Verifiable Proxy Re-encryption from Indistinguishability Obfuscation

Muhua Liu[1], Ying Wu[1], Jinyong Chang[1], Rui Xue[1(✉)], and Wei Guo[2]

[1] State Key Laboratory of Information Security,
Institute of Information Engineering, Chinese Academy of Sciences,
Beijing 100093, China
{liumuhua,wuying,changjinyong,xuerui}@iie.ac.cn
[2] Information Center, Guangdong Power Grid Company Limited,
Guangzhou 510000, Guangdong, China
guowei@gdxx.csg.cn

Abstract. Proxy re-encryption scheme allows a semi-trusted proxy to re-encrypt ciphertexts of a client into ciphertexts of a receiver. However, the proxy might not as honest or reliable as supposed in practice.

Ohata et al. [16] recently introduced a new functionality for proxy re-encryption with verifiability of the re-encryption soundness. However, careful inspection reveals that the construction in their work can not resist against normal collusion attack. Specifically, if the proxy and the receiver collude, the master key of the client will be leaked. We consider this as a serious weakness. Moreover, the storage of keys for a receiver in that work will increase linearly with the number of clients.

In this paper, we present a novel generic scheme for verifiable proxy re-encryption from indistinguishability obfuscation. It can ensure the security of master secret key even when the proxy and the receiver collude. In addition, our scheme possesses the advantage that any receiver's key storage will remain constant, no matter how many clients he deals with. Furthermore, the re-encryption mechanism in our construction is very succinct in that the size of re-encrypted ciphertext relies only on the size of the encrypted message and the security parameter, compared to that in [16], which relies on the size of the original ciphertext and the receiver's public key as well.

Keywords: Verifiability · Indistinguishability obfuscation · Proxy re-encryption

1 Introduction

There are many applications that require to covert encrypted messages of one client, per a proxy, into ciphertexts of a receiver, such as secure file systems, outsourced filtering of encrypted spam [1]. Blaze et al. [3] proposed the notion of atomic proxy cryptography, which allows a semi-trusted proxy to convert a ciphertext of a client into a ciphertext of a receiver without seeing the underlying plaintext and the secret key of either the client or the receiver.

S. Qing et al. (Eds.): ICICS 2015, LNCS 9543, pp. 363–378, 2016.
DOI: 10.1007/978-3-319-29814-6_31

In the ordinary proxy re-encryption schemes, the proxy is modeled as a semi-trusted party, who will perform the re-encryption algorithm honestly. But, in some real scenarios, the proxy is not as honest or reliable as it might be. Considering the following scenario, for example, a client stores a large encrypted data CT in the cloud. When he wants to share his encrypted data with a receiver, he will give his transformation key TK to a cloud proxy. Then, the proxy may convert CT into a re-encrypted ciphertext CT_{out} that can be decrypted under the receiver's secret key. However, the proxy may make mistakes for various reasons like a faulty implementation of the re-encryption algorithm, or returning a random result even for saving computation time. Therefore, a proxy re-encryption protocol that allows the receiver to make verification of the correctness of the re-encrypted ciphertexts is much expected.

Ohata et al. [16] recently proposed a verifiable proxy re-encryption protocol. In the protocol, the receiver is enhanced with the function to verify whether a received ciphertext is correctly transformed from an original ciphertext by a proxy, and thus can detect illegal actives of the proxy. The scheme achieves re-encryption verifiability by adding a re-encryption verifiable algorithm, which takes two ciphertexts CT and CT_{out}, a secret key sk_R of a receiver, and a public key pk of a client as input, and allows to check the faithfulness of the transformation of CT_{out}.

The idea of the construction is as follows: the client splits his secret key sk into two shares sk_1 and sk_2 by a threshold public key encryption scheme [5], and sends the original ciphertext CT and sk_1 to the proxy, and $\phi = \mathsf{Enc}_{pk_R}(sk_2)$ to the receiver. Then the proxy re-encrypts the ciphertext CT by $\mu_1 = \mathsf{Dec}_{sk_1}(CT)$ and $CT_{out} = \mathsf{Enc}_{pk_R}(\mu_1 \parallel CT)$. After getting the ciphertexts ϕ and CT_{out}, the receiver computes $\mu_1 \parallel CT = \mathsf{Dec}_{sk_R}(CT_{out})$ and $\mu_2 = \mathsf{Dec}_{sk_2}(CT)$, and outputs a plaintext m by the combining algorithm of the threshold public key encryption scheme. In order to achieve proxy re-encryption with re-encryption verifiability, their scheme complies with the following properties: When re-encrypting CT into CT_{out}, CT is somehow embedded into CT_{out} in such a way that when the receiver decrypts CT_{out}, the embedded ciphertext CT can be extracted. In re-encryption verification, the receiver checks whether an extracted ciphertext CT equals to the given candidate CT'.

In the above scheme, if the proxy and the receiver collude together, the two shares sk_1 and sk_2 can be obtained that could further recover the secret key of the client. Another drawback in their scheme is that the receiver has to store each secret key sk_2 to decrypt the received ciphertexts of one client. When the receiver deals with a couple of clients, the key storage of the receiver grows linearly with the number of clients. In addition, the size of the re-encrypted ciphertext in their construction not only relies on the size of the encrypted message and the security parameter, but also relies on the size of the original ciphertext and the receiver's public key. That is not preferred when a large amount of messages need to be processed.

In this paper, we present a novel verifiable proxy re-encryption scheme that makes up the drawbacks as above. We focus on the unidirectional non-interactive

verifiable proxy re-encryption scheme. Here is our desired properties for a verifiable proxy re-encryption scheme:

- **Verifiability:** A malicious proxy can not persuade a receiver to accept a wrong re-encrypted ciphertext.
- **Weak Master Secret Key Security:** An adversary can not get the master secret key even if the proxy and the receiver collude. In our scheme, we just consider the adversary which only gets one transformation key. We call this as weak master secret key security. One motivation [1], to consider this stems from some proxy re-encryption schemes define two or more types of ciphertext, some of which may only be decrypted using the master secret key. A scheme which provides the master key security will protect those ciphertexts.
- **Key Optimality:** The size of the receiver's key storage remains constant, regardless of how many clients from which he receives ciphertext.
- **Succinct:** The size of the re-encrypted ciphertext just relies on the size of the encrypted message and the security parameter.
- **Efficiency:** The time complexity of the verifiable algorithm should be smaller than the transformation algorithm. Otherwise, the receiver can complete the transformation process by himself.

