mathsf{rk}_{j\to j'}\}_{j,j'\in\mathbb{C}\cup\mathbb{H}\cup\{j^*\}}$ and pk_{j^*}. It chooses $m_0, m_1 \in \mathsf{M}$ and returns them to the challenger.
3. The challenger picks $b \in \{0,1\}$ at random, and generates the challenge ciphertext C^* as an obfuscation of the program Target-Cipher-I in Figure 4.
4. $\mathcal{A}$ receives C^* and outputs a guess b'.

Target-Cipher-I

Constants: m_b, t_{j^*}.
Input: s_{j^*}.
1. If $\mathsf{PRG}(s_{j^*}) = t_{j^*}$, output m_b and exit.
2. Output "$\perp$".

Fig. 4. The program Target-Cipher-I

• **Hyb**$_1$ This hybrid experiment is the same as Hyb$_0$ except that t_{j^*} is randomly chosen.
– This hybrid experiment is indistinguishable from the previous one by the pseudo-randomness property of PRG.

• **Hyb**$_2$ This hybrid experiment is the same as Hyb$_1$ except that we let the challenge ciphertext C^* be an obfuscation of the program Target-Cipher-I* in Figure 5.
– Since t_{j^*} is chosen at random, with high probability it is true t_{j^*} is not in the image of the PRG and the program Target-Cipher-I will always output "$\perp$". Therefore, the replacement will not change the functionality of the program Target-Cipher-I. Thus, the indistinguishability of Hyb$_1$ and Hyb$_2$ follows from the $i\mathcal{O}$ security property.

Target-Cipher-I*

Constants: t_{j^*}.
Input: s_{j^*}.
1. If $\mathsf{PRG}(s_{j^*}) = t_{j^*}$, output $\underline{0}$ and exit.
2. Output "$\perp$".

Fig. 5. The program Target-Cipher-I*

In the last hybrid experiment, since the challenge ciphertext C^* contains no information of m_b, the probability that the attacker will output a correct guess b' is $\mathsf{negl}(\lambda) + 1/2$.

Thus, the lemma follows. □

Lemma 2. *If* PRG *is a secure pseudo-random generator,* Auth *a k-unforgeable authentication scheme, and $i\mathcal{O}$ a secure indistinguishability obfuscator, then our PRE scheme is CPA secure at the second level ciphertext.*

Experiments	Description	Reduction to
Hyb_0	the real experiment	–
Hyb_1	use $\mathsf{FakeAuthGen}(\lambda, \{t_j\}_{j\in\mathbb{H}})$ for user i^*	security of authentication
Hyb_2	t_i is chosen at random for all $i \in \mathbb{H} \cup \{i^*\}$	security of PRG
Hyb_3	the challenge ciphertext C^* is the obfuscations of the programs Target-Cipher-II-L* and Target-Cipher-II-R in Figure 7	$i\mathcal{O}$ security
Hyb_4	the challenge ciphertext C^* is the obfuscations of the programs Target-Cipher-II-L* and Target-Cipher-II-R* in Figure 8	$i\mathcal{O}$ security

Table 2. Description of the experiments in the proof of Lemma 2

Proof. We describe a proof as a sequence of hybrid experiments where the first hybrid experiment corresponds to the original CPA security experiment of the 2nd level ciphertext. We prove that the attacker $\mathcal{A}$'s advantage must be negligibly close between each successive hybrid experiment and that the attacker has negligible advantage in the final experiment. Table 2 roughly describes the hybrid experiments.

• **Hyb**$_0$ The first hybrid is the real experiment.

