# Cloud-based Data-Sharing Scheme using Verifiable and CCA-Secure Re-encryption from Indistinguishability Obfuscation*

Mingwu Zhang[1,2], Yan Jiang[3], Hua Shen[1], and Willy Susilo[4]

[1] School of Computers, Hubei University of Technology
[2] Hubei Key Laboratory of Intelligent Geo-Information Processing, China University of Geosciences
[3] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics
[4] Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Australia
csmwzhang@gmail.com

**Abstract.** A cloud-based re-encryption scheme allows a semi-trusted cloud proxy to convert a ciphertext under delegator's public-key into a ciphertext of delegatee's. However, for an untrusted cloud proxy, as the *re-encryption program was outsourced on the cloud, the cloud can debug the program and might have illegal activities in practice, such as monitoring the program executing, returning an incorrect re-encryption ciphertext, or colluding with the participants to obtain the sensitive information.* In this work, we propose a construction of cloud-based verifiable re-encryption by incorporating new cryptographic primitives of *indistinguishability obfuscation* and *puncturable pseudorandom functions*, which can achieve the *master-secret security* even if the proxy colludes with the delegatee. Furthermore, our scheme can provide the white-box security in re-encryption procedure to *implement the sensitive-data protection in the presence of white-box access*, and it resists on chosen-ciphertext attacks in both the first-level encryption and the second-level encryption. The decryption is very efficient since it only requires several symmetric PRF operations, which can be deployed and applied in the light-weight security device such as Mobile Phones (MPs), Wireless Body Area Networks (WBANs) and nodes in Internet-of-Things (IoTs).

**Keywords:** · Data sharing · E-mail forwarding · White-box access· Re-encryption· Indistinguishability obfuscation· Puncturable PRF.

* This work is supported by the National Natural Science Foundation of China ( 61672010, 61702168), the open research project of The Hubei Key Laboratory of Intelligent Geo-Information Processing (KLIGIP-2017A11), and the fund of Hubei Key Laboratory of Transportation Internet of Things (WHUTIOT-2017B001).

## 1 Introduction

Proxy re-encryption (PRE), initially introduced by Blaze et al. [4], allows a semi-trusted proxy to convert a ciphertext under the delegator's public-key into a ciphertext of the delegatee, without observing the underlying plaintext and the secret key of either delegator or delegatee. In traditional PRE schemes, a proxy is modeled as a *semi-trusted server*, who should execute the functionality of re-encryption honestly. However, it may not be suitable for some applications, e.g., electronic mail forwarding, and secure cloud-based data sharing.

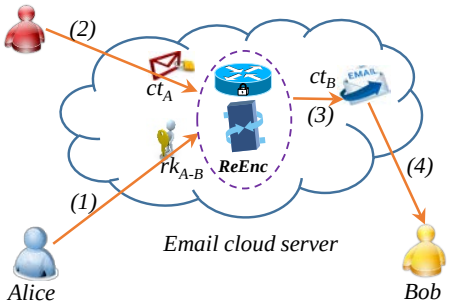


**Fig. 1.** Scenario of encrypted email forwarding

Consider *encrypted email-forwarding* as an example which is depicted in Fig. 1. An encrypted email is sent to user Alice but unfortunately she is on vacation and hence, cannot read the email. She can set an *automatic forwarding functionality* in the cloud email server. If Alice forwards an encrypted email to Bob directly, then Bob could not read the encrypted email correctly since the email was indeed encrypted by Alice's public-key. In order to allow Bob to decrypt and read the encrypted email, a salutary approach is to specify and create a re-key $rk_{A\to B}$ by Alice and sends to the mail server. When Bob requests the email from the mail server, the server can convert the encrypted mail of Alice into a re-encrypted email of Bob using $rk_{A\to B}$. However, as the cloud-based mail server might be *conscious and malicious*, and it may return a fake email, or produce a re-encryption email but not what user Bob needed. Hence, it is mandatory to check and verify the correctness of a re-encrypted email ciphertext.

We consider an encrypted data-sharing scenario in cloud environments, in which the cloud server can perform the data-sharing procedure and also attempts to obtain more sensitive information. Additionally, we allow the cloud to execute the data-sharing program in a white-box access model such that it can debug the program, set the breakpoints and monitor the memories or variables during the executing.

Our contribution and techniques. In this work, we explore a verifiable cloud-based proxy re-encryption that employs the primitives of indistinguisha-

bility obfuscation $i\mathcal{O}$ and puncturable pseudorandom functions. Our scheme can be used in the secure and sensitive data-sharing in cloud system while keeping the confidentiality of the sensitive information. Besides the security of general proxy re-encryption, our contribution is benefit as follows:

- The scheme reinforces the *master secret-key security* so that the adversary (i.e., a dishonest cloud server) cannot obtain the master secret-key of the sender even if the cloud proxy colludes with the delegatee.
- The proposed scheme obtains the *white-box access security for the cloud proxy*. As the re-encryption program was executed in the cloud, to obtain more sensitive data in the program, the cloud can debug the re-encryption program step-by-step, trace into the program, and monitor the memory and register. We employ the *program obfuscation technique* to prevent the cloud from watching and stealing the sensitive data embedded in the program.
- In order to avoid the cloud server providing a fake re-encryption transformation, we present a functionality of *verifiability of the re-encryption* to ensure the consistency and correctness of the re-encryption procedure while keeping the underlying sensitive information security (i.e., secret keys and plaintexts).
- Our scheme gives a *strong CCA security* (i.e., resist on active attacks on ciphertexts) to guarantee *for two levels of encryptions*. That is, the (original) second-level encryption and the (transformed) first-level encryption are all secure against adaptively chosen-ciphertext attacks.

Actually, program obfuscation can make an (outsourced) computer program unintelligible while preserving its functionality, which provides an effective mechanism to securely and perfectly hide the sensitive data in outsourced program even the program executer has access the program in a white-box manner, such as debugging the program, tracing into the variables or setting the breakpoints.

We are now ready to describe our construction that employs the indistinguishability obfuscation. The setup algorithm at firsts picks up the puncturable keys $(k_1, k_2)$. Next, it creates the public key as an obfuscation version of a program to perfectly hide the keys $(k_1, k_2)$. Then the encryption algorithm can call this obfuscated program to encrypt a cleartext as follows: Compute $u = F_1(k_1, (m, r))$ where $r$ was a randomness, run the obfuscated program on input $r$ and $u$, and output the ciphertext $ct$ as $(\alpha = H(m, r, u), \beta = F_2(k_2, \alpha) \oplus (m, r))$, where $F_1$ and $F_2$ are puncturable pseudorandom functions.

To avoid the cloud obtaining the clear re-encryption key $rk_{r\to j}$, the re-encryption key $rk_{i\to j}$ will be created as an obfuscated program, which will take as input a second-level ciphertext $ct_i = (\alpha, \beta)$ and outputs a transformed first-level ciphertext $ct_j$. The program first computes $(m, r) = \beta \oplus F_2(k_2, \alpha)$, $u = F_1(k_1, (m, r))$ and then checks whether $\alpha = H(m, r, u)$.

To invalidate the re-encryption key query from the adversary, we hardwire the punctured PRF key to generate a randomness $r'$ for the re-encryption ciphertext and then puncture the key using the challenge ciphertext. Due to the security of puncturable PRF, we can invalidate the re-encryption key from the challenge user to another. In order to verify the original ciphertext and the re-encryption

ciphertext that holds the same message $m$, we compute $u' = F_1(k_1, (m, r'))$ and run the obfuscated encryption circuit on $r'$ and $u'$, and output the re-encryption ciphertext $\widehat{ct}$ as $(\alpha' = H(m, r', u'), \beta' = F_j(k_{j,2}, \alpha'), u')$.

The security proof of our scheme is proceeded by a sequence of indistinguishable games. At first, we use puncturable PRF keys $(k_1, k_2)$ to create the obfuscated program $\mathcal{P}^{\mathsf{Enc}}$ and $\mathcal{Q}^{\mathsf{REnc}}$. Next, we employ the punctured programming techniques to replace those normal evaluations of programs with hardwired and randomly sampled value. In the final game, any $p.p.t$ adversary $\mathcal{A}$ has negligible advantage in guessing the underlying cleartext.

Related works. Blaze et al. [4] proposed the first bi-directional PRE scheme based on ElGamal PKE scheme. Subsequently, Ateniese et al. [2], Canetti and Hohenberger [7], Libert and Vergnud [24], and Chow et al. [9] proposed different PRE schemes with various properties. Avoid employing the pairings, Shao and Cao proposed a PRE scheme without pairing. However, Zhang et al. [8] pointed out that it is not secure in the Libert and Vergnaud security model.