The starting idea for our construction is to pick a standard public-key encryption key pair (pk_1, sk_1) and a symmetric encryption scheme in the setup step of the verifiable proxy re-encryption scheme. Choosing a random string K_1 as the secret key of symmetric encryption scheme, an encryption of a message m will simply be an encryption of K_1 using the public key pk_1 and an encryption of m using the symmetric key K_1. Specifically, our approach is built on the key encapsulated mechanism. We compress the random string K_1 to a shorter string using a hash function, and then, use the output of the hash function to check the correctness of the re-encrypted ciphertext. However, the hash value will leak some entropy of the session key which is no longer uniform. Hence, we apply a key extractor to extract a nearly uniform symmetric key K_{SE}, which will replace the random string K_1 as the symmetric encryption key. To verify the integrity of the symmetric encrypted ciphertext, we compute a hash value on the concatenation of the symmetric encrypted ciphertext and the hash value of K_1, and use the second hash value as the verifiable key. The transformation key TK is an obfuscation of a circuit $\mathcal{G}$ which has the master secret key sk_1 hardwired in its description. It takes a original ciphertext as input, and outputs a re-encrypted ciphertext CT_{out}. While this solution would work if the obfuscator achieves the black box obfuscation definition [2], there is no reason to believe that an indistinguishability obfuscator would necessarily hide sk_1.

Recall that the indistinguishability chosen plaintext attack (CPA) security definition requires that the adversary will not be able to distinguish an encryption of m_0 from an encryption of m_1. A natural first step would be to have two public key pk_1 and pk_2, and require the encryption of random string K_1 to consist of encryption of K_1 under both public keys. Then, we encrypt the message m by using the extracted key from the random string K_1, and get the ciphertext

(CT_1, CT_2, CT_3), where CT_1 and CT_2 are the ciphertexts of the random string K_1, and CT_3 is the ciphertext of the message m. However, in this case, the original decryptor (i.e., client) cannot generate a proof on his own that the ciphertexts CT_1 and CT_2 encrypt the same message to provide to the obfuscator transformation circuit. Thus, we must require the encryptor to generate a proof π , which must hide K_1. A solution is to have the encryptor generate a non-interactive witness indistinguishable proof. One statement is that the ciphertexts CT_1 and CT_2 are encryptions of the same message, the other statement is a commitment to $CT_1 \parallel CT_2$. After getting the ciphertext $CT = (CT_1, CT_2, CT_3, \pi)$, the obfuscated circuit firstly check the proof π, if it checks out, it would use secret key sk_1 for decryption and computation.

Note that the encryption algorithm of our construction is not efficient compared with Ohata's work [16], because of using the tool of NIWI. Our scheme just achieves the CPA security. To construct a scheme which satisfies the CCA security is our future work.

1.1 Related Works

Mambo and Okamoto [14] proposed the notion of proxy encryption, which delegates the ability of decryption through an interaction. Blaze et al. [3] proposed the first bidirectional proxy re-encryption scheme based on the ElGamal encryption scheme. Their construction is CPA secure under the Decisional Diffie-Hellman assumption. However, their scheme is not master secret key secure, and if the proxy and the receiver collude, they can recover client's secret key.

Ivan and Dodis [11] proposed an unidirectional non-interactive proxy encryption for ElGamal encryption scheme by sharing the client's secret to two participants. Their construction also can not achieve the master secret key security and key optimality. Ateniese et al. [1] proposed an unidirectional proxy re-encryption scheme based on bilinear maps. Their construction is weak master secret key secure, efficient, and chosen-plaintext secure under the Bilinear Diffie-Hellman assumption.

Canetti and Hohenberger [4] described a bidirectional construction of proxy re-encryption providing chosen-ciphertext security. Libert and Vergnaud [13] presented the first unidirectional proxy re-encryption schemes with replayable chosen-ciphertext security in the standard model. Shao and Cao [18] proposed the first CCA-secure proxy re-encryption without pairings. But their construction is secure in the random oracle model. Subsequently, Matsuda et al. [15] improved Shao's [18] result and constructed a bidirectional chosen-ciphertext security proxy re-encryption scheme without bilinear maps in the standard model. But later, Weng et al. [19] pointed out that Matsuda's [15] scheme is not chosen-ciphertext secure. Hohenberger et al. [9] proposed a novel proxy re-encryption scheme based on the obfuscation, but their security is weak since the adversary is only allowed to black-box access to re-encryption oracle. Hanaoka et al. [8] presented the first generic construction of chosen-ciphertext secure unidirectional proxy re-encryption scheme based on threshold public key encryption [5]. Isshiki et al. [10] proposed a proxy re-encryption scheme in a stronger security model

extended from Hanaoka's scheme [8]. Recently, Kirshanova [12] proposed a new unidirectional proxy scheme based on the hardness of the LWE problem. Their scheme is provably CCA-1 secure in the selective model. However, none of these work considered the re-encryption verifiability. Ohata et al. [16] firstly introduced this property, but their construction can not resist the attack of collusion. In this work, we give a novel construction which has some better properties than the construction of Ohata et al. [16].

1.2 Organization

The rest of this paper is organized as follows. We start by giving the definitions for verifiable proxy re-encryption in Sect. 2. Next, in Sect. 3, we recall the definitions for various cryptographic primitives used in our construction. We then present our construction for verifiable proxy re-encryption scheme in Sect. 4. Finally, we give the conclusion of this work in Sect. 5.

Basic Notation. In what follows we will denote with $\lambda \in \mathbb{N}$ a security parameter. We say $\mathrm{negl}(\lambda)$ is negligible if $|\mathrm{negl}(\lambda)| < 1/\mathrm{poly}(\lambda)$ holds for all polynomials $\mathrm{poly}(\lambda)$ and all sufficiently large λ. Denote PPT as probabilistic polynomial time. "$x \parallel y$" denotes a concatenation of x and y. We write $x \xleftarrow{R} X$ for sampling x from the set X uniformly at random.

2 Verifiable Proxy Re-encryption

In this section, we give the definition of verifiable proxy re-encryption which is different from the definition of Ohata's [16]. Because our verifiable algorithm needs additional information to verify the correctness of the re-encrypted ciphertext, we adopt a new definition. We start by presenting the syntax for verifiable proxy re-encryption and then proceed to give the security definitions.