1. The challenger generates $\{\mathsf{pk}_j, \mathsf{sk}_j\}_{j\in\mathbb{C}\cup\mathbb{H}}, \{\mathsf{rk}_{i^*\to j}\}_{j\in\mathbb{H}}, \{\mathsf{rk}_{i\to j}\}_{i\in\mathbb{H}, j\in\mathbb{C}\cup\mathbb{H}\cup\{i^*\}}$ and pk_{i^*} as follows.
   (a) For all $j \in \mathbb{C}$, pick $(\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{KeyGen}(\lambda)$.
   (b) For all $j \in \mathbb{H}$, pick s_j at random and compute $t_j = \mathsf{PRG}(s_j)$. Pick $(\mathsf{avk}_j, \mathsf{ask}_j) \leftarrow \mathsf{AuthGen}(\lambda)$. Set $\mathsf{pk}_j = (t_j, \mathsf{avk}_j)$.
   (c) For user i^*, pick s_{i^*} at random and compute $t_{i^*} = \mathsf{PRG}(s_{i^*})$. Pick $(\mathsf{avk}_{i^*}, \mathsf{ask}_{i^*}) \leftarrow \mathsf{AuthGen}(\lambda)$. Set $\mathsf{pk}_{i^*} = (t_{i^*}, \mathsf{avk}_{i^*})$.
   (d) For all $j \in \mathbb{H}$, generate $\mathsf{rk}_{i^*\to j} \leftarrow \mathsf{AuthProve}(\mathsf{ask}_{i^*}, t_j)$.
   (e) For all $i \in \mathbb{H}, j \in \mathbb{C}\cup\mathbb{H}\cup\{i^*\}$, generate $\mathsf{rk}_{i\to j} \leftarrow \mathsf{AuthProve}(\mathsf{ask}_i, t_j)$.
   It returns $\{\mathsf{pk}_j, \mathsf{sk}_j\}_{j\in\mathbb{C}}, \{\mathsf{pk}_j\}_{j\in\mathbb{H}}, \{\mathsf{rk}_{i^*\to j}\}_{j\in\mathbb{H}}, \{\mathsf{rk}_{i\to j}\}_{i\in\mathbb{H}, j\in\mathbb{C}\cup\mathbb{H}\cup\{i^*\}}$ and pk_{i^*}.
2. The attacker receives $\{\mathsf{pk}_j, \mathsf{sk}_j\}_{j\in\mathbb{C}}, \{\mathsf{pk}_j\}_{j\in\mathbb{H}}, \{\mathsf{rk}_{i^*\to j}\}_{j\in\mathbb{H}}, \{\mathsf{rk}_{i\to j}\}_{i\in\mathbb{H}, j\in\mathbb{C}\cup\mathbb{H}\cup\{i^*\}}$ and pk_{i^*}. It chooses $m_0, m_1 \in \mathsf{M}$ and returns them to the challenger.
3. The challenger picks $b \in \{0, 1\}$ at random and generate the challenge ciphertext $C^* = (c_l^*, c_r^*)$ as two obfuscations of the programs in Figure 6, separately.
4. $\mathcal{A}$ receives C^* and outputs a guess b'.

Target-Cipher-II-L

Constants: m_b, t_{i^*}.
Input: s_{i^*}.
1. If $\mathsf{PRG}(s_{i^*}) = t_{i^*}$, output m_b and exit.
2. Output "$\perp$".

Target-Cipher-II-R

Constants: m_b, avk_{i^*}.
Input: $(\mathsf{rk}_{i^*\to j}, s_j, t_j)$.
1. If $\mathsf{Verify}(\mathsf{avk}_{i^*}, t_j, \mathsf{rk}_{i^*\to j}) = 1$ and $\mathsf{PRG}(s_j) = t_j$ output m_b and exit.
2. Output "$\perp$".

Fig. 6. The programs Target-Cipher-II-L and Target-Cipher-II-R

• **Hyb**$_1$ This hybrid experiment is the same as Hyb$_0$ except that we change the key generation algorithm for user i^* to use $\mathsf{FakeAuthGen}(\lambda, \{t_j\}_{j\in\mathbb{H}})$ instead of $\mathsf{AuthGen}(\lambda)$.

– This hybrid experiment is indistinguishable from the previous one by the security of the authentication scheme.

• **Hyb**$_2$ This hybrid experiment is the same as Hyb$_1$ except that for all $i \in \mathbb{H} \cup \{i^*\}$, t_i is randomly chosen.

– The indistinguishability of Hyb$_2$ from Hyb$_1$ follows from the pseudo-randomness property of PRG.

• **Hyb**$_3$ This hybrid experiment is the same as Hyb$_2$ except that we let the challenge ciphertext be the obfuscations of the programs in Figure 7.

– Since t_{i^*} is chosen at random, with high probability it is true t_{i^*} is not in the image of the PRG and the program Target-Cipher-II-L will always output "$\perp$". Therefore, the replacement will not change the functionality of the program Target-Cipher-II-L. Thus, the indistinguishability of Hyb$_3$ and Hyb$_2$ follows from the $i\mathcal{O}$ security property.