Hohenberger et al. [19] introduced a mechanism in how to securely obfuscate the re-encryption functionality. Hanaoka et al. [18] and Isshiki et al. [21] presented the construction of chosen-ciphertext secure uni-directional PRE scheme, respectively. Kirshanova [22] proposed a lattice-based PRE scheme. However, none of those schemes consider the verifiability of re-encryption procedure.

Verifiable PRE proposed by Ohata et al.[26] is constructed by employing re-encryption verification. Using this approach, the delegator splits his secret key into $tsk_1$ and $tsk_2$ by a re-splittable threshold public key encryption[18]. Next, it computes $\psi = \mathsf{Enc}(pk_j, tsk_1)$, and sends $\psi$ to user $j$. Then, the proxy re-encrypts the original ciphertext $c_i$ by using the re-encryption key $rk_{i\to j}$(imply $tsk_2$). That is, it computes $u_2 = \mathsf{Dec}(tsk_2, c_i)$ and sets the re-encrypted ciphertext as $\widehat{c}_j = \mathsf{Enc}(\widehat{pk}_j, u_2||c_i)$. The delegatee computes $u_2||c_i = \mathsf{Dec}(\widehat{dk}_j, \widehat{c}_j)$ and $u_1 = \mathsf{Dec}(dk_j, \psi)$, and outputs $m \leftarrow TCom(c, u_1, u_2)$. To achieve the re-encryption verifiability, by augmenting the dedicated re-encryption algorithm. On input $(pk_i, sk_j, c, \widehat{c})$, it executes the first-level decryption algorithm to recover the "*embedded*" second-level ciphetext $c'$, and checks whether $c = c'$.

Liu et al.[25] indicated that the Ohata et al.'s scheme can not resist against collusion attack and they also proposed a new scheme to achieve CPA security based on $i\mathcal{O}$. More precisely, their approach is built on the key encapsulated mechanism. That is, an encryption of a message $m$ using symmetric key $k_{SE}$ and an encryption of $k_{SE}$ using the public key $pk_1$, where $k_{SE}$ is the output of a key extractor. Zhang et al.[28] presented a flexible and controllable obfuscated multi-hop re-encryption in (somewhat inefficient) multilinear groups.

Paper organization. The rest of this paper is organized as follows. Section 2 reviews some preliminaries including mathematical notations, indistinguishability obfuscation and puncturable pseudo random functions. In Section 3, we propose the model and security definition for verifiable PRE. In Section 4, we present our concrete construction and give the security analysis. We provide the practical deployment of secure data-sharing in Section 5 and draw the conclusion in Section 6.

## 2 Preliminaries

Throughout of this paper, we use $\lambda$ to denote the security parameter, and let *p.p.t* denote a probabilistic polynomial-time algorithm (Turing Machine). For an integer $n$, we write $[n]$ to denote the set $\{1, 2, \cdots, n\}$.

A negligible function $\mu(n)$ is a function that for all positive polynomial $p$ there exists a positive integer $\mathbf{N}$ s.t. for all $n > \mathbf{N}$, $\mu(n) < 1/p(n)$. Let $\mathcal{A}$ be an algorithm, and $x$ be the input of $\mathcal{A}$, the evaluation of the Turing machine running the algorithm $\mathcal{A}$ on the input tape with the encodings of $x$ is denoted by $y \leftarrow \mathcal{A}(x)$ where the result $y$ is the output of $\mathcal{A}$. An algorithm $\mathcal{A}$ is said to have oracle access to machine $\mathcal{O}$ if $\mathcal{A}$ can write an input for $\mathcal{O}$ on a special tape, and tell the oracle to execute on that input and then write its output to the tape, which is denoted by $\mathcal{A}^{\mathcal{O}}$.

For any polynomial-size distinguisher $\mathcal{D}$, the advantage $\delta = |\Pr[\mathcal{D}(\mathbf{Expt}(1^\lambda, 0)) = 1] - \Pr[\mathcal{D}(\mathbf{Expt}(1^\lambda, 1))]|$ is bounded by $\delta$, we write as $\mathbf{Expt}(1^\lambda, 0) \approx_\delta \mathbf{Expt}(1^\lambda, 1)$. If $\delta$ is negligible in parameter $\lambda$, we call two distributions (experiments) indistinguishable.

We now recall the notion of indistinguishability obfuscation ($i\mathcal{O}$) and puncturable pseudorandom function and their security requirements.

**Definition 1.** *(**Indistinguishability Obfuscation** ($i\mathcal{O}$)). A uniform p.p.t algorithm $i\mathcal{O}$ is called an indistinguishability obfuscator for a circuit class $\{\mathcal{C}_\lambda\}_{\lambda \in \boldsymbol{N}}$ if the following conditions are satisfied:*

- ***Correctness**: For all security parameters $\lambda \in \boldsymbol{N}$, for all $C \in \{\mathcal{C}_\lambda\}$ and all inputs $x \in \{0,1\}^{\mathsf{poly}(\lambda)}$, it holds that,*

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

- ***Indistinguishability**: For any p.p.t algorithm* Samp *and distinguisher $\mathcal{D}$, there exists a negligible function $\mu(\cdot)$ such that the following holds, i.e., if*

$$\Pr[\forall x, C_0(x) = C_1(x) : (C_0, C_1, aux) \leftarrow \mathrm{Samp}(1^\lambda)] > 1 - \mu(\cdot) \tag{2}$$

*we have,*

$$\Big| \Pr\left[\mathcal{D}(aux, i\mathcal{O}(\lambda, C_0)) = 1 : \ (C_0, C_1, aux) \leftarrow \mathrm{Samp}(1^\lambda)\right] - \Pr\left[\mathcal{D}(aux, i\mathcal{O}(\lambda, C_1)) = 1 : \ (C_0, C_1, aux) \leftarrow \mathrm{Samp}(1^\lambda)\right] \Big| \leq \mu(\lambda) \tag{3}$$

*where aux is the auxiliary information output by the algorithms in the system.*

**Definition 2.** *(**Puncturable Pseudorandom Function**) A puncturable family of* PRFs F *consisting of three Turing Machines* $(\mathrm{Key}_F, \mathrm{Punc}_F, \mathrm{Eval}_F)$*, and a pairs of computing functions $n(\cdot)$ and $m(\cdot)$, satisfies the following properties:*

– ***Functionality preserved under puncturing point****: For every p.p.t algorithm $\mathcal{A}$ that takes as input $1^\lambda$ and outputs a set $S \subseteq \{0,1\}^{n(\lambda)}$, for all $x \notin S$, we have,*

$$\Pr\left[\mathrm{Eval}(k,x) = \mathrm{Eval}(k_S,x) :\ k \leftarrow \mathrm{Key}_F(1^\lambda), k_S = \mathrm{Punc}_F(k,S)\right] = 1 \quad (4)$$

– ***Pseudorandom at punctured points****: For every p.p.t adversary $(\mathcal{A}_1,\mathcal{A}_2)$ such that $\mathcal{A}_1$ takes as input $1^\lambda$ and outputs a set $S \subseteq \{0,1\}^{n(\lambda)}$. For all $k \leftarrow \mathrm{Key}_F(1^\lambda)$, $k_S = \mathrm{Punc}_F(k,S)$, and $x \in S$, we have,*

$$\left|\Pr\left[\mathcal{A}_2(k_S,x,\mathrm{Eval}(k,x))\right] = 1\ - \Pr\left[\mathcal{A}_2(k_S,x,U_{m(\lambda)}\right] = 1\right| \leq \mu(\lambda) \quad (5)$$

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

*Remark 1.* In order to simplify notation, in this paper, we write $F(k,S)$ to stand for $Eval_F(k,x)$ and $k(S)$ for $Punc_F(k,S)$, respectively.

Puncturable PRFs can easily be constructed from GGMs PRFs[12] which are based on one-way functions. The following lemma states that the statistical injective PPRF can be constructed.