Syntax. Denote PKE_R as the receiver's encryption system, and the key pairs (pk_R, sk_R) as the receiver's key. A verifiable proxy re-encryption $\mathcal{VPRE}$ consists of five algorithms $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Trans}, \mathsf{Dec})$:

- $\mathsf{Setup}(1^\lambda) \rightarrow (mpk, msk)$: The setup algorithm takes as input a security parameter λ and outputs the master public key mpk and the master secret key msk.
- $\mathsf{Enc}(mpk, m) \rightarrow (CT, VK)$: The encryption algorithm takes as input the master public key mpk and a message $m \in \mathcal{M}$, and outputs a ciphertext CT and a verifiable key VK.
- $\mathsf{KeyGen}(pk_R, msk) \rightarrow TK$: The transformation key generation algorithm takes as input the master secret key msk and the receiver's public key pk_R, and outputs a transformation key TK.
- $\mathsf{Trans}(CT, TK) \rightarrow CT_{out}$: The ciphertext transformation algorithm takes as input a ciphertext CT and a transformation key TK, and outputs a re-encrypted ciphertext CT_{out}.

– $\mathsf{Dec}(VK, sk_R, CT_{out}) \rightarrow m$ or $\perp$: The decryption algorithm takes as input the verifiable key VK, the ciphertext CT_{out}, and the receiver's secret key sk_R, and outputs a message m or $\perp$, where $\perp$ indicates that the transformation ciphertext is invalid.

Definition 1 ***(Correctness).*** *A verifiable proxy re-encryption scheme $\mathcal{VPRE}$ is correct if the ciphertext generated by the encryption algorithm allows an honest worker to output a transformation ciphertext that will be decrypted as the original message. More formally, for any $m \in \mathcal{M}$, any $(mpk, msk) \leftarrow \mathsf{Setup}(1^\lambda)$, $(CT, VK) \leftarrow \mathsf{Enc}(mpk, m)$, and any $TK \leftarrow \mathsf{KeyGen}(pk_R, msk)$, if $CT_{out} \leftarrow \mathsf{Trans}(CT, TK)$, then $m \leftarrow \mathsf{Dec}(sk_R, VK, CT_{out})$ holds with all but negligible probability.*

CPA Security Experiment. We adopt the chosen plaintext attack (CPA) security for the verifiable proxy re-encryption scheme. The game is described as follows:

Setup: A challenger firstly generates the honest receiver's key pairs $(sk_{R_i}, pk_{R_i}) \leftarrow \mathsf{PKE}_{R_i}.\mathsf{Setup}(1^\lambda)$ for $i = i, \ldots, \ell$, and sets $PK = \{pk_{R_i}\}_{i=1}^{\ell}$. Then, the challenger runs the $\mathsf{Setup}(1^\lambda)$ to generate a master public key mpk and a master secret key msk. It gives the master public key mpk and the public key set $PK = \{pk_{R_i}\}_{i=1}^{\ell}$ to the adversary $\mathcal{A}$.

Phase 1: Proceeding adaptively, the adversary can repeatedly make the transformation key generation queries:

– The adversary $\mathcal{A}$ submits the pk_{R_i} to the challenger.
– If $pk_{R_i} \notin \{pk_{R_i}\}_{i=1}^{\ell}$, then the challenger outputs $\perp$. Else, the challenger runs the $\mathsf{KeyGen}(pk_{R_i}, msk)$ to generate the transformation key TK_i, and sends TK_i to the adversary.

Challenge: $\mathcal{A}$ submits two equal-length messages m_0 and m_1 to the challenger. The challenger randomly picks a bit $b \in \{0, 1\}$, and runs $\mathsf{Enc}(mpk, m_b)$ to obtain a challenge ciphertext CT^* and a verification key VK^* of the message m_b. It returns (CT^*, VK^*) to the adversary.

Phase 2: This phase is the same as Phase 1.

Guess: $\mathcal{A}$ outputs a guess b'.

Definition 2 ***(CPA security).*** *A $\mathcal{VPRE}$ scheme is CPA secure if for any PPT adversary $\mathcal{A}$,*

$$\mathrm{Adv}^{CPA}_{\mathcal{VPRE},\mathcal{A}}(\lambda) := \left| \Pr[b' = b] - \frac{1}{2} \right| \leq \mathrm{negl}(\lambda).$$

Weak Master Secret Key Security. We require that the adversary can not get the master secret key even if the proxy and the receiver collude. A $\mathcal{VPRE}$

scheme is weak master secret key secure if the adversary can not learn the master secret key of the client, even when he is given the transformation key and the receiver's secret key. The adversary is allowed to query the transformation key generation oracle only once. The game is described as follows:

Setup: A challenger runs the $\mathsf{Setup}(1^\lambda)$ to generate a master public key mpk and a master secret key msk, and sends the master public key mpk to the the adversary $\mathcal{A}$.

Query: The adversary can make the transformation key generation query one time:

- $\mathcal{A}$ runs $(pk_R, sk_R) \leftarrow \mathsf{PKE}_R.\mathsf{Setup}(1^\lambda)$, and submits the public key pk_R to the challenger.
- The challenger runs the $\mathsf{KeyGen}(msk, pk_R)$ to generate a transformation key TK, and sends TK to $\mathcal{A}$.

Challenge: $\mathcal{A}$ outputs a secret key α. The adversary succeeds if and only if $\alpha = msk$.

Definition 3 ***(Weak Master Secret Key Security).*** *A $\mathcal{VPRE}$ scheme is weak master secret key secure if for any PPT adversary $\mathcal{A}$,*

$$\mathrm{Adv}^{MKS}_{\mathcal{VPRE},\mathcal{A}}(\lambda) := \Pr[\alpha = msk] \leq \mathrm{negl}(\lambda).$$

Verifiability. A $\mathcal{VPRE}$ scheme is verifiable security if a malicious server can not persuade the receiver to accept an incorrect transformation ciphertext, even when he is allowed to query the transformation key generation oracle and the decryption oracle. The game is described as follows:

Setup: First, the challenger generates the honest receiver's key pairs $(sk_{R_i}, pk_{R_i}) \leftarrow \mathsf{PKE}_{R_i}.\mathsf{Setup}(1^\lambda)$ for $i = 1, \ldots, \ell$, and sets $PK = \{pk_{R_i}\}_{i=1}^{\ell}$. Then, the challenger runs $(mpk, msk) \leftarrow \mathsf{Setup}(1^\lambda)$, and sends the master public key mpk and the public key set $PK = \{pk_{R_i}\}_{i=1}^{\ell}$ to the adversary $\mathcal{A}$.

Phase 1: Proceeding adaptively, the adversary can repeatedly make the transformation key generation queries:

- The adversary $\mathcal{A}$ submits the pk_{R_i} to the challenger.
- If $pk_{R_i} \notin \{pk_{R_i}\}_{i=1}^{\ell}$, then the challenger outputs $\perp$. Otherwise, the challenger runs the $\mathsf{KeyGen}(pk_{R_i}, msk)$ to generate the transformation key TK_i, and sends TK_i to the adversary.