Target-Cipher-II-L*

Constants: t_{i^*}.
Input: s_{i^*}.
1. If $\mathsf{PRG}(s_{i^*}) = t_{i^*}$, output $\underline{0}$ and exit.
2. Output "$\perp$".

Target-Cipher-II-R

Constants: m_b, avk_{i^*}.
Input: $(\mathsf{rk}_{i^*\to j}, s_j, t_j)$.
1. If $\mathsf{Verify}(\mathsf{avk}_{i^*}, t_j, \mathsf{rk}_{i^*\to j}) = 1$ and $\mathsf{PRG}(s_j) = t_j$ output m_b and exit.
2. Output "$\perp$".

Fig. 7. The programs Target-Cipher-II-L* and Target-Cipher-II-R

• **Hyb**$_4$ This hybrid experiment is same as Hyb$_3$ except that we let the challenge ciphertext be the obfuscations of the programs in Figure 8.

– Since $(\mathsf{avk}_{i^*}, \mathsf{ask}_{i^*}) \leftarrow \mathsf{FakeAuthGen}(\lambda, \{t_j\}_{j\in\mathbb{H}})$, with high probability it is true that there is no authentication for user $j \notin \mathbb{H}$. Since for all $j \in \mathbb{H}$, t_j is chosen at random, with high probability it is true t_j is not in the image of the PRG. Therefore with high probability it is true the program Target-Cipher-II-R will output "$\perp$" and the replacement will not change the functionality of the program Target-Cipher-II-R. Thus, the indistinguishability of Hyb$_4$ and Hyb$_3$ follows from the $i\mathcal{O}$ security property.

Target-Cipher-II-L*

Constants: t_{i^*}.
Input: s_{i^*}.
1. If $\mathsf{PRG}(s_{i^*}) = t_{i^*}$, output 0 and exit.
2. Output "$\perp$".

Target-Cipher-II-R*

Constants: avk_{i^*}.
Input: $(\mathsf{rk}_{i^*\to j}, s_j, t_j)$.
1. If $\mathsf{Verify}(\mathsf{avk}_{i^*}, t_j, \mathsf{rk}_{i^*\to j}) = 1$ and $\mathsf{PRG}(s_j) = t_j$ output $\underline{0}$ and exit.
2. Output "$\perp$".

Fig. 8. The programs Target-Cipher-II-L* and Target-Cipher-II-R*

In the last hybrid, since C^* contains no information of m_b, the probability that the attacker will output a correct guess b' is $\mathsf{negl}(\lambda) + 1/2$.

Thus, the lemma follows. □

5.2 Non-Transferability

Theorem 2. *If* PRG *is a secure pseudo-random generator,* Auth *a k-unforgeable authentication scheme, and $i\mathcal{O}$ a secure indistinguishability obfuscator, then our PRE scheme is non-transferable against one-pair-of-keys collusion attacks.*

Keys	Ciphertext	Experiments	Description	Reduction to
Normal	Fake	Hyb_0	the real experiment	–
Normal		Hyb_1	use $\mathsf{FakeAuthGen}(\lambda, \{t_j\}_{j\in\mathbb{D}})$ for user i^*	security of authentication
→Fake		Hyb_2	t_i is chosen at random for all $i \in \mathbb{H} \cup \{i^*\}$	security of PRG
Fake	Fake→ Normal	Hyb_3	the fake ciphertext C_f is the obfuscations of the programs Fake-Cipher-II-L and Fake-Cipher-II-R* in Figure 11	$i\mathcal{O}$ security
		Hyb_4	the fake ciphertext C_f is the obfuscations of the programs Fake-Cipher-II-L and Fake-Cipher-II-R** C_f in Figure 12	$i\mathcal{O}$ security
		Hyb_5	the fake ciphertext C_f is the obfuscations of the programs Fake-Cipher-II-L* and Fake-Cipher-II-R** in Figure 13	$i\mathcal{O}$ security
Fake→ Normal	Normal	Hyb_6	$t_i = \mathsf{PRG}(s_i)$, where s_i is randomly chosen, for all $i \in \mathbb{H} \cup \{i^*\}$	security of PRG
		Hyb_7	use $\mathsf{AuthGen}(\lambda)$ for user i^*	security of authentication

Table 3. Description of the hybrid experiments in the proof of Theorem 2

Proof. The proposed construction allows one-pair-of-keys collusion attacks, that is $|\mathbb{C} \cap \mathbb{D}| = 1$, where $\mathbb{C}$ denotes the set of corrupted users and $\mathbb{D}$ denotes the set of delegatees whose corresponding re-encryption keys are corrupted. Let $\{j^*\} = \mathbb{C} \cap \mathbb{D}$.