**Lemma 1.** *[27] If one-way functions exist, then for all eciently computable functions $\ell(\lambda)$, $m(\lambda)$ and $e(\lambda)$ such that $\ell(\lambda) \geq 2m(\lambda) + e(\lambda)$, there exists a statistically injective puncturable PRF family with failure probability $1/2^{e(\lambda)}$ that maps $\ell(\lambda)$ bits to $m(\lambda)$ bits.*

## 3 Models and Definitions

### 3.1 Algorithms and definitions of VPRE

Our syntax for verifiable proxy re-encryption (VPRE) roughly follows in the line of [26] except that we generate an additional verifiable key in the encryption process for the verifiability of re-encryption ciphertext. A single-hop unidirectional VPRE comprises of the following six algorithms whose flowchart is described in Fig. 2:

– $\mathsf{KGen}(1^\lambda) \rightarrow (pk, sk)$: Key generation algorithm is a polynomial algorithm that takes the security parameter $\lambda$, and outputs public keys $pk$ and secret keys $sk$.
– $\mathsf{Enc}(pk_i, m) \rightarrow (ct, vk)$: Original encryption algorithm takes the public key $pk_i$, a message $m$ and a randomness $r$ and outputs an original second-level ciphertext $ct_i$, and a verifiable key $vk$.
– $\mathsf{RKey}(sk_i, pk_j) \rightarrow rk_{i\rightarrow j}$: Re-key generation algorithm takes the secret key of user $i$, i.e., $sk_i$, and the public key $pk_j$ of user $j$, and outputs a re-key $rk_{i\rightarrow j}$.
– $\mathsf{REnc}(rk_{i\rightarrow j}, ct_i) \rightarrow \widehat{ct_j}$: Re-encryption algorithm takes the re-key $rk_{i\rightarrow j}$ and a second-level ciphertext $ct_i$, and outputs a first-level ciphertext $\widehat{ct_j}$.

- $\mathsf{Dec}_2(sk_i, ct_i) \rightarrow m|\perp$: Second-level decryption algorithm takes a secret key $sk_i$, and an original second-level ciphertext $ct_i$, and outputs a message $m$ or the special symbol $\perp$.
- $\mathsf{Dec}_1(sk_j, \widehat{ct_j}) \rightarrow m|\perp$: First-level decryption algorithm takes a secret key $sk_j$ and a first-level ciphertext $\widehat{ct_j}$, and outputs a message $m$ or $\perp$, which indicates that the $ct_i$ is invalid.

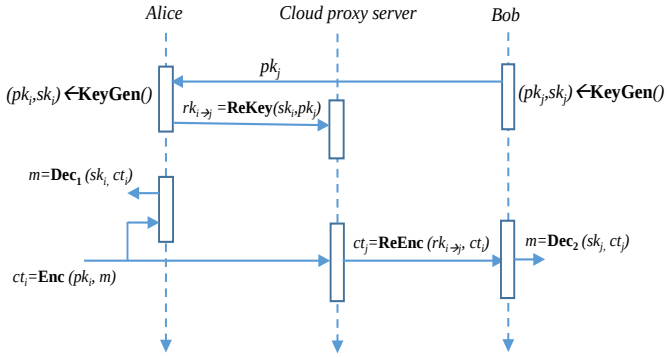


**Fig. 2.** Workflow of cloud-based verifiable proxy re-encryption

Let $\mathcal{M}$ be the message space. A VPRE scheme is consistent and correct if for all messages $m \in \mathcal{M}$ and any key pairs $(pk_i, sk_i), (pk_j, sk_j) \leftarrow \mathsf{KGen}(1^\lambda)$, the following conditions hold:

1. The second-level decryption correctness:

$$\Pr\left[\mathsf{Dec}_2(sk_i, \mathsf{Enc}(pk_i, m)) \neq m\right] \leq \mu(\lambda) \tag{6}$$

2. The first-level decryption consistency:

$$\Pr\left[\mathsf{Dec}_1\big(sk_j,\ \mathsf{REnc}(\mathsf{RKey}(sk_i, pk_j),\ \mathsf{Enc}(pk_i, m))\big) \neq m\right] \leq \mu(\lambda)$$

### 3.2 CCA security

We adopt the chosen-ciphertext attack (CCA) security of VPRE scheme that is defined as follows.

**Definition 3.** *(**CCA Security for Second-level Ciphertext**). A uni-directional VPRE scheme is said to be CCA secure at second level if the probability is negligibly close to* $1/2$ *for any p.p.t adversary* $\mathcal{A}$ *which is shown as follows*

$$\Pr\left[\begin{array}{l} (pk^*, sk^*) \leftarrow \mathsf{KGen}(1^\lambda),\ \{(pk_c, sk_c) \leftarrow \mathsf{KGen}(1^\lambda)\}, \\ \{(pk_h, sk_h) \leftarrow \mathsf{KGen}(1^\lambda))\},\ \{rk_{c\to *} \leftarrow \mathsf{RKey}(sk_c, pk^*)\}, \\ \{rk_{*\to h} \leftarrow \mathsf{RKey}(sk^*, pk_h)\},\ \{rk_{h\to *} \leftarrow \mathsf{RKey}(sk_h, pk^*)\}, \\ \{rk_{h\to c} \leftarrow \mathsf{RKey}(sk_h, pk_c)\},\ \{rk_{c\to h} \leftarrow \mathsf{RKey}(sk_c, pk_h)\}, \\ \{rk_{h\to h'} \leftarrow \mathsf{RKey}(sk_h, pk_{h'})\},\ \{rk_{c\to c'} \leftarrow \mathsf{RKey}(sk_c, pk_{c'})\}, \\ b' = b : (m_0, m_1, aux) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Dec}_1}, \mathcal{O}_{\mathsf{Dec}_2}, \mathcal{O}_{\mathsf{REnc}}}\Big(pk^*, \{pk_c, sk_c\}, \\ \qquad\qquad \{pk_h\}, \{rk_{c\to *}\}, \{rk_{h\to *}\}, \{rk_{*\to h}\}, \\ \qquad\qquad \{rk_{c\to h}\}, \{rk_{h\to c}\}, \{rk_{h\to h'}\}, \{rk_{c\to c'}\}\Big), \\ b \xleftarrow{R} \{0,1\},\quad ct^* = \mathsf{Enc}(pk^*, m_b), \\ b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Dec}_1}, \mathcal{O}_{\mathsf{Dec}_2}, \mathcal{O}_{\mathsf{REnc}}}(ct^*, aux) \end{array}\right] \leq \mu(\lambda)$$

In this case, $aux$ is a state information held by $\mathcal{A}$ and $(pk^*, sk^*)$ is the challenge user's key pair generated by the challenger. For honest users, keys are indicated by $h$ or $h'$ and we indicate corrupt users by $c$ or $c'$. The adversary is given all re-encryption keys except for those that could re-encrypt the ciphertext from the challenge one to the corrupt one. In the security experiment, $\mathcal{A}$ is said to have advantage $\varepsilon$ if this probability is at least $1/2 + \varepsilon$.

In the above CCA security experiments, Oracles $\mathcal{O}_{\mathsf{Dec}_1}, \mathcal{O}_{\mathsf{Dec}_2}, \mathcal{O}_{\mathsf{REnc}}$ work as follows:

- **Re-encryption Oracle** $\mathcal{O}_{\mathsf{REnc}}$: for a re-encryption query $(pk_i, pk_j, ct_i)$, the oracle responds follows: If $(pk_i, ct_i) = (pk^*, ct_i^*)$ and $pk_j \notin pk_h$, then the oracle returns the special symbol $\perp$ to $\mathcal{A}$. Otherwise, the oracle answers with $\mathsf{REnc}(\mathsf{RKey}(sk_i, pk_j), ct_i))$.
- **First-level Decryption Oracle** $\mathcal{O}_{\mathsf{Dec}_1}$: For a first-level decryption query $(pk_i, \widehat{ct_i})$, the oracle responds as follows: If $\mathcal{A}$ has required a re-encryption query $(pk^*, pk_i, ct_i^*)$ and obtained $\widehat{ct_i}$ before, then the oracle searches the tuple in the record table and returns the tuple to $\mathcal{A}$. If the adversary $\mathcal{A}$ has requested a re-encryption key query $(pk^*, pk_i)$ previously and $\mathsf{Dec}_1(sk_i, \widehat{ct_i}) \in \{m_0, m_1\}$, then the oracle ansers with "*test*" to $\mathcal{A}$. Otherwise, the oracle answers with $\mathsf{Dec}_1(sk_i, \widehat{ct_i})$.
- **Second-level Decryption Oracle** $\mathcal{O}_{\mathsf{Dec}_2}$: For a second-level decryption query $(pk_i, ct_i)$, the oracle responds with $\mathsf{Dec}_2(sk_i, ct_i)$, except for the challenge ciphertext. i.e., if $(pk_i, ct_i) = (pk^*, ct_i^*)$, then the oracle answers with a symbol $\perp$.

In the security of first-level ciphertexts for uni-directional $\mathsf{VPRE}$ schemes, $\mathcal{A}$ is allowed to have access to all the re-encryption keys in the definition. Since all first-level ciphertexts cannot be re-encrypted, there is indeed no reason to keep adversary from obtaining all honest to corrupt re-encryption keys. The re-encryption oracle becomes futile because of all the re-encryption keys are available to adversary $\mathcal{A}$.