Phase 2: The adversary can repeatedly query the decryption oracle:

- The adversary $\mathcal{A}$ submits the (VK, CT_{out}, TK_i) to the challenger.
- The challenger finds the corresponding secret key sk_{R_i}, and computes m or $\perp \leftarrow \mathsf{Dec}(sk_{R_i}, VK, CT_{out})$, and returns the result to $\mathcal{A}$.

Challenge: The adversary $\mathcal{A}$ submits a message m to the challenger. The challenger obtains a challenge ciphertext CT^* and a verification key VK^* of the message m. It returns (CT^*, VK^*) to the adversary.

Phase 3: This phase is the same as Phase 1.

Phase 4: This phase is the same as Phase 2.

Verify: The adversary $\mathcal{A}$ outputs a transformation ciphertext (CT^*_{out}, pk_{R_j}). The challenger finds the corresponding sk_{R_j}, and computes $m' \leftarrow \mathsf{Dec}(sk_{R_j}, VK^*, CT^*_{out})$. If $m' \neq m$ and $m' \neq \perp$, then the adversary $\mathcal{A}$ succeeds, and the game outputs 1.

Definition 4 ***(Verifiability).*** *A $\mathcal{VPRE}$ scheme is verifiable security if for any $m \in \mathcal{M}$, and for any PPT adversary $\mathcal{A}$,*

$$\mathrm{Adv}^{Verify}_{\mathcal{VPRE},\mathcal{A}}(\lambda) := \Pr[\textit{Game outputs 1}] \leq \mathrm{negl}(\lambda).$$

Definition 5 ***(Efficiency).*** *A $\mathcal{VPRE}$ scheme is efficient if for any $(CT, VK) \leftarrow \mathsf{PKE.Enc}(mpk, m)$, and $TK \leftarrow \mathsf{KeyGen}(msk, pk_R)$, the time required to verify the correctness of transformation ciphertext is $o(T)$, where T is the time required to compute $CT_{out} \leftarrow \mathsf{Trans}(CT, TK)$.*

3 Preliminaries

In this section, we present definitions for various cryptographic primitives that we will use in our construction. We assume familiarity with standard semantically secure public-key encryption, standard semantically secure symmetric encryption, and omit their formal definition from this section. For reasons of space, we give the definition of non-interactive witness indistinguishable proofs and commitment schemes in Appendix A. We recall the notions of indistinguishability obfuscation, puncturable pseudorandom functions, and randomness extractor.

3.1 Indistinguishability Obfuscation

We present the formal definition following the syntax of Garg et al. [7]

Definition 6 *(Indistinguishability Obfuscation ($i\mathcal{O}$)). A uniform PPT machine $i\mathcal{O}$ is called an indistinguishability obfuscator for a circuit class $\{\mathcal{C}_\lambda\}$ if the following holds:*

- ***(Correctness):*** *For all security parameters $\lambda \in \mathbb{N}$, $C \in \mathcal{C}_\lambda$, and inputs x, we have that*

$$\Pr[C'(x) = C(x) : C' \leftarrow i\mathcal{O}(\lambda, C)] = 1.$$

– ***(Indistinguishability):*** *For any (not necessarily uniform) PPT distinguisher* $(\mathsf{Samp}, \mathcal{D})$*, there exists a negligible function* negl *such that the following holds: if* $\Pr[\forall x,\ C_0(x) = C_1(x); (C_0, C_1, \sigma) \leftarrow \mathsf{Samp}(1^\lambda)] \geq 1 - \mathrm{negl}(\lambda)$*, then:*

$$\begin{aligned} &|\Pr[\mathcal{D}(\sigma, i\mathcal{O}(\lambda, C_0)) = 1 : (C_0, C_1, \sigma) \leftarrow \mathsf{Samp}(1^\lambda)] \\ &- \Pr[\mathcal{D}(\sigma, i\mathcal{O}(\lambda, C_1)) = 1 : (C_0, C_1, \sigma) \leftarrow \mathsf{Samp}(1^\lambda)]| \leq \mathrm{negl}(\lambda). \end{aligned}$$

Recently, Garg et al. [7] gave the first candidate construction for an indistinguishability obfuscator $i\mathcal{O}$ for the circuit class P/poly.

3.2 Puncturable Pseudorandom Functions

In our construction, we will use the puncturable PRFs, which are PRFs that can be defined on all bit strings of a certain length, except for any polynomial-size set of inputs. Below we recall their definition, as given by Sahai and Waters [17]:

Definition 7. *A puncturable family of PRFs* F *is given by a triple of Turing machines* $\mathsf{Key}, \mathsf{Puncture}, \mathsf{Eval}$*, and a pair of computable functions* $n(\cdot)$ *and* $m(\cdot)$*, satisfying the following conditions:*

– ***(Functionality Preserved Under Puncturing).*** *For every PPT adversary* $\mathcal{A}$ *such that* $\mathcal{A}(1^\lambda)$ *outputs a set* $S \subseteq \{0,1\}^{n(\lambda)}$*, then for all* $x \in \{0,1\}^{n(\lambda)}$ *where* $x \notin S$*, we have that:*

$$\Pr[\mathsf{Eval}(K, x) = \mathsf{Eval}(K_S, x) : K \leftarrow \mathsf{Key}(1^\lambda), K_S = \mathsf{Puncture}(K, S)] = 1.$$

– ***(Pseudorandom at Punctured Points).*** *For every PPT adversary* $(\mathcal{A}_1, \mathcal{A}_2)$ *such that* $\mathcal{A}_1(1^\lambda)$ *outputs a set* $S \subseteq \{0,1\}^{n(\lambda)}$ *and* $x \in S$*, consider an experiment where* $K \leftarrow \mathsf{Key}(1^\lambda)$ *and* $K_S = \mathsf{Puncture}(K, S)$*. Then we have*

$$|\Pr[\mathcal{A}_2(K_S, x, \mathsf{Eval}(K, x)) = 1] - \Pr[\mathcal{A}_2(K_S, x, U_{m(\lambda)}) = 1]| \leq \mathrm{negl}(\lambda),$$

where $U_{m(\lambda)}$ *denotes the uniform distribution over* $m(\lambda)$ *bits.*

3.3 Randomness Extractor

For a discrete distribution X over Σ, we denote its min-entropy by $\mathbf{H}_\infty(X) = -\log(\max_{\sigma \in \Sigma} \Pr[X = \sigma])$. The average min-entropy of X conditioned on Y is defined as $\tilde{\mathbf{H}}_\infty(X|Y) = -\log(E_{y \leftarrow Y}[2^{-\mathbf{H}_\infty(X|Y=y)}])$.