For any polynomial time algorithm $\mathcal{A}$ which outputs an ϵ-useful decryption box $L_{i^*,\epsilon}$ for user i^*, we construct a PPT algorithm $L_{j^*,\epsilon}$ which uses $L_{i^*,\epsilon}$ as a subroutine to invert any given second level ciphertext $C^* \leftarrow \mathsf{Enc}_2(\mathsf{pk}_{j^*}, m)$ of user j^* with probability ϵ, where $m \leftarrow \mathsf{M}$.

Target-Cipher-II-L

Constants: m, t_{j^*}.
Input: s_{j^*}.

1. If $\mathsf{PRG}(s_{j^*}) = t_{j^*}$, output m and exit.
2. Output "$\perp$".

Target-Cipher-II-R

Constants: m, avk_{j^*}.
Input: $(\mathsf{rk}_{j^*\to j'}, s_{j'}, t_{j'})$.

1. If $\mathsf{Verify}(\mathsf{avk}_{j^*}, t_{j'}, \mathsf{rk}_{j^*\to j'}) = 1$ and $\mathsf{PRG}(s_{j'}) = t_{j'}$ output m and exit.
2. Output "$\perp$".

Fig. 9. The programs Target-Cipher-II-L and Target-Cipher-II-R (of user j^*)

To complete the proof, we describe the rest of the proof as a sequence of hybrid experiments and prove that $L_{j^*,\epsilon}$'s advantage is negligibly close between each successive experiment. In the first hybrid experiment, the key generation is in the normal mode and the ciphertext for extracting j^*'s decryption capability is a fake ciphertext. In Hyb_1 and Hyb_2, we change the key generation algorithm from the normal mode to the fake mode. In Hyb_3 to Hyb_5, the fake ciphertext is changed to a normal one. In Hyb_6 and Hyb_7, the key generation algorithms are back to the normal mode. In the final experiment, the decryption box $L_{j^*,\epsilon}$ would output the encrypted message m with non-negligible probability ϵ. Table 3 roughly describes the hybrid experiments.

Fake-Cipher-II-L

Constants: $0, t_{i^*}$.
Input: s_{i^*}.

1. If $\mathsf{PRG}(s_{i^*}) = t_{i^*}$, output 0 and exit.
2. Output "$\perp$".

Fake-Cipher-II-R

Constants: $i\mathcal{O}$(Target-Cipher-II-L), $\mathsf{avk}_{i^*}, t_{j^*}$.
Input: $\mathsf{rk}_{i^*\to j}, s_j, t_j$.

1. If $t_j = t_{j^*}$ and $\mathsf{Verify}(\mathsf{avk}_{i^*}, t_j, \mathsf{rk}_{i^*\to j}) = 1$, go to step 2; Else, output "$\perp$" and exit.
2. Compute $i\mathcal{O}$(Target-Cipher-II-L) with s_j as input.
3. Output "$\perp$".

Fig. 10. The programs Fake-Cipher-II-L and Fake-Cipher-II-R (of user i^*)

- **Hyb**$_0$ The first hybrid experiment is the real experiment.
  1. The challenger generates the keys $\{\mathsf{pk}_j, \mathsf{sk}_j\}_{j\in\mathbb{C}}$, $\{\mathsf{pk}_j\}_{j\in\mathbb{H}}$, $\{\mathsf{rk}_{i^*\to j}\}_{j\in\mathbb{D}}$ and pk_{i^*} as follows.
     (a) For all $j \in \mathbb{C}$, pick $(\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{KeyGen}(\lambda)$.
     (b) For all $j \in \mathbb{H}$, pick s_j at random and compute $t_j = \mathsf{PRG}(s_j)$. Pick $(\mathsf{avk}_j, \mathsf{ask}_j) \leftarrow \mathsf{AuthGen}(\lambda)$. Set $\mathsf{pk}_j = (t_j, \mathsf{avk}_j)$.
     (c) For user i^*, pick s_{i^*} at random and compute $t_{i^*} = \mathsf{PRG}(s_{i^*})$. Pick $(\mathsf{avk}_{i^*}, \mathsf{ask}_{i^*}) \leftarrow \mathsf{AuthGen}(\lambda)$. Set $\mathsf{pk}_{i^*} = (t_{i^*}, \mathsf{avk}_{i^*})$.
     (d) For all $j \in \mathbb{D}$, generate $\mathsf{rk}_{i^*\to j} \leftarrow \mathsf{AuthProve}(\mathsf{ask}_{i^*}, t_j)$.