**Definition 4.** *(**CCA Security for First-level Ciphertext**). A single-hop unidirectional VPRE scheme is said to be CCA secure at first-level if the probability is negligible for any p.p.t adversary $\mathcal{A}$, where the challenge user's key pair $(sk^*, pk^*)$ and the challenge ciphertext $\widehat{ct^*}$ are generated by the challenger, which is shown as follows*

$$\Pr\left[\begin{array}{l} (pk^*, sk^*) \leftarrow \mathsf{KGen}(1^\lambda), \quad \{(pk_c, sk_c) \leftarrow \mathsf{KGen}(1^\lambda)\}, \\ \{(pk_h, sk_h) \leftarrow \mathsf{KGen}(1^\lambda))\}, \quad \{rk_{c\to *} \leftarrow \mathsf{RKey}(sk_c, pk^*)\}, \\ \{rk_{*\to h} \leftarrow \mathsf{RKey}(sk^*, pk_h)\}, \quad \{rk_{h\to *} \leftarrow \mathsf{RKey}(sk_h, pk^*)\}, \\ \{rk_{h\to c} \leftarrow \mathsf{RKey}(sk_h, pk_c)\}, \quad \{rk_{c\to h} \leftarrow \mathsf{RKey}(sk_c, pk_h)\}, \\ \{rk_{h\to h'} \leftarrow \mathsf{RKey}(sk_h, pk_{h'})\}, \quad \{rk_{c\to c'} \leftarrow \mathsf{RKey}(sk_c, pk_{c'})\}, \\ b' = b : (m_0, m_1, sk_{\mathcal{A}}, pk_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Dec}_1}, \mathcal{O}_{\mathsf{Dec}_2}}\Big(pk^*, \{pk_c, sk_c\}, \{pk_h, sk_h\}, \\ \quad \{rk_{c\to *}\}, \{rk_{h\to *}\}, \{rk_{*\to h}\}, \{rk_{c\to h}\}, \{rk_{h\to c}\}, \{rk_{h\to h'}\}, \{rk_{c\to c'}\}\Big), \\ b \xleftarrow{R} \{0,1\}, \\ ct = \mathsf{Enc}(pk_{\mathcal{A}}, m_b), \quad \widehat{ct^*} = \mathsf{REnc}(\mathsf{RKey}(sk_{\mathcal{A}}, pk^*), ct), \\ b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Dec}_1}, \mathcal{O}_{\mathsf{Dec}_2}}(\widehat{ct^*}) \end{array}\right] \leq \mu(\lambda)$$

Ohata et al.[26] introduces a new functionality for proxy re-encryption with verifiability of re-encryption procedure. Ateniese et al.[2] defines the property for unidirectional *PRE* schemes, i.e., master secret key security. We give the security requirements of verifiable PRE with master-key security as follows:

**Definition 5.** *(**CCA-VPRE Security**). A VPRE scheme is said to be CCA-secure verifiable VPRE against master-key exposure if both the first-level encryption and the-second level encryption are CCA secure.*

1. ***Master secret security**. Master secret security captures the inability to obtain the master secret key even if the cloud proxy and the delegatee collude. More formally, the following probability should be negligible in security parameter $\lambda$,*

$$\Pr\left[\begin{array}{l} (pk^*, sk^*) \leftarrow \mathsf{KGen}(1^\lambda), \\ \{(pk_c, sk_c) \leftarrow \mathsf{KGen}(1^\lambda)\}, \\ \chi = sk^* : \{rk_{c\to *} \leftarrow \mathsf{RKey}(sk_c, pk^*)\}, \\ \{rk_{*\to c} \leftarrow \mathsf{RKey}(sk^*, pk_c)\}, \\ \chi \leftarrow \mathcal{A}\big(\{pk_c, sk_c\}, \{rk_{c\to *}\}, \{rk_{*\to c}\}\big) \end{array}\right] \leq \mu(\lambda)$$

2. ***Re-encryption verifiability**. Re-encryption verifiability ensues that,*
   (a) *If the adversary who obtains a re-encryption key $rk_{i\to j}$ and is given an original (second-level) ciphertext $ct_i$, it can produce only a re-encrypted ciphertext $\widehat{ct_j}$ that can decrypt the same message as the decryption result of $ct_i$.*
   (b) *If the adversary does not have the re-encryption key $rk_{i\to j}$, then it cannot create a valid re-encryption ciphertext $\widehat{ct_j}$ at all.*

*Concretely, for any p.p.t adversary $\mathcal{A}$, the following probability is negligible.*

$$\Pr\left[\begin{array}{l} m' \neq m^* \wedge m' \neq \perp : \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array}\right.\left.\begin{array}{l} (pk^*, sk^*) \leftarrow \mathsf{KGen}(1^\lambda), \quad \{(pk_c, sk_c) \leftarrow \mathsf{KGen}(1^\lambda)\}, \\ \{(pk_h, sk_h) \leftarrow \mathsf{KGen}(1^\lambda))\}, \quad \{rk_{c\to *} \leftarrow \mathsf{RKey}(sk_c, pk^*)\}, \\ \{rk_{*\to h} \leftarrow \mathsf{RKey}(sk^*, pk_h)\}, \quad \{rk_{h\to *} \leftarrow \mathsf{RKey}(sk_h, pk^*)\}, \\ \{rk_{h\to c} \leftarrow \mathsf{RKey}(sk_h, pk_c)\}, \quad \{rk_{c\to h} \leftarrow \mathsf{RKey}(sk_c, pk_h)\}, \\ \{rk_{h\to h'} \leftarrow \mathsf{RKey}(sk_h, pk_{h'})\}, \quad \{rk_{c\to c'} \leftarrow \mathsf{RKey}(sk_c, pk_{c'})\}, \\ m \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Dec}_1}, \mathcal{O}_{\mathsf{REnc}}}\Big(pk^*, \{pk_c, sk_c\}, \ \{pk_h\}, \{rk_{c\to *}\}, \{rk_{h\to *}\}, \\ \qquad\qquad \{rk_{*\to h}\}, \{rk_{c\to h}\}, \{rk_{h\to c}\}, \quad \{rk_{h\to h'}\}, \{rk_{c\to c'}\}\Big), \\ ct^* = \mathsf{Enc}(pk^*, m^*), \\ m' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Dec}_1}, \mathcal{O}_{\mathsf{Dec}_2}}(\widehat{ct^*}, pk_{R_j}) \end{array}\right] \leq \mu(\lambda)$$

## 4 Proposed Construction

### 4.1 Main idea

We now describe the main idea of our scheme. The functionality of re-encryption is easily realized that if it is allowed to decrypt the ciphertext and to re-encrypt the underlying cleartext by the cloud proxy. However, we should guarantee that the cloud proxy cannot gain any sensitive data during performing the transformation. That is,"*decrypt-then-encrypt*" procedure guarantees that be not able to expose the secret key of the delegator, or the embedded cleartext.

Let the length of a message be $\ell$. Let $F_1$ be a puncturable PRF that takes input of $(\ell + \lambda)$-bit and outputs $\ell_\delta$-bit, and $F_2$ be a puncturable PRF that takes input of $\ell_\alpha$-bit and outputs $(\ell + \lambda)$-bit. Let $H$ be a collision-resistant cryptographic hash function that takes input of $(\ell + \lambda + \ell_\delta)$-bit and outputs the size of $\ell_\alpha$-bit.

In our scheme, to protect the secret key of delegator from exposing from the re-encryption key, we set the re-encryption key as the obfuscated program by the use of indistinguishability obfuscation $i\mathcal{O}$. To achieve the re-encryption verifiability, we design the $\mathsf{VPRE}$ scheme by employing Sahai and Waters' short signature scheme[27]. Before executing the obfuscated circuit $\mathcal{P}^{\mathsf{Enc}}$, it at first evaluates the signature $u = F_1(k_1, (m, r))$ on message $m$ and randomness $r$.

In the re-encryption circuit, we need to re-randomize the signature and randomness for user $j$. To complete the security proof, we add the puncturable PRF key $k_3$ in the re-encryption circuit and generate the updated randomness for re-encryption by using $k_3$.

### 4.2 Our construction

The concrete construction of $\mathsf{VPRE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{RKey}, \mathsf{REnc}, \mathsf{Dec}_1, \mathsf{Dec}_2)$ is described as follows:

– $\mathsf{KGen}(1^\lambda)$: The key generation algorithm at first chooses a puncturable key $k_1 \leftarrow Key_{F_1}(1^\lambda)$ and $k_2 \leftarrow Key_{F_2}(1^\lambda)$. Next, it creates an obfuscation of program `Encrypt-Circuit` as

$$\mathcal{P}^{\mathsf{Enc}} \leftarrow i\mathcal{O}(1^\lambda, \texttt{Encrypt-Circuit} : [k_1]) \tag{7}$$

The circuit `Encrypt-Circuit` is formally defined in Fig. 3. This obfuscated program, $\mathcal{P}^{\mathsf{Enc}}$, servers as the public key, $pk = \mathcal{P}^{\mathsf{Enc}}$, and the corresponding secret key is $sk = (k_1, k_2)$.