We recall a useful lemma that will be used in our proof.

Lemma 1 *[6]. Let* X, Y *and* Z *be random variables. If* Y *has at most* 2^r *possible values, then* $\tilde{\boldsymbol{H}}_\infty(X|(Y,Z)) \geq \tilde{\boldsymbol{H}}_\infty(X|Z) - r$*.*

Definition 8 ***(Randomness Extractor).*** *An efficient function* $\mathsf{Ext} : \mathcal{X} \times \{0,1\}^t \to \mathcal{Y}$ *is an average-case* (k, ϵ)*-strong extractor if for all random variables* (X, Z) *such that* $X \in \mathcal{X}$ *and* $\tilde{\boldsymbol{H}}_\infty(X|Z) \geq k$*, we have* $\Delta((Z, s, \mathsf{Ext}(X, s)), (Z, s, U_\mathcal{Y})) \leq \epsilon$*, where* $s \xleftarrow{R} \{0,1\}^t, U_\mathcal{Y} \xleftarrow{R} \mathcal{Y}$*, and* $\Delta(\cdot, \cdot)$ *denotes the statistical distance between two distributions.*

By the leftover hash lemma [6], any family of pairwise independent hash functions $\mathcal{H} := \{h : \mathcal{X} \to \mathcal{Y}\}$ is an average case $(\tilde{\mathbf{H}}_\infty(X|Z), \epsilon)$-strong extractor if $\tilde{\mathbf{H}}_\infty(X|Z) \geq \log|\mathcal{Y}| + 2\log(1/\epsilon)$.

4 Construction

In this section we present our construction of the verifiable proxy re-encryption scheme. Our construction relies on the following components: A public key scheme $\mathsf{PKE} = (\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$. A symmetric encryption scheme $\mathsf{SE} = (\mathsf{SE.Enc}, \mathsf{SE.Dec})$ with key space $\{0,1\}^{\ell_{\mathsf{SE}}}$. Two collision-resistant hash functions: $H_0 : \mathcal{K} \to \{0,1\}^{\ell_{H_0}}$, $H_1 : \{0,1\}^* \to \{0,1\}^{\ell_{H_1}}$. A family of pairwise independent hash functions $\mathcal{H}$ from $\mathcal{K}$ to $\{0,1\}^{\ell_{\mathsf{SE}}}$. The above parameters satisfy the following condition: $0 < \ell_{\mathsf{SE}} \leq (\log|\mathcal{K}| - \ell_{H_0}) - 2\log(1/\epsilon)$, where ϵ is a negligible value in λ. Let $len_c = len_c(1^\lambda)$ denote the length of ciphertexts in $(\mathsf{PKE.Setup}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$. We will use a parameter $len = 2 \cdot len_c$ in the description of our scheme.

Let $(\mathsf{NIWI.Setup}, \mathsf{NIWI.Prove}, \mathsf{NIWI.Verify})$ be a NIWI proof system. Let Com be a perfectly binding commitment scheme. Let $i\mathcal{O}$ be an indistinguishability obfuscator for all efficiently computable circuits. Let $(\mathsf{Key}, \mathsf{Puncture}, \mathsf{Eval})$ be a puncturable family of PRF. We now proceed to describe our scheme $\mathcal{VPRE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Trans}, \mathsf{Dec})$.

- $\mathsf{Setup}(1^\lambda) \to (mpk, msk)$:
  1. sample two key pairs for the public key encryption scheme $(sk_1, pk_1) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$, and $(sk_2, pk_2) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$,
  2. compute a CRS $crs \leftarrow \mathsf{NIWI.Setup}(1^\lambda)$ for the NIWI proof system,
  3. choose an extractor $h \in \mathcal{H}$, two hash functions H_0 and H_1, and a symmetric encryption scheme SE,
  4. compute a commitment $C \leftarrow \mathsf{Com}(0^{len})$,
  5. set the master public key to be $mpk = (pk_1, pk_2, h, H_0, H_1, crs, C, \mathsf{SE})$, and the master secret key to be $msk = sk_1$.
- $\mathsf{Enc}(m, mpk) \to (CT, VK)$:
  1. choose a random key K_1, and compute a hash evaluation $\mathsf{Tag} = H_0(K_1)$ of K_1, and an extraction evaluation $K_{\mathsf{SE}} = h(K_1)$,
  2. compute ciphertexts $CT_1 \leftarrow \mathsf{PKE.Enc}(pk_1, K_1; r_1)$, and $CT_2 \leftarrow \mathsf{PKE.Enc}(pk_2, K_1; r_2)$ under the public key encryption scheme, and a ciphertext $CT_3 \leftarrow \mathsf{SE.Enc}(K_{\mathsf{SE}}, m)$ under the symmetric encryption scheme, where r_1 and r_2 are chosen randomly,
  3. compute a NIWI proof $\pi \leftarrow \mathsf{NIWI.Prove}(crs, y, w)$ for the NP statement $y = (CT_1, CT_2, C, pk_1, pk_2)$:

- either CT_1 and CT_2 are encryptions of the same message, or
- C is a commitment to $CT_1 \parallel CT_2$.

A witness $w_{real} = (K_1, r_1, r_2)$ for the first part of the statement, referred to as the real witness, includes the randomness key K_1 and the randomness r_1 and r_2 used to compute the ciphertexts CT_1 and CT_2, respectively. A witness $w_{trap} = s$ for the second part of the statement, referred to as the trapdoor witness, includes the randomness s used to compute C.

4. compute $VK = H_1(\mathsf{Tag} \parallel CT_3)$, and output the ciphertext $CT = (CT_1, CT_2, CT_3, \pi)$ and the verifiable key VK.

– $\mathsf{KeyGen}(pk_R, msk) \rightarrow TK$:
1. choose a fresh PRF key $K \leftarrow \mathsf{Key}(1^\lambda)$,
2. compute the transformation key $TK \leftarrow i\mathcal{O}(\mathcal{G})$, where the circuit $\mathcal{G}$ is described in Fig. 1. Note that $\mathcal{G}$ has the receiver's public key pk_R, the master secret key sk_1, the public key mpk and the PRF key K hardwired in it.

– $\mathsf{Trans}(CT, TK) \rightarrow CT_{out}$:
1. on input CT and a transformation key TK, the transformation algorithm computes and outputs $CT_{out} = \mathcal{G}(CT)$.