     It returns $\{\mathsf{pk}_j, \mathsf{sk}_j\}_{j\in\mathbb{C}}$, $\{\mathsf{pk}_j\}_{j\in\mathbb{H}}$, $\{\mathsf{rk}_{i^*\to j}\}_{j\in\mathbb{D}}$ and pk_{i^*}.
  2. $\mathcal{A}$ receives $\{\mathsf{pk}_j, \mathsf{sk}_j\}_{j\in\mathbb{C}}$, $\{\mathsf{pk}_j\}_{j\in\mathbb{H}}$, $\{\mathsf{rk}_{i^*\to j}\}_{j\in\mathbb{D}}$ and pk_{i^*}. It outputs an ϵ-useful decryption box $L_{i^*,\epsilon}$ of user i^*.
  3. I generates a PPT algorithm $L_{j^*,\epsilon}$ which runs $L_{i^*,\epsilon}$ as a subroutine. It performs as follows.
     (a) When given $C^* = (i\mathcal{O}(\text{Target-Cipher-II-L}), i\mathcal{O}(\text{Target-Cipher-II-R}))$ as shown in Figure 9 where $m \leftarrow \mathsf{M}$, use $i\mathcal{O}$(Target-Cipher-II-L) to generate the programs Fake-Cipher-II-L and Fake-Cipher-II-R in Figure 10.
     (b) Obfuscate the programs and generate a fake ciphertext of user i^*

$$C_f = (i\mathcal{O}(\text{Fake-Cipher-II-L}), i\mathcal{O}(\text{Fake-Cipher-II-R})).$$

     (c) Run $L_{i^*,\epsilon}$ with C_f as input and output what $L_{i^*,\epsilon}$ outputs.
- **Hyb**$_1$ This hybrid experiment is the same as Hyb$_0$ except that we change the key generation algorithm for user i^* to use $\mathsf{FakeAuthGen}(\lambda, \{t_j\}_{j\in\mathbb{D}})$ instead of $\mathsf{AuthGen}(\lambda)$.
  – This hybrid is indistinguishable from the previous one by the security of the authentication scheme.
- **Hyb**$_2$ This hybrid experiment is the same as Hyb$_1$ except that t_i is chosen at random for all $i \in \mathbb{H} \cup \{i^*\}$.
  – The indistinguishability of Hyb$_2$ from Hyb$_1$ follows from the pseudo-randomness property of PRG.
- **Hyb**$_3$ This hybrid experiment is the same as Hyb$_2$ except that we let the fake ciphertext C_f be the obfuscations of the programs Fake-Cipher-II-L and Fake-Cipher-II-R* in Figure 11.
  – In this hybrid experiment, we modify the fake ciphertext C_f by replacing $i\mathcal{O}$(Target-Cipher-II-L) in program Fake-Cipher-II-R with the corresponding unobfuscated program without changing its functionality. Thus, Hyb$_3$ and Hyb$_2$ are indistinguishable from the $i\mathcal{O}$ security.
- **Hyb**$_4$ This hybrid experiment is the same as Hyb$_3$ except that we let the fake ciphertext C_f be the obfuscation of the program Fake-Cipher-II-L and Fake-Cipher-II-R** in Figure 12.
  – In this hybrid experiment, we remove the check $t_j = t_{j^*}$ in the program Fake-Cipher-II-R*. Since $(\mathsf{avk}_{i^*}, \mathsf{ask}_{i^*}) \leftarrow \mathsf{FakeAuthGen}(\lambda, \{t_j\}_{j\in\mathbb{D}})$, with high probability it is true that there is no authentication for user $j \notin \mathbb{D}$. Since for all $j \in \mathbb{H}$, t_j is chosen at random, with high probability it is true t_j is

Fake-Cipher-II-L

Constants: $0, t_{i^*}$.
Input: s_{i^*}.
1. If $\mathsf{PRG}(s_{i^*}) = t_{i^*}$, output 0 and exit.
2. Output "$\perp$".