**Encrypt-Circuit**

CONSTANT: puncturable PRF keys $k_1$.
INPUT: message $m \in \{0,1\}^\ell$,
randomness $r \in \{0,1\}^\lambda$,
signature $u \in \{0,1\}^{\ell_\delta}$.
PROCEDURE:
1. Compute $\alpha = H(u, m, r)$.
2. Compute $\beta = F_2(k_2, \alpha) \oplus (m, r)$.
3. Output $ct = (\alpha, \beta)$.

**Fig. 3.** Program of Encrypt-Circuit

– $\mathsf{Enc}(pk = \mathcal{P}^{\mathsf{Enc}}, m \in \{0,1\}^\ell, r \in \{0,1\}^\lambda)$: The encryption algorithm at first computes $u = F_1(k_1, (m, r))$. Next it produces an obfuscated program $\Gamma^{vk}$, which is defined in Fig. 4. It at random chooses $r \in \{0,1\}^\lambda$, and then runs the obfuscated program $\mathcal{P}^{\mathsf{Enc}}$ on inputs $r$, $m$ and $u$ to obtain: $(\alpha, \beta) \leftarrow \mathcal{P}^{\mathsf{Enc}}(m, r, u)$. The output of second-level ciphertext is $ct = (\alpha, \beta)$.

**Verify-key**

CONSTANTS: puncturable PRF key $k_1$.
INPUT: message $m \in \{0,1\}^\ell$,
randomness $r \in \{0,1\}^\lambda$,
signature $u \in \{0,1\}^{\ell_\delta}$.
PROCEDURE:
Check $f(u) = f(F_1(k_1, (m, r)))$. Output "*accept*" if true, "*reject*" otherwise.

**Fig. 4.** Program of Verify-key

– $\mathsf{RKey}(sk_i = (k_{i,1}, k_{i,2}), pk_j = \mathcal{P}_j^{\mathsf{Enc}})$: Let $sk_i = (k_{i,1}, k_{i,2})$ be the delegator's secret key, and $pk_j = \mathcal{P}_j^{\mathsf{Enc}}$ be the public key of delegatee $j$. The re-key generation algorithm at random chooses a puncturable PRF key $k_3 \leftarrow Key_{F_3}(1^\lambda)$, and then produces an obfuscated program $\mathcal{Q}^{\mathsf{REnc}}$ by obfuscating

$$\mathcal{Q}^{\mathsf{REnc}} \leftarrow i\mathcal{O}\big(1^\lambda,\ \texttt{ReEnc-Circuit} : [k_1, k_2, k_3, i\mathcal{O}(\mathcal{P}_j^{\mathsf{Enc}})]\big) \tag{8}$$

which is described in Fig. 5. The re-encryption key is set as $rk_{i\to j} = \mathcal{Q}^{\mathsf{REnc}}$.

**ReEnc-Circuit**

CONSTANTS: puncturable PRF keys $k_{i,1}$, $k_{i,2}$: second-level secret key of user $i$,
$k_3$: puncturable PRF key,
$i\mathcal{O}(\mathcal{P}_j^{\mathsf{Enc}})$: public key of user $j$.
INPUT: $ct_i = (\alpha_i, \beta_i)$: second-level ciphertext
PROCEDURE:
1. Compute $(m, r) = F_2(k_{i,2}, \alpha_i) \oplus \beta_i$.
2. Compute $u = F_1(k_{i,1}, (m, r))$.
3. If $\alpha_i \neq H(u, m, r)$, outputs $\perp$ and aborts. Otherwise, continue to the next steps.
4. Compute $r' = F_3(k_3, ct_i)$.
5. Compute $u' = F_1(k_{i,1}, (m, r'))$.
6. Evaluate $i\mathcal{O}(\mathcal{P}_j^{\mathsf{Enc}}, \lambda)(m, r', u')$.
7. Output $ct_j = (\alpha', \beta', u')$

**Fig. 5.** Program of Re-encryption Circuit

- $\mathsf{REnc}(rk_{i\to j} = \mathcal{Q}^{\mathsf{REnc}}, ct_i = (\alpha_i, \beta_i))$: The re-encryption algorithm takes as inputs $ct_i$ of a second-level ciphertext of user $i$ and a re-encryption key $rk_{i\to j}$ which is an obfuscated program $\mathcal{Q}_{i\to j}^{\mathsf{REnc}}$. It then runs the circuit $\mathcal{Q}_{i\to j}^{\mathsf{REnc}}(ct_i)$ and outputs a first-level ciphertext $ct_j = (\alpha', \beta', u')$.
- $\mathsf{Dec}_2(sk_i = (k_{i,1}, k_{i,2}), ct_i = (\alpha_i, \beta_i))$: The second-level decryption algorithm takes as inputs a secret key $sk$ and a ciphertext $ct_i$. At first it computes $(m, r) = F_2(k_{i,2}, \alpha_i) \oplus \beta_i$. Next, it computes $u = F_1(k_{i,1}, (m, r))$ and then checks the equation $\alpha_i = H(u, m, r)$. If the equation does not hold, it outputs $\perp$ and stops. Otherwise, it outputs $m$.
- $\mathsf{Dec}_1(sk_j, \widehat{ct_j})$: The first-level decryption algorithm takes as inputs a secret key $sk_j$ and a first-level ciphertext $\widehat{ct_j}$. At first it computes $(m, r) = F_2(k_{j,2}, \alpha')$, and checks if $vk(m, r, u') = 1$ and $\alpha = H(m, r, u') = 1$. Finally, it outputs a message $m$ if the equations hold or symbol $\perp$ otherwise.

**Correctness**. At first, we ensure the correctness of decryption of the original (second level) ciphertext. Actually, the second-level ciphertext has the form: $ct = (\alpha, \beta) = (H(m, r, u), F_2(k_2, \alpha) \oplus (m, r))$, and the secret key is $(k_1, k_2)$. When we decrypt the ciphertext, we calculate $(m, r) = F_2(k_2, \alpha) \oplus \beta$ and $u = F_1(k_1, (m, r))$. If the check $\alpha = H(m, r, u)$ holds, the result of decryption is valid.

Furthermore, we ensure the consistency of decryption of the (transformed) first-level ciphertext. The transformed (first-level) re-encrypted ciphertext has the form: $\widehat{ct} = (\alpha', \beta', u') = (H(m, r', u'), F_2(k_{j,2}, \alpha' \oplus (m, r')), F_{i,1}(k_{i,1}, (m, r')))$, and the secret key has the form: $sk_j = (k_{j,1}, k_{j,2})$. When decrypting a re-encrypted first-level ciphertext, we will compute $(m, r') = F_2(k_{j,2}, \alpha') \oplus \beta'$. If the check $vk(m, r', u') = 1$ and $\alpha = H(m, r', u')$ hold, the result of this decryp-

tion outputs the message $m$ which satisfies the consistency of the second-level ciphertext.

### 4.3 Proof of security

We use a series of games to prove the security of our scheme. In the sequence of games, the first game is defined as the original experiment of second-level CCA security. Then we show that any *p.p.t* adversary's advantage in each game must be negligibly close to the previous game and the adversary has negligible advantage in the final game. We have the following theorem.

**Theorem 1.** *Suppose that the indistinguishability obfuscation scheme is a secure $i\mathcal{O}$, $F_1$ and $F_2$ are secure puncturable PRFs, $f(\cdot)$ is a cryptographically secure one-way function and $H$ is modeled as a collision-resistant hash function, the proposed VPRE scheme is CCA secure for the second-level encryption.*

**Proof**. We give the proof that is based on a series of games as follows.

**Expt**$_0$: The first game $\text{Expt}_0$ is set as the original second-level CCA security game instantiated in our construction, which works as follows.