– $\mathsf{Dec}(VK, sk_R, CT_{out}) \rightarrow m$ or $\perp$:
1. parse the transformed ciphertext as $CT_{out} = (C_1, C_2)$ and a verifiable key $VK = H_1(\mathsf{Tag} \parallel CT_3)$,
2. recover a random key K_1 from $\mathsf{PKE}_R.\mathsf{Dec}(sk_R, C_1)$,
3. compute $\mathsf{Tag} = H_0(K_1)$, if $H_1(\mathsf{Tag} \parallel C_2) \neq VK$, return $\perp$, otherwise, compute $K_{\mathsf{SE}} = h(K_1)$ and return $m = \mathsf{SE.Dec}(K_{\mathsf{SE}}, C_2)$.

Input: Ciphertext CT
Constants: mpk, sk_1, K, pk_R

1. Parse $CT = (CT_1, CT_2, CT_3, \pi)$.
2. If $\mathsf{NIWI.Verify}(crs, y, \pi) = 0$, then output $\perp$ and stop, otherwise continue to the next step. Here $y = (CT_1, CT_2, C, pk_1, pk_2)$ is the statement corresponding to π.
3. Compute $K_1 \leftarrow \mathsf{PKE.Dec}(sk_1, CT_1)$.
4. Compute $r \leftarrow \mathsf{Eval}(K, CT_1 \parallel CT_2)$.
5. Compute $C_1 \leftarrow \mathsf{PKE}_U.\mathsf{Enc}(pk_R, K_1; r)$ by using the receiver's encryption algorithm, set $C_2 = CT_3$, and output $CT_{out} = (C_1, C_2)$.

Fig. 1. Functionality $\mathcal{G}$

One thing we emphasize is that the transformed ciphertexts in our scheme are succinct in that their size only depends on the message size and security parameter. However, the size of re-encrypted ciphertext depends on the size of original ciphertext and the receiver's public key as well in Ohata's construction [16].

Theorem 1. *The above verifiable proxy re-encryption is CPA secure if it is instantiated with a secure punctured PRF, a CPA secure public key encryption, a CPA secure symmetric encryption, a perfect binding commitment, and indistinguishability secure obfuscator.*

Here we show the intuition of the proof. Suppose the ciphertext $(CT_0 = (CT_{1,0},\ CT_{2,0}, CT_{3,0}, \pi_0), VK_0)$ encrypts the message m_0, and the ciphertext $(CT_1 = (CT_{1,1},\ CT_{2,1}, CT_{3,1}, \pi_1), VK_1)$ encrypts the message m_1. We need to prove that the two ciphertexts are computational indistinguishable. In one of our hybrid experiments, we need to move from an obfuscation that on input $CT = (CT_{1,0}, CT_{2,0}, CT_{3,0}, \pi_0)$ would yield the output $((\mathsf{Enc}(pk_R, K_1; r)), C_2)$ to another obfuscation that on the same input would yield the output $((\mathsf{Enc}(pk_R, K_2; r))$, where the random string r is generated by a pseudorandom function. However, the adversary may not be able to perform such a transformation, since our construction is based on indistinguishability obfuscation, which only guarantees that the obfuscation of circuit that implement identical functions are indistinguishable. Hence, this hybrid change would not be indistinguishable because of $\mathsf{Enc}(pk_R, K_1; r) \neq \mathsf{Enc}(pk_R, K_2; r)$. In order to solve this problem, we introduce five new values that can change the nature of the circuit that we are obfuscating to disable all ciphertexts except for the five ciphertexts. The detailed proof of the theorem is given in the full version of the paper.

Theorem 2. *Suppose that H_0 and H_1 are collision-resistant hash functions, Then, the proxy re-encryption scheme is verifiable security.*

Proof. Given an adversary $\mathcal{A}$ against the verifiable security, we construct an efficient adversary $\mathcal{B}$ to break the collision-resistance of the underlying hash functions H_0 or H_1. Given two challenge hash functions (H_0^*, H_1^*), the adversary $\mathcal{B}$ simulates the verifiability game described as follows.

$\mathcal{B}$ generates honest receiver's key pairs $(sk_{R_i}, pk_{R_i}) \leftarrow \mathsf{PKE}_{R_i}.\mathsf{Setup}(1^\lambda)$ for $i = 1, \ldots, \ell$, and sets $PK = \{pk_{R_i}\}_{i=1}^{\ell}$. Then, the adversary $\mathcal{B}$ generates the public parameter mpk and the master secret key msk as $\mathsf{Setup}(1^\lambda)$, except for hash function H_0^* and H_1^*, and sends the master public key mpk and the public key set PK to the adversary. Note that, the adversary $\mathcal{B}$ knows the master secret key msk and the receiver's secret key set $SK = \{sk_{R_i}\}_{i=1}^{\ell}$. Therefore, it can answer the query of the transformation key generation oracle and the decryption oracle. For a challenge message $m \in \mathcal{M}$ submitted by $\mathcal{A}$, the adversary $\mathcal{B}$ invokes $\mathsf{Enc}(mpk, m)$ to obtain a ciphertext $CT^* = (CT_1^*, CT_2^*, CT_3^*, \pi^*)$ of a random string $K_1^* \in \mathcal{K}$. It then computes $\mathsf{Tag} = H_0^*(K_1^*)$ and $VK^* = H_1^*(\mathsf{Tag} \parallel CT_3)$, and sends (CT^*, VK^*) to the adversary $\mathcal{A}$. The adversary $\mathcal{A}$ outputs $CT_{out} = (C_1, C_2)$. $\mathcal{B}$ computes the random string $K_1' \leftarrow \mathsf{PKE}_R.\mathsf{Dec}(sk_R, C_1)$ and $\mathsf{Tag}' = H_0^*(K_1')$. We observe that $\mathcal{A}$ succeeds if and only if $m' \notin \{m, \perp\}$ and $H_1^*(\mathsf{Tag}' \parallel C_2) = VK^*$. If $\mathcal{A}$ succeeds, we consider the following two cases:

Case 1. $(\mathsf{Tag}' \parallel C_2) \neq (\mathsf{Tag} \parallel CT_3)$. If this case occurs, $\mathcal{B}$ immediately obtains a collision of the hash function H_1^*.

Case 2. $(\mathsf{Tag}' \parallel C_2) = (\mathsf{Tag} \parallel CT_3)$, but $K_1^* \neq K_1'$. Because $H_0^*(K_1^*) = \mathsf{Tag} = \mathsf{Tag}' = H_0^*(K_1')$, $\mathcal{B}$ obtains a collision of the hash function H_0^*.