Fake-Cipher-II-R*

Constants: $m, \mathsf{avk}_{i^*}, t_{j^*}$.
Input: $\mathsf{rk}_{i^*\to j}, s_j, t_j$.
1. If $t_j = t_{j^*}$ and $\mathsf{Verify}(\mathsf{avk}_{i^*}, t_j, \mathsf{rk}_{i^*\to j}) = 1$, go to step 2; Else, output "$\perp$" and exit.
2. If $\mathsf{PRG}(s_j) = t_j$ output m and exit.
3. Output "$\perp$".

Fig. 11. The programs Fake-Cipher-II-L and Fake-Cipher-II-R* (of user i^*)

not in the image of the PRG. Therefore with high probability it is true that only when $j \in \mathbb{C} \cap \mathbb{D}$ (i.e. $t_j = t_{j^*}$) could the tuple $(\mathsf{rk}_{i\to j}, s_j, t_j)$ pass the check $t_j = t_{j^*}$. Thus, the remove will not change the functionality of the program Fake-Cipher-II-R*. This hybrid is indistinguishable from the previous one by the $i\mathcal{O}$ security.

Fake-Cipher-II-L

Constants: $0, t_{i^*}$.
Input: s_{i^*}.
1. If $\mathsf{PRG}(s_{i^*}) = t_{i^*}$, output 0 and exit.
2. Output "$\perp$".

Fake-Cipher-II-R**

Constants: m, avk_{i^*}.
Input: $\mathsf{rk}_{i^*\to j}, s_j, t_j$.
1. If $\mathsf{Verify}(\mathsf{avk}_{i^*}, t_j, \mathsf{rk}_{i^*\to j}) = 1$ and $\mathsf{PRG}(s_j) = t_j$ output m and exit.
2. Output "$\perp$".

Fig. 12. The programs Fake-Cipher-II-L and Fake-Cipher-II-R** (of user i^*)

• **Hyb**$_5$ This hybrid experiment is the same as Hyb$_4$ except that we let the fake ciphertext C_f be the obfuscation of the programs Fake-Cipher-II-L* and Fake-Cipher-II-R** in Figure 13.
– Since t_{i^*} is chosen at random, with high probability it is true t_{i^*} is not in the image of the PRG and the program Fake-Cipher-II-L will output "$\perp$". Therefore, the replacement will not change the functionality of the program Fake-Cipher-II-L. Thus, the indistinguishability of Hyb$_5$ and Hyb$_4$ follows from the $i\mathcal{O}$ security property.

Fake-Cipher-II-L*

Constants: m, t_{i^*}.
Input: s_{i^*}.
1. If $\mathsf{PRG}(s_{i^*}) = t_{i^*}$, output m and exit.
2. Output "$\perp$".

Fake-Cipher-II-R**

Constants: m, avk_{i^*}.
Input: $\mathsf{rk}_{i^*\to j}, s_j, t_j$.
1. If $\mathsf{Verify}(\mathsf{avk}_{i^*}, t_j, \mathsf{rk}_{i^*\to j}) = 1$ and $\mathsf{PRG}(s_j) = t_j$ output m and exit.
2. Output "$\perp$".

Fig. 13. The programs Fake-Cipher-II-L* and Fake-Cipher-II-R** (of user i^*)

• **Hyb**$_6$ This hybrid experiment is the same as Hyb$_5$ except that for all $i \in \mathbb{H} \cup \{i^*\}$ we compute $t_i = \mathsf{PRG}(s_i)$, where s_i is randomly chosen.
– The indistinguishability of Hyb$_6$ and Hyb$_5$ follows from the pseudo-randomness property of PRG.

• **Hyb**$_7$ This hybrid experiment is same as Hyb$_6$ except that we change the key generation algorithm for i^* to use $\mathsf{AuthGen}(\lambda)$.
– This hybrid is indistinguishable from the previous one by the security of the authentication scheme.

In the last hybrid, C_f is a normal ciphertext that encrypts m for user i^*. Since $L_{i^*,\epsilon}$ is an ϵ-useful decryption box for user i^*, the probability that $L_{j^*,\epsilon}$ would output m' such that $m' = m$ is ϵ. Therefore, in the real experiment $L_{j^*,\epsilon}$ would be an ϵ-useful second level decryption box with overwhelming probability.
Thus, the theorem follows. □

6 Impossibility Result for Non-Transferability

In this section, we show that a PRE scheme which is error-free cannot achieve non-transferability.