1. The adversary $\mathcal{A}$ selectively gives the challenger the messages $m^*$.
2. The challenger $\mathcal{C}$ at random selects keys $k_1 \leftarrow Key_{F_1}(1^\lambda), k_2 \leftarrow Key_{F_2}(1^\lambda), k_3 \leftarrow Key_{F_3}(1^\lambda)$ and also picks a random coin $b \in \{0,1\}$.
3. $\mathcal{C}$ computes $u^* = F_1(k_1, (m^*, r^*))$.
4. The challenger $\mathcal{C}$ creates $\mathcal{P}^{\mathsf{Enc}} \leftarrow i\mathcal{O}(1^\lambda, \texttt{Encrypt-Circuit} : [k_1])$ and sends $\mathcal{P}^{\mathsf{Enc}}$ to the adversary $\mathcal{A}$.
5. Phase-1 queries and response as follows in an adaptive manner:
   (a) The challenger $\mathcal{C}$ generates the re-encryption key $rk_{i\to j}$ by calling $\mathcal{Q}^{\mathsf{REnc}} \leftarrow i\mathcal{O}(1^\lambda, \texttt{ReEnc-Circuit} : [k_1, k_2, k_3, i\mathcal{O}(\mathcal{P}_j^{\mathsf{Enc}})])$ and returns $\mathcal{Q}^{\mathsf{REnc}}$ to $\mathcal{A}$.
   (b) $\mathcal{A}$ asks the query for ciphertext $ct$ to oracle $\mathcal{O}_{\mathsf{REnc}}$.
   (c) $\mathcal{A}$ requests the query for ciphertext $\widehat{ct}$ to oracle $\mathcal{O}_{\mathsf{Dec}_2}$ and re-encryption ciphertext $ct$ to oracle $\mathcal{O}_{\mathsf{Dec}_1}$.
6. The challenger $\mathcal{C}$ runs $ct^* \leftarrow i\mathcal{O}(1^\lambda, \texttt{Encrypt-Circuit}: [k_1])(m^*, r^*, u^*)$, and returns $ct^*$ to the adversary.
7. Phase-2 queries are the same as in Phase-1, except that, for the adversary $\mathcal{A}$, the following additional restrictions are satisfied:
   (a) Cannot request a re-encryption query to tuple $(pk_{i^*}, pk_j, ct^*)$ s.t. $pk_j \in pk_c$.
   (b) Cannot request a decryption query to $(pk_k, ct_k)$ so that $ct_k$ is the result of a re-encryption query $(pk_{i^*}, pk_k, ct^*)$.
   (c) Cannot request the decryption query to tuple $(pk_{i^*}, ct^*)$
8. The adversary $\mathcal{A}$ outputs a bit $b'$ and wins the game if $b' = b$.

**Expt**$_1$: The challenger $\mathcal{C}$ sets $\alpha^* = H(m^*, r^*, u^*)$ and $\widetilde{ct^*} = F_2(k_2, \alpha^*) \oplus (m^*, r^*)$. It creates the obfuscated program of $\mathcal{P}^{\mathsf{Enc}^*}$ as the obfuscation version of $\texttt{Encrypt-circuit}^*$ defined in Fig. 6. By the $i\mathcal{O}$ security, no adversary can distinguish **Expt**$_1$ and **Expt**$_0$.

**Encrypt-Circuit**$^*$

CONSTANTS: puncturable PRF keys $k_2\{\alpha^*\}$, $\alpha^*$ and $\widetilde{ct^*}$.
INPUT: message $m \in \{0,1\}^\ell$,
randomness $r \in \{0,1\}^\lambda$,
signature $u \in \{0,1\}^{\ell_\delta}$.
PROCEDURE:

1. If $\alpha^* = H(m, r, u)$, output $\widetilde{ct^*}$.
2. Else compute $\alpha = H(u, m, r)$.
3. Compute $\beta = F_2(k_2, \alpha) \oplus (m, r)$.
4. Output $ct = (\alpha, \beta)$.

**Fig. 6.** Program of Encrypt-Circuit$^*$

**Expt**$_2$: $\mathcal{C}$ computes $z^* = f(F_1(k_1, (m^*, r^*)))$ and sets $vk$ as the obfuscation of program `Verify-key`$^*$ defined in Fig. 7. It is easily to see that no adversary can distinguish **Expt**$_2$ and **Expt**$_1$ by the security of indistinguishability obfuscation.

**Verify-key**$^*$

CONSTANTS: puncturable PRF key $k_1\{m^*, r^*\}$,
$m^* \in \{0,1\}^\ell$,
$r^* \in \{0,1\}^\lambda$,
$z^*$.
INPUT: message $m \in \{0,1\}^\ell$,
randomness $r \in \{0,1\}^\lambda$,
signature $u \in \{0,1\}^{\ell_\delta}$.
PROCEDURE:

1. If $(m, r) = (m^*, r^*)$,
   Then, check whether $f(u) = z^*$.
   Output "*accept*" if the equation holds,
   and output "*reject*" otherwise.
2. Else, check if $f(u) = f(F_1(k_1, (m, r)))$.
   Output "*accept*" if the equation holds, and output "*reject*" otherwise.

**Fig. 7.** Program of Verify-key$^*$

**Expt**$_3$: The challenger sets $z^* = f(y)$ for randomly selected $y$ from $\{0,1\}^{\ell_\delta}$. By the security of puncturable PRF, no adversary can distinguish **Expt**$_3$ and **Expt**$_2$.

**Expt**$_4$: The challenger $\mathcal{C}$ at first computes the output when $ct^*$ are input to the re-encryption circuit $\mathcal{Q}^{\mathsf{REnc}^*}_{*\to j}$ defined in Fig. 8. Here, it hardwires the output $\widehat{ct^*}$ (i.e., re-encrypted ciphertext) to $\mathcal{Q}^{\mathsf{REnc}^*}$. Next, it computes punctured

keys $k^*_{i,1}\{m^*, r^*\}$, $k^*_{i,2}\{\alpha^*\}$ and $k_3\{ct^*\}$. By the security of indistinguishability obfuscation $i\mathcal{O}$ and the collision-resistance of hash function, it is easily to show that no adversary can distinguish $\mathbf{Expt}_4$ and $\mathbf{Expt}_3$.

**ReEnc-Circuit$^*$**

CONSTANTS: punctuable PRF keys $k^*_{i,1}\{m^*, r^*\}$, $k^*_{i,2}\{\alpha^*\}$(secret key of user $i$),
$k_3\{ct^*\}$ (puncturable PRF key),
$i\mathcal{O}(P^{\mathsf{Enc}}_j)$(public key of user $j$),
ciphertext $\widehat{ct^*}$.
INPUT: $ct_i$.
PROCEDURE:

1. If the input ciphertext is $ct^*$, then it outputs $\widehat{ct^*}$.
2. Compute $(m, r) = F_2(k^*_{i,2}\{\alpha^*\}, \alpha_i) \oplus \beta_i$.
3. Compute $u = F_1(k^*_{i,1}\{m^*, r^*\}, (m, r))$.
4. Check whether $\alpha_i \neq H(u, m, r)$ holds. If not, output $\perp$ and abort.
5. Compute $r' = F_3(k_3\{ct^*\}, ct_i)$ and $u' = F_1(k^*_{i,1}\{m^*, r^*\}, (m, r'))$
6. Compute $i\mathcal{O}(\mathcal{P}^{\mathsf{Enc}}_j, \lambda)(m, r', u')$.
7. Output $ct_j = (\alpha', \beta', u')$

**Fig. 8.** Program of REnc-Circuit$^*$

$\mathbf{Expt}_5$: The challenger replaces the second component of the hardwired ciphertext to a random one. By the pseudorandom security in punctured points of puncturable PRF, no adversary can distinguish $\mathbf{Expt}_5$ and $\mathbf{Expt}_4$.

$\mathbf{Expt}_6$: The challenger answers the re-encryption query $(pk_i, pk_j, ct_i)$ such that $pk_i = pk^*$. It does as follows:

1. If the input ciphertext is $ct^*$, then output $\widehat{ct^*}$.
2. Compute $(m, r) = F_2(k^*_{i,2}\{\alpha^*\}, \alpha_i) \oplus \beta_i$.
3. Compute $u = F_1(k^*_{i,1}\{m^*, r^*\}, (m, r))$.
4. Check whether $\alpha_i \neq H(u, m, r)$. If not, output $\perp$ and abort.
5. Compute $r' = F_3(k_3\{ct^*\}, ct_i)$ and $u' = F_1(k^*_{i,1}\{m^*, r^*\}, (m, r'))$
6. Compute $i\mathcal{O}(\mathcal{P}^{\mathsf{Enc}}_j, \lambda)(m, r', u')$ and send the result to the adversary.

By the collision-resistant security of hash function, there does not exist any adversary in distinguishing $\mathbf{Expt}_6$ and $\mathbf{Expt}_5$.

$\mathbf{Expt}_7$: The challenger $\mathcal{C}$ answers the first-level decryption query $(pk_i, \widehat{ct_i})$ as follows.

1. Compute $(m, r') = F_2(k_{i,2}, \alpha_i) \oplus \beta_i$.
2. Check whether $vk(m, r', u')$ and $\alpha_i \neq H(m, r', u')$. Output $\perp$ if fail.
3. Output $m$ as the answer.