Therefore, the adversary $\mathcal{B}$ is able to break the security of collision-resistant hash functions. This completes the proof.

Theorem 3. *If an adversary $\mathcal{A}$ can break the weak master secret key security with probability ϵ, then, there exists a PPT adversary $\mathcal{B}$, who attacks on the indistinguishability obfuscation, such that $Adv^{Obf}_{i\mathcal{O},\mathcal{B}}(1^\lambda) = \epsilon$.*

Proof. Since the distinguisher $\mathcal{B}$ of obfuscator $i\mathcal{O}$ consists of two parts: Samp and $\mathcal{D}$, we respectively construct them as follows.

The algorithm $\mathsf{Samp}(\cdot)$ takes 1^λ as input. Then randomly choose a PRF key $K \leftarrow \mathsf{Key}(1^\lambda)$ and a random key K_1, sample two key pairs $(pk_1, sk_1) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$ and $(pk_2, sk_2) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$, compute a CRS $crs \leftarrow \mathsf{NIWI.Setup}(1^\lambda)$ for the NIWI proof system, an extractor $h \in \mathcal{H}$, two hash functions H_0 and H_1, and a symmetric encryption scheme SE, compute a commitment $C \leftarrow \mathsf{Com}(0^{len})$, two ciphertexts $CT_1^* = \mathsf{PKE.Enc}(pk_1, K_1)$ and $CT_2^* = \mathsf{PKE.Enc}(pk_2, K_1)$, a punctured key $K' \leftarrow \mathsf{Puncture}(K, CT_1^* \parallel CT_2^*)$, and a random string $r^* \leftarrow \mathsf{Eval}(K, CT_1^* \parallel CT_2^*)$, and return $mpk = (pk_1, pk_2, h, H_0, H_1, crs, C, \mathsf{SE})$ to $\mathcal{A}$. When $\mathcal{A}$ makes the transformation key generation query, it sets $\mathcal{G}_0 = \mathcal{G}$ which is a circuit in our construction with (mpk, pk_R, sk_1, K) hardwired in it. It computes $C^* = \mathsf{PKE.Enc}(pk_R, K_1; r^*)$, and constructs the circuit $\mathcal{G}_1$ which is described in Fig. 2, and has $(mpk, pk_R, sk_2, K', CT_1^* \parallel CT_2^*, C^*)$ hardwired in it. Finally, it outputs the two challenge circuits $(\mathcal{G}_0, \mathcal{G}_1)$.

Input: Ciphertext CT
Constants: $mpk, sk_2, K', pk_R, CT_1^* \parallel CT_2^*, C^*$

1. Parse $CT = (CT_1, CT_2, CT_3, \pi)$.
2. If $\mathsf{NIWI.Verify}(crs, y, \pi) = 0$, then output $\perp$ and stop, otherwise continue to the next step. Here $y = (CT_1, CT_2, C, pk_1, pk_2)$ is the statement corresponding to π.
3. If $CT_1 \parallel CT_2 = CT_1^* \parallel CT_2^*$, output $CT_{out} = (C_1 = C^*, C_2 = CT_3)$ and stop.
4. Compute $K_1 \leftarrow \mathsf{PKE.Dec}(sk_2, CT_2)$.
5. Compute $r \leftarrow \mathsf{Eval}(K', CT_1 \parallel CT_2)$.
6. Compute $C_1 \leftarrow \mathsf{PKE}_R\mathsf{.Enc}(pk_R, K_1; r)$ by using the user's encryption algorithm, set $C_2 = CT_3$, and output $CT_{out} = (C_1, C_2)$.

Fig. 2. Functionality $\mathcal{G}_1$

The sub-distinguisher $\mathcal{D}$ takes as input $TK_b = i\mathcal{O}(\lambda, \mathcal{G}_b)$, where b is the challenge bit for $\mathcal{D}$. It sends TK_b to the adversary $\mathcal{A}$, and receives a secret key sk. If $sk = sk_1$, output $b = 0$, otherwise output $b = 1$.

Now, we prove that the two circuits $\mathcal{G}_0$ and $\mathcal{G}_1$ are equivalent on functionality. First, for any input $CT = (CT_1, CT_2, CT_3, \pi)$, $\mathcal{G}_0$ outputs $\perp$ if and only if $\mathcal{G}_1$ outputs $\perp$. Note that both $\mathcal{G}_0$ and $\mathcal{G}_1$ output $\perp$ if and only if the proof π does not

verify, i.e., $\mathsf{NIWI.Verify}(crs, y, \pi) = 0$, where $y = (CT_1, CT_2, pk_1, pk_2, \pi)$. Next, we prove that both $\mathcal{G}_0$ and $\mathcal{G}_1$ have the same functionality for all valid inputs. We consider two cases: $CT_1 \parallel CT_2 \neq CT_1^* \parallel CT_2^*$ and $CT_1 \parallel CT_2 = CT_1^* \parallel CT_2^*$. For the first case, by the first property of puncturable PRF, it follows that $\mathsf{Eval}(K, CT_1 \parallel CT_2) = \mathsf{Eval}(K', CT_1 \parallel CT_2) = r$. Since NIWI is statistical soundness, both $\mathcal{G}_0$ and $\mathcal{G}_1$ can get K_1 by decrypting CT_1 and CT_2 respectively, and output $(C_1 = \mathsf{PKE}_R.\mathsf{Enc}(pk_R, K_1; r), C_2 = CT_3)$ at the same time. In the second case, $\mathcal{G}_0$ computes $\mathsf{Eval}(K, CT_1^* \parallel CT_2^*) = r$, then decrypts $K_1 \leftarrow \mathsf{PKE.Dec}(sk_1, CT_1^*)$ by using the secret key sk_1, and outputs $C_1 = \mathsf{PKE}_U\mathsf{Enc}(pk_R, K_1; r)$. On the other hand, because $CT_1^* \parallel CT_2^* = CT_1 \parallel CT_2$, $\mathcal{G}_1$ outputs the hardwired value C^*. Note that $C_1 = C^*$ and $C_2 = CT_3$ dose not be changed, we can get that $\mathcal{G}_0(CT) = \mathcal{G}_1(CT)$.

Therefore, we have $\mathrm{Adv}^{MKS}_{\mathcal{VPRE},\mathcal{A}}(\lambda) = \mathrm{Adv}^{Obf}_{i\mathcal{O},\mathcal{B}}(1^\lambda) = \epsilon$.