The notion of error-freeness is a strong correctness requirement introduced by Zhang et al. [23] to achieve CCA security for PRE schemes. Many existing schemes such as [2][7][18] are actually error-free. It is also worth noting that error-freeness requires the correctness of re-encryption is preserved for all ciphertexts of Alice, even including malformed ciphertexts, while the correctness of a PRE scheme only requires the correctness of re-encryption is preserved for all valid ciphertexts of Alice.

Definition 8 (Error-Freeness). A PRE scheme Π is said to be error-free, if for any second level ciphertext C_i under pk_i, $m = \mathsf{Dec}_2(\mathsf{sk}_i, C_i)$, and any re-encryption key $\mathsf{rk}_{i\to j}$, we have the probability

$$\Pr[\mathsf{Dec}_1(\mathsf{sk}_j, \mathsf{ReEnc}(\mathsf{rk}_{i\to j}, C_i)) = m] = 1.$$

Theorem 3. *For any CPA secure PRE scheme, if it is error-free, then it cannot be non-transferable.*

Proof. Assume $\Pi' = (\mathsf{KeyGen}', \mathsf{ReKeyGen}', \mathsf{Enc}'_1, \mathsf{Enc}'_2, \mathsf{ReEnc}', \mathsf{Dec}'_1, \mathsf{Dec}'_2)$ is a CPA secure and error-free PRE scheme. Suppose user i^* delegates his decryption rights to user j^*, and an adversary $\mathcal{A}$ corrupts the proxy and user j^*. $\mathcal{A}$'s goal is to construct an ϵ-useful decryption box $L_{i^*,\epsilon}$ such that $L_{i^*,\epsilon}$ doesn't contain any useful information for decrypting any ciphertext of user j^*.

When $\mathcal{A}$ obtains $\mathsf{pk}_{i^*}, \mathsf{pk}_{j^*}, \mathsf{sk}_{j^*}, \mathsf{rk}_{i^*\to j^*}$, it generates $L_{i^*,\epsilon}$ as the obfuscation of the program Transfer in Figure 14. It is obvious that $L_{i^*,\epsilon}$ is useful in decrypting user i^*'s ciphertexts. The rest is to see that $L_{i^*,\epsilon}$ brings no advantages for decrypting j^*'s ciphertexts. Consider another program $L'_{i^*,\epsilon}$ which is an obfuscation of the program Transfer* in Figure 15.

Transfer

Constants: $\mathsf{sk}_{j^*}, \mathsf{rk}_{i^*\to j^*}$.
Input: C_{i^*}.

1. Output $m = \mathsf{Dec}_1(\mathsf{sk}_{j^*}, \mathsf{ReEnc}(\mathsf{rk}_{i^*\to j^*}, C_{i^*}))$.

Fig. 14. The program Transfer

The output $m = \mathsf{Dec}_1(\mathsf{sk}_{j^*}, \mathsf{ReEnc}(\mathsf{rk}_{i^*\to j^*}, C_{i^*}))$ in the program Transfer is replaced by $m = \mathsf{Dec}_2(\mathsf{sk}_{i^*}, C_{i^*})$ in the program Transfer*. Since Π' is error-free, the replacement would not change the functionality of the program Transfer. Thus, $L_{i^*,\epsilon}$ is indistinguishable from $L'_{i^*,\epsilon}$ by the $i\mathcal{O}$ security.

Since $L'_{i^*,\epsilon}$ is generated without sk_{j^*} and Π' is CPA secure, the probability of using $L'_{i^*,\epsilon}$ to generate j^*'s decryption box $L_{j^*,\epsilon}$ is negligible, which completes our proof. □

Transfer*

Constants: sk_{i^*}.
Input: C_{i^*}.

1. Output $\underline{m = \mathsf{Dec}_2(\mathsf{sk}_{i^*}, C_{i^*})}$.

Fig. 15. The program Transfer*

7 Conclusion

Non-transferability is an intriguing problem both in practice and theory. In this paper, we formalized the notion of non-transferability and proposed the first non-transferable PRE construction based on indistinguishability obfuscation and k-unforgeable authentication scheme, which allows to leak one pair of keys consisting of a re-encryption key and the related delegatee's secret key. Indeed, the realization of non-transferable PRE scheme answers the open problem proposed by Ateniese et al. in 2005. Also, we proved the impossibility of non-transferability for "error-free" PRE schemes. Our negative result rules out a large class of construction methods for achieving non-transferability. Since this is the first step in formal investigation of non-transferable PRE, it would be interesting to see other methodologies for concrete constructions, e.g., getting rid of $i\mathcal{O}$.