By the security of one-way function and collision-resistance of hash function, there does not exist any adversary in distinguishing $\mathbf{Expt}_7$ and $\mathbf{Expt}_6$.

$\mathbf{Expt}_8$: The challenger $\mathcal{C}$ answers the second-level decryption query $(pk_i, ct)$ such that $pk_i = pk^*$ as follows:

1. Compute $(m, r) = F_2(k_{i,2}, \alpha_i) \oplus \beta_i$.
2. Compute $u = F_1(k_{i,1}, (m, r))$.
3. Check whether $\alpha_i \neq H(u, m, r)$ holds. If not, outputs $\perp$.
4. Return $m$ as the answer.

By the collision-resistance of hash function, it is easily to show that no adversary can distinguish $\mathbf{Expt}_8$ and $\mathbf{Expt}_7$.

$\mathbf{Expt}_9$: Replace $F_1(k_1, (m^*, r^*))$ with a randomly and uniformly selected value. By the security of puncturable PRF, no adversary can distinguish $\mathbf{Expt}_9$ from $\mathbf{Expt}_8$.

$\mathbf{Expt}_{10}$: The challenger $\mathcal{C}$ sets $\alpha^* = t^*$ for randomly selected $t^* \leftarrow \{0,1\}^{\ell_\alpha}$. By the security of puncturable PRF, no adversary can distinguish $\mathbf{Expt}_{10}$ and $\mathbf{Expt}_9$.

$\mathbf{Expt}_{11}$: The challenger $\mathcal{C}$ at random chooses $x^* \leftarrow \{0,1\}^{\ell+\lambda}$ and sets $(t^*, x^*)$ as the challenge ciphertext.

Notice that, in $\mathbf{Expt}_{11}$, the challenge ciphertext $ct^* = (t^*, x^*)$ where $t^*$ and $x^*$ are distributed uniformly, and thus, the adversary $\mathcal{A}$ has a negligible advantage in the second-level CCA-VPRE game. Therefore, the advantage of the adversary in $\mathbf{Expt}_0$ is negligible in actual attack experiment. This completes the proof of Theorem 1.

**Theorem 2.** *If the obfuscation scheme is a secure indistinguishably obfuscator, $F_1$ and $F_2$ are secure punctured PRFs, $f(\cdot)$ is a cryptographically secure one-way function and $H$ is a collision-resistant hash function, the proposed VPRE scheme is CCA secure for the first-level encryption.*

**Proof**. We also use a series of games that are proved to be indistinguishable as follows.

$\mathbf{Expt}_0$: $\mathrm{Expt}_0$ is described as the first-level CCA experiment of VPRE scheme.

$\mathbf{Expt}_1$: This game is the same as $\mathbf{Expt}_0$, except that the re-encrypted ciphertext is set as $(\alpha', \beta', u' = W)$ for randomly selected $W \in \{0,1\}^{\ell_\delta}$. By the security of puncturable PRF, it is easily to see that no adversary can distinguish $\mathbf{Expt}_1$ and $\mathbf{Expt}_0$.

$\mathbf{Expt}_2$: This game is the same as $\mathbf{Expt}_1$, except that the re-encrypted ciphertext is set as $(\alpha' = U, \beta', W)$ for randomly selected $U \in \{0,1\}^{\ell_\alpha}$. By the security of puncturable PRF and collision-resistance of hash function, no adversary can distinguish $\mathbf{Expt}_2$ and $\mathbf{Expt}_1$.

$\mathbf{Expt}_3$: This game is the same as $\mathbf{Expt}_1$, except that the re-encrypted ciphertext is set as $(U, \beta' = V, W)$ for randomly chosen $V \in \{0,1\}^{\ell+\lambda}$. By the security of puncturable PRF, it is easily to show that no adversary is able to distinguish $\mathbf{Expt}_3$ and $\mathbf{Expt}_2$.

By a series of hybrid arguments, it declares that a *p.p.t* adversary's advantage in the original security $\mathbf{Expt}_0$ can be at most negligibly greater than its advantage in $\mathbf{Expt}_3$. We note that the advantage of the adversary in $\mathbf{Expt}_3$ is

negligible in security parameter $\lambda$, since it provides no information on the coin $b$ and thus completes the proof of Theorem 2.

**Theorem 3.** *Suppose that $i\mathcal{O}$ is a secure indistinguishability obfuscator in Definition 1, then the proposed scheme is master secret-key secure.*

**Proof**. Suppose that, in the VPRE scheme, the master secret-key $sk_i$ is revealed when the malicious cloud colludes with the delegatee $j$, then we can construct an $i\mathcal{O}$ distinguisher $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ to distinguish the obfuscated circuits in the circuit family. The deployment of $\mathcal{B}$ works as follows:

At first, $\mathcal{B}_1$ constructs a re-encryption key $\mathcal{Q}^{\mathsf{REnc}}$ as in Fig. 5. We denote this circuit as $C_0$. Next, $\mathcal{B}_1$ constructs a re-encryption key $\mathcal{Q}^{\mathsf{REnc}^*}$ as in Fig. 8. We denote this circuit as $C_1$. Note that the functionality of these two circuits $C_0$ and $C_1$ are completely the same. $\mathcal{B}_1$ outputs $(C_0, C_1)$ and aborts.

$\mathcal{B}_2$ is given $i\mathcal{O}(1^\lambda, C^*)$ from the challenger $\mathcal{C}$. That is, this $i\mathcal{O}(1^\lambda, C^*)$ is either $i\mathcal{O}(1^\lambda, C_0)$ or $i\mathcal{O}(1^\lambda, C_1)$. When the re-encryption key queried from the challenge user to $j$ by the adversary$\mathcal{A}$, $\mathcal{B}_2$ returns the $i\mathcal{O}(1^\lambda, C^*)$ to $\mathcal{A}$ and receives a secret key $sk$. If $sk = sk^*$, $\mathcal{B}_2$ decides that $i\mathcal{O}(1^\lambda, C^*)$ is $C_0$. Otherwise, $\mathcal{B}_2$ decicdes that $i\mathcal{O}(1^\lambda, C^*)$ is $C_1$. Obviously, a $p.p.t$ adversary can distinguish between $C_0$ and $C_1$ which will lead the constructed algorithm $\mathcal{B}$ to break the indistinguishability security of $i\mathcal{O}$. As we employ the secure $i\mathcal{O}$, we conclude that the propose VPRE scheme satisfies the master secret-key security.

**Theorem 4.** *Suppose that $f(\cdot)$ is a secure one-way function and $i\mathcal{O}$ is a secure indistinguishability obfuscator, then the proposed VPRE scheme is verifiably secure of re-encryption.*

**Proof**. If there exists an adversary $\mathcal{A}$ who can against the verifiable security of re-encryption, we can construct an algorithm $\mathcal{B}$ to break the security of one-way function $f$.

At first, algorithm $\mathcal{B}$ sets a verify key circuit defined in Fig. 4. Next, $\mathcal{B}$ runs $\mathcal{P}^{\mathsf{Enc}}$ to obtain a challenge ciphertext $ct^* = (\alpha^*, \beta^*)$, and sends the tuple $(ct^*, vk)$ to adversary $\mathcal{A}$. Later, $\mathcal{A}$ outputs $\widehat{ct} = (\alpha', \beta', \sigma)$. By the definition of $\mathcal{B}$ that has computed $\sigma$ such that $f(\sigma) = y$. We say that $\mathcal{A}$ will win if and only if (1) $m' \neq m$, (2) $m' \neq \perp$ and, (3) $vk(m, r', \sigma) = 1$. If the one-way function $f$ is cryptographically secure, it is easily to show that no $p.p.t$ adversary $\mathcal{A}$ in the above equations with non-negligible advantage.

## 5 Deployment in Secure Data-Sharing in Cloud

In this section, we present a practical deployment of secure data-sharing that empolys our scheme as the basic primitive.

Assume that user $A$ wants to share his sensitive sports data or holiday photos in his smart watch to his friend circle shown in Fig. 9. In order to keep the privacy of the data, user $A$ needs to encrypt the sensitive data with his own public-key $pk_A$ and then stores $ct_A$ on the clouds[29]. When he is going to share

the encrypted data, he can generate a re-encryption key for the friend group, e.g., $rk_{A\to G_1}$, and requests the cloud server to perform the re-encryption program to create the re-encrypted ciphertext so that all the members in the group can obtain the clear sport data or photos by decrypting the re-encrypted ciphertexts.