Efficiency. Our verifiable proxy re-encryption is efficient. Compared with the transformation algorithm, the verifiable algorithm only introduces a hash value which is the verification key, and two hash value computations in the final decryption algorithm.

5 Conclusion

In this work, we construct a verifiable proxy re-encryption scheme from indistinguishability obfuscation. It can ensure the security of master secret key even when the proxy and the receiver collude. In addition, the key storage of the receiver will remain constant, no matter how many clients he deals with. Furthermore, the size of re-encrypted ciphertexts in our scheme only depends on the message size and security parameter.

Acknowledgment. This work is supported by the "Strategic Priority Research Program" of the Chinese Academy of Sciences, Grants No. XDA06010701, National Natural Science Foundation of China (No. 61402471, 61472414, 61170280), and IIE's Cryptography Research Project.

A Preliminaries (Cont.)

A.1 Non-interactive Witness Indistinguishable Proofs

In this section, we present the definition for non-interactive witness-indistinguishable (NIWI) proofs. We emphasize that we are interested in proof systems, i.e., where the soundness guarantee holds against computationally unbounded cheating provers.

Definition 9. *(NIWI). A non-interactive witness-indistinguishable proof system for a language L with a PPT relation R is a tuple of algorithms* $(\mathsf{NIWI.Setup}, \mathsf{NIWI.Prove}, \mathsf{NIWI.Verify})$ *such that the following properties hold:*

- ***(Perfect Completeness).*** *For every* $(x, w) \in R$*, it holds that*

$$\Pr[\mathsf{NIWI.Verify}(crs, x, \mathsf{NIWI.Prove}(crs, x, w)) = 1] = 1$$

where $crs \leftarrow \mathsf{NIWI.Setup}(1^\lambda)$*, and the probability is taken over the coins of* NIWI.Setup*,* NIWI.Prove *and* NIWI.Verify*.*
- ***(Statistical Soundness).*** *For every adversary* $\mathcal{A}$*, it holds that*

$$\Pr\left[\mathsf{NIWI.Verify}(crs, x, \pi) = 1 \wedge x \notin L \middle| \begin{array}{c} crs \leftarrow \mathsf{NIWI.Setup}(1^\lambda); \\ (x, \pi) \leftarrow \mathcal{A}(crs) \end{array}\right] = \mathrm{negl}(1^\lambda).$$

- ***(Witness Indistinguishability).*** *For any triplet* (x, w_0, w_1) *such that* $(x, w_0) \in R$ *and* $(x, w_1) \in R$*, the distributions* $\{crs, \mathsf{NIWI.Prove}(crs, x, w_0)\}$ *and* $\{crs, \mathsf{NIWI.Prove}(crs, x, w_1)\}$ *are computationally indistinguishable, where* $crs \leftarrow \mathsf{NIWI.Setup}(1^\lambda)$*.*

A.2 Commitment Schemes

A commitment scheme Com is a PPT algorithm that takes as input a string x and a randomness r and outputs $c \leftarrow \mathsf{Com}(x; r)$. A perfectly binding commitment scheme must satisfy the following properties:

- **(Perfectly Binding).** This property states that two different strings cannot have the same commitment. More formally, $\forall x_1 \neq x_2$ and r_1, r_2, $\mathsf{Com}(x_1; r_1) \neq \mathsf{Com}(x_2; r_2)$.
- **(Computational Hiding).** For all strings x_0 and x_1 (of the same length), for all PPT adversaries $\mathcal{A}$, we have that:

$$|\Pr[\mathcal{A}_1(\mathsf{Com}(x_0))] = 1 - \Pr[\mathcal{A}_1(\mathsf{Com}(x_1)) = 1]| \leq \mathrm{negl}(\lambda).$$

References

1. 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), 1–30 (2006)
2. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001)
3. Blaze, M., Bleumer, G., Strauss, M.J.: Divertible protocols and atomic proxy cryptography. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 127–144. Springer, Heidelberg (1998)
4. Canetti, R., Hohenberger, S.: Chosen-ciphertext secure proxy re-encryption. In: Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, pp. 185–194. Alexandria, 28–31 October 2007
5. Desmedt, Y.G., Frankel, Y.: Threshold cryptosystems. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 307–315. Springer, Heidelberg (1990)
6. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractors: how to generate strong keys from biometrics and other noisy data. SIAM J. Comput. **38**(1), 97–139 (2008)

7. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, pp. 40–49. Berkeley, 26–29 October 2013
8. Hanaoka, G., Kawai, Y., Kunihiro, N., Matsuda, T., Weng, J., Zhang, R., Zhao, Y.: Generic construction of chosen ciphertext secure proxy re-encryption. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 349–364. Springer, Heidelberg (2012)
9. Hohenberger, S., Rothblum, G.N., Shelat, A., Vaikuntanathan, V.: Securely obfuscating re-encryption. J. Cryptol. **24**(4), 694–719 (2011)
10. Isshiki, T., Nguyen, M.H., Tanaka, K.: Proxy re-encryption in a stronger security model extended from CT-RSA2012. In: Dawson, E. (ed.) CT-RSA 2013. LNCS, vol. 7779, pp. 277–292. Springer, Heidelberg (2013)
11. Ivan, A., Dodis, Y.: Proxy cryptography revisited. In: Proceedings of the Network and Distributed System Security Symposium, NDSS 2003. San Diego (2003)
12. Kirshanova, E.: Proxy re-encryption from lattices. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 77–94. Springer, Heidelberg (2014)
13. 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)
14. Mambo, M., Okamoto, E.: Proxy cryptosystems: delegation of the power to decrypt ciphertexts (special section on cryptography and information security). IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **80**(1), 54–63 (1997)
15. Matsuda, T., Nishimaki, R., Tanaka, K.: CCA proxy re-encryption without bilinear maps in the standard model. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 261–278. Springer, Heidelberg (2010)
16. Ohata, S., Kawai, Y., Matsuda, T., Hanaoka, G., Matsuura, K.: Re-encryption verifiability: how to detect malicious activities of a proxy in proxy re-encryption. In: Nyberg, K. (ed.) CT-RSA 2015. LNCS, vol. 9048, pp. 410–428. Springer, Heidelberg (2015)
17. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: Symposium on Theory of Computing, STOC 2014, pp. 475–484. New York, May 31–June 03 2014
18. Shao, J., Cao, Z.: CCA-secure proxy re-encryption without pairings. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 357–376. Springer, Heidelberg (2009)
19. Weng, J., Zhao, Y., Hanaoka, G.: On the security of a bidirectional proxy re-encryption scheme from PKC 2010. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 284–295. Springer, Heidelberg (2011)