References

1. Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. In *NDSS*, 2005.
2. Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security (TISSEC)*, 9(1):1–30, 2006.
3. Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *Advances in cryptology–CRYPTO 2001*, pages 1–18. Springer, 2001.
4. Mihir Bellare and Igors Stepanovs. Point-function obfuscation: A framework and generic constructions. 2015. http://eprint.iacr.org/2015/703.pdf.
5. Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In *Advances in Cryptology-EUROCRYPT 1998*, pages 127–144. Springer, 1998.
6. Dan Boneh, Divya Gupta, Ilya Mironov, and Amit Sahai. Hosting services on an untrusted cloud. In *Advances in Cryptology-EUROCRYPT 2015*, pages 404–436. Springer, 2015.
7. Ran Canetti and Susan Hohenberger. Chosen-ciphertext secure proxy re-encryption. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 185–194. ACM, 2007.
8. Nishanth Chandran, Melissa Chase, Feng-Hao Liu, Ryo Nishimaki, and Keita Xagawa. Re-encryption, functional re-encryption, and multi-hop re-encryption: A framework for achieving obfuscation-based security and instantiations from lattices. In *Public Key Cryptography–PKC 2014*, pages 95–112. Springer, 2014.
9. Nishanth Chandran, Melissa Chase, and Vinod Vaikuntanathan. Functional re-encryption and collusion-resistant obfuscation. In *Theory of Cryptography*, pages 404–421. Springer, 2012.
10. Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *Advances in Cryptology-Eurocrypt 2013*, volume 7881, pages 1–17. Springer, 2013.
11. Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 40–49. IEEE, 2013.
12. Hui Guo, Zhenfeng Zhang, and Jiang Zhang. Proxy re-encryption with unforgeable re-encryption keys. In *Cryptology and Network Security*, pages 20–33. Springer, 2014.
13. Ryotaro Hayashi, Tatsuyuki Matsushita, Takuya Yoshida, Yoshihiro Fujii, and Koji Okada. Unforgeability of re-encryption keys against collusion attack in proxy re-encryption. In *Advances in Information and Computer Security*, pages 210–229. Springer, 2011.

14. Susan Hohenberger, Guy N Rothblum, Abhi Shelat, and Vinod Vaikuntanathan. Securely obfuscating re-encryption. In *Theory of Cryptography*, pages 233–252. Springer, 2007.
15. Toshiyuki Isshiki, Manh Ha Nguyen, and Keisuke Tanaka. Attacks to the proxy re-encryption schemes from IWSEC2011. In *Advances in Information and Computer Security*, pages 290–302. Springer, 2013.
16. Ravi Kumar, Sridhar Rajagopalan, and Amit Sahai. Coding constructions for blacklisting problems without computational assumptions. In *Advances in Cryptology-CRYPTO 1999*, pages 609–623. Springer, 1999.
17. Benoît Libert and Damien Vergnaud. Tracing malicious proxies in proxy re-encryption. In *Pairing-Based Cryptography-Pairing 2008*, pages 332–353. Springer, 2008.
18. Benoît Libert and Damien Vergnaud. Unidirectional chosen-ciphertext secure proxy re-encryption. In *Public Key Cryptography-PKC 2008*, pages 360–379. Springer, 2008.
19. Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 475–484. ACM, 2014.
20. Gelareh Taban, Alvaro A Cárdenas, and Virgil D Gligor. Towards a secure and interoperable drm architecture. In *Proceedings of the ACM workshop on Digital rights management*, pages 69–78. ACM, 2006.
21. Lihua Wang, Licheng Wang, Masahiro Mambo, and Eiji Okamoto. New identity-based proxy re-encryption schemes to prevent collusion attacks. In *Pairing 2010*, pages 327–346. Springer, 2010.
22. Lei Xu, Xiaoxin Wu, and Xinwen Zhang. A certificateless proxy re-encryption scheme for secure data sharing with public cloud. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, pages 1–10. ACM, 2012.
23. Jiang Zhang, Zhenfeng Zhang, and Yu Chen. PRE: Stronger security notions and efficient construction with non-interactive opening. *Theoretical Computer Science*, 542(3):1–16, 2014.