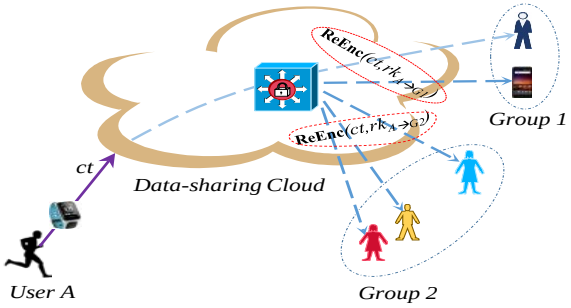


**Fig. 9.** Scenario of secure data-sharing in clouds

It is easy to see that, using our proposed VPRE as primitive in cloud-based data-sharing environments, it has the following benefits:

1. *Data sharing and storing security.* The sensitive data are encrypted and shared in a secure manner in which the cloud server can perform the sharing program without obtaining any shared clear-data. Actually, the original data are encrypted by the sharer, i.e., using the public key of user $A$ in Fig. 9, and the encrypted data are stored on the cloud.
2. *Sensitive-data protection in the transformation in the presence of white-box access.* If $A$ wants to share his data, he can creates a re-encryption key to the cloud and allows the cloud to perform the data-sharing transformation (i.e., re-encryption program) on inputs the re-encryption key and the encrypted data. We ensure that, even the cloud executes the re-encryption program in white-box manner (i.e., debug the program, monitor the memory and register and set the breakpoints etc.), the cloud server cannot gain any embedded sensitive information such as cleartext data and secret key.
3. *Sharer privacy preservation.* Even the cloud colludes with terminal users, i.e., user in Group 1 and Group 2 in Fig. 9, it cannot obtain the sharer's secret key, which is guaranteed by the master secret-key security of our scheme.
4. *Data-sharing for group users.* We can facilitate a group of users and set a group key, and use this group public-key to create the re-encryption key. Any user in the group can decrypt the re-encrypted ciphertext and it will improve the data-sharing efficiency.
5. *Reasonable allocate the operation.* In our deployment, the time-consuming operations are computed by the cloud, and the operations of terminal user in

Group 1 or Group 2 is very fast as it only needs several symmetric PRF operations. We can effectively deploy the data-sharing to light-weight nodes such as Wireless Sensor Networks, Wireless Body Area Network and Internet-of-Things etc.

## 6 Conclusion

In this paper, we presented a cloud-based data-sharing scheme that is based on a cloud-based re-encryption scheme by using the cryptographic primitives of indistinguishability obfuscation and puncturable pseudorandom functions. Our scheme provides several helpful properties such as white-box security in the secure data-sharing (re-encryption), CCA security of both first-level cipertext and second-ciphertext, re-encryption verifiability of master secret-key security, and reasonable allocation of the operations. Moreover, the proposed scheme is efficient in decryption since it only needs several symmetric PRF operations, which is fruitful to deploy the scheme in light-weight nodes such as WSNs, WBANs and IoTs.

## References

1. Asharov, G., Segev, G., Limits on the power of indistinguishability obfuscation and functional encryption. In: 56th FOCS'15, pp. 191-209 (2015)
2. Ateniese, G., Fu, K., Green, M., and Hohenberger, S., Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Transactions on Information and System Security, 9(1), 1-30 (2006).
3. Bishop, A., Kowalczyk, L., Malkin, T., Pastro, V., Raykova, M., and Shi, K., A simple obfuscation scheme for pattern-matching with wildcards. CRYPTO'18, LNCS 10993, pp 731-752 (2018)
4. Blaze, M., Gerrit, B., and Martin, S., Divertible protocols and atomic proxy cryptography. In: EURCRYPT'98, LNCS 1403, pp. 127-144 (1998).
5. Boneh, D., Gupta, D., Mironov, I., Sahai, A., Hosting services on an untrusted cloud. In: EUROCRYPT'15. LNCS 9057, pp 404-436 (2015)
6. Boneh, D., Waters, B., Constrained pseudorandom functions and their applications. In: ASIACRYPT'13, Part II. LNCS 8270, pp 280-300 (2013)
7. Canetti, R., and Hohenberger, S.: Chosen-ciphertext secure proxy re-encryption. In Proceedings of the 14th ACM conference on Computer and communications security, pp. 185-194, ACM (2007).
8. Chen, M.R., Zhang, X., Li, X., Comments on Shao-Cao's unidirectional proxy re-encryption scheme from PKC 2009. Journal of Information Science and Engineering, 27 (3): 1153-1158 (2011)
9. Chow, S.S., Weng, J., Yang, Y., and Deng, R. H., Efficient unidirectional proxy re-encryption. In Africacrypt'10, LNCS 6055, pp. 316-332 (2010).
10. Cohen, A., Holmgren, J., Nishimaki, R., Vaikuntanathan, V., Wichs, D., Watermarking cryptographic capabilities. In: 48th ACM STOC'16, pp. 1115-1127 (2016)
11. Gentry, C., Lewko, A.B., Sahai, A., Waters, B., Indistinguishability obfuscation from the multilinear subgroup elimination assumption. In: FOCS'15, pp. 151170 (2015)

12. Goldreich, O., Goldwasser, S., and Micali, S., How to construct random functions. Journal of the ACM, 33(4):792807 (1986).
13. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., and Waters, B., Candidate indistinguishability obfuscation and functional encrytion for all circuits. In: FOCS'13, pp. 40-49. IEEE (2013).
14. Gaurav, P. Purushothama, B. R., Proxy visible re-encryption scheme with application to e-mail forwarding. Proceedings of the 10th International Conference on Security of Information and Networks(SIN'17), pp 212-217 (2017)
15. Gaurav, P. Purushothama, B. R., On efficient access control mechanisms in hierarchy using unidirectional and transitive proxy re-encryption schemes. In Proceedings of the 14th International Joint Conference on e-Business and Telecommunications (ICETE'17), pp 519-524 (2017)
16. Komargodski, I., and Yogev, E., Another step towards realizing random oracles: non-malleable point obfuscation. EUROCRYPT'18, LNCS LNCS 10820 Part I, pp 259-279 (2018)
17. Kitagawa, F., Nishimaki, R., and Tanaka, K., Obfustopia built on secret-key functional encryption. EUROCRYPT'18, LNCS 10821 Part II, pp 603-648 (2018)
18. Hanaoka, G., Kawai, Y., Kunihiro, N., Matsuda, T., Weng, J., Zhang, R., and Zhao, Y., Generic construction of chosen ciphertext secure proxy re-encryption. In: CT-RSA'12, LNCS 7178, pp. 349-364 (2012).
19. Hohenberger, S., Rothblum, G. N., and Vaikuntanathan, V.: Securely obfuscating re-encryption. In: TCC'07, LNCS 4392, pp. 233-252 (2007).
20. Hohenberger, S., Koppula, V., Waters, B., Adaptively secure puncturable pseudorandom functions in the standard model. ASIACRYPT'15, LNCS 9452, pp 79-102 (2015)
21. Isshiki, T., Nguyen, M. H., and Tanaka, K., Proxy re-encryption in a stronger security model extended from CT-RSA2012. In: CT-RSA'13, LNCS 7779, pp. 277-292 (2013).
22. Kirshanova, E., Proxy re-encryption from lattices. In: PKC'14, LNCS 8383 pp. 77-94 (2014).
23. Lai, J., Huang, Z., Au, M.H, and Mao X., Constant-size CCA-secure multi-hop unidirectional proxy re-encryption from indistinguishability obfuscation. ACISP'18, LNCS 10946, pp. 805-812(2018)
24. Libert, B., and Vergnaud, D.: Unidirectional chosen-ciphertext secure proxy re-encryption. In: PKC'08, LNCS 4939, pp. 360-379 (2008).
25. Liu, M., Wu, Y., Chang, J., Xue, R., and Guo, W., Verifiable proxy re-encryption from indistinguishability obfuscation. In: ICICS'15, LNCS 9543, pp. 363-378. Springer (2015).
26. Ohata, S., Kawai, Y., Matsuda, T., Hanaoka, G., and Matsuura, K.: Re-encryption verifiability: how to detect malicious activities of a proxy in proxy re-encryption. In: CT-RSA'15, LNCS 9048, pp. 410-428. Springer (2015).
27. Sahai, A., and Waters, B., How to use indistinguishability obfuscation: deniable encryption, and more. In: STOC'14, pp. 475-484. ACM (2014).
28. Zhang, M., Jiang, Y., Mu, Y. and Susilo, W., Obfuscating re-encryption algorithm with flexible and controllable multi-hop on untrusted outsourcing server. IEEE Access, 5(1):26419-26434 (2017).
29. Zhang, M., Yao, Y., Li, B., Tang, C., Accountable mobile e-commerce scheme in intelligent cloud system transactions. Journal of Ambient Intelligence and Humanized Computing, 9(6):1889-1899, Springer (2018).