JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 33, 993-1006 (2017)
DOI: 10.6688/JISE.2017.33.4.9

Efficiently Obfuscating Anonymous Re-encryption Functionality with Average-Case Virtual Black-Box Security

MINGWU ZHANG[1,2], YUDI ZHANG[1], HUA SHEN[1], CHUNMING TANG[3] AND LEIN HARN[4]
[1]*School of Computers*
Hubei University of Technology
Wuhan, 430068 P.R. China
[2]*School of Computer and Software*
Nanjing University of Information Science and Technology
Nanjing, 210044 P.R. China
[3]*Key Laboratory of Mathematics and Interdisciplinary Sciences of Guangdong Higher Education Institutes*
Guangzhou University
Guangzhou, 510006 P.R. China
[4]*University of Missouri*
Kansas City, MO 65211 USA
E-mail: csmwzhang@gmail.com

Outsource computing might reveal some private information in open cloud systems and the concern of users' privacy will be a crucial issue. Program obfuscation is an important cryptographic primitive that perfectly hides the secrets inside a program while preserving its functionality. In this paper, we give a two-form re-encryption scheme that achieves the security against adaptively chosen-ciphertext attacks, which allows any third party (*e.g.*, cloud server) to re-encrypt the ciphertext that delegates the decryption right from one to another. However, as the third party knows the sensitive secret (*i.e.*, re-key), and thus we should guarantee the trustworthy of this cloud server. Based the re-encryption algorithm, we propose an efficient obfuscator that implements the re-encryption functionality in such a way that not only the private information is protected but also cloud users' privacy is preserved, and thus the re-encryption procedure can be run on untrusted outsourcing server. We prove the virtual black-box security of the obfuscation. The proposed obfuscator can be run on untrusted server to help the perform of the functionality of re-encryption without any sensitive secret revealing.

Keywords: cyptographic obfuscation, unidirectional, re-encryption, virtual-black box, bilinear map

1. INTRODUCTION

Cloud computing has become an increasingly popular computing paradigm that provides users and enterprises with various capabilities to store and process their data in third-party data centers, such as in an untrusted server. However, when outsourcing data and heavy computation tasks to the cloud, cloud users can enjoy these benefits only when the cloud is fully trusted. Therefore, cloud users are more concerned about their privacy whether it is leaked in this environment.

In this paper, we present an algorithm ***ReEncObf*** to obfuscate a re-encryption cir-

Received July 16, 2016; revised August 17, 2016; accepted September 12, 2016.
Communicated by Zhe Liu.
[*] This work is supported by the National Natural Science Foundation of China under grant 61370224 and 61672010, the CICAEET fund and the PAPD fund, and the Key Laboratory of Mathematics and Interdisciplinary Sciences of Guangdong Higher Education Institutes, Guangzhou University.

cuit efficiently. Actually, a re-encryption algorithm is to delegate the decryption ability by re-encrypting a ciphertext of cloud user *Alice* to a new form ciphertext of *Bob*. Concretely, this re-encryption algorithm is performed by (untrusted) outsourced cloud server, however, the server may obtain the sensitive information such as the keys of *Alice* or *Bob* since the re-encryption procedure needs the keys of *Alice* or *Bob*. We consider the issues as follows:

(1) Does the algorithm executor (*i.e.*, untrusted outsourcing server) cannot learn sensitive information from the re-encryption key or ciphertext? *e.g.* plaintext or secret key during the computation;
(2) May *Bob* re-delegate the re-encrypted ciphertext to another for decryption successfully? *e.g.* only a single-hop;
(3) Could the algorithm executor derive users' identities, the delegator *Alice* or the delegatee *Bob*?

In order to solve above issues, we develop a program obfuscator to perform the re-encryption while preventing the untrusted outsourcing server from gaining the private information. A program obfuscator is a compiler that takes a program (namely, a binary circuit) as input and outputs an equivalent and unintelligible program. According to the requirement and the property defined in literatures [1, 2], an obfuscator must satisfy the properties: preserves the functionality, incurs a polynomial slowdown (compared to the original program) and satisfies the virtual black-box security property. That is, an obfuscated program can prevent the attacker from reversing engineering thus it could be applied on intellectual property protection for embedded sensitive algorithms [1, 3, 4] or softwares/fingerprints [5-11], controlled delegation [11-19] and encrypted signatures [20] *etc.*

In the program obfuscator ***ReEncObf***, the main goal is to securely enable the re-encryption of message from one to another, without relying on trusted proxy. We also require that process of the re-encryption is unidirectional, anonymous and single-hop.

The main contribution of this paper is to design a virtual black-box (VBB) secure program obfuscator with the functionality of anonymous re-encryption algorithm. Concretely, the obfuscator algorithm can re-encrypt the cipertext for one-hop manner and provides the anonymous property. Moreover, the output of the obfuscated circuit cannot be re-encrypted anymore [4, 6, 21-23]. At first we give two-type of encryption schemes under an indicator β. The first form ciphertext $\beta = 0$ can be converted to the second form ciphertext $\beta = 1$ under some auxiliary information. We then propose the obfuscation algorithm to implement the anonymous re-encryption algorithm, and provide the analysis of functionality and prove the virtual black-box property of security. The obfuscator algorithm can be run by any untrusted proxy without the leakage of sensitive information such as secret key and plaintext during the execution.

2. PRELIMINARIES

A function is said to be negligible in security parameter l (denoted $negl(l)$) if it is smaller than inverse of any polynomial, for all large enough value of l. In this paper, we use the term PPT to denote the Probabilistic Polynomial Time algorithm. Let $s \xleftarrow{r} S$

denote selecting element s from set S at random. Let $\mathbb{G} = <g>$ be a finite group $\mathbb{G}$ of order q created by a generator g, and denote $\mathbb{G}_T$ be a bilinear group such that a map e is defined by $\mathbb{G}\times\mathbb{G}\rightarrow\mathbb{G}_T$.

Definition 1 (Average-Case Secure Obfuscation): An algorithm ***Obf*** that takes as input a (probabilistic) circuit C and outputs a new (probabilistic) circuit C', is an average-case secure obfuscator for the family $C = \{C_k\}$, if it holds the following properties:

1. **Preserving Functionality:** $Pr[C(x)\neq C'(x)] < neg(l)$
2. **Polynomial Slowdown:** There exists a polynomial $p(|C|)$ such that for sufficiently large input lengths n, for any $C\in\{C_k\}$, **Obf** only enlarges C by a factor p, *i.e.* $|Obf(C)| \leq p(|C|)$.
3. **Virtual Black-box Security:** For any PPT adversary $\mathcal{A}$, there exists a PPT simulator Sim, for every efficient distinguisher D, every input length n and for every polynomial-size auxiliary input aux:

$$|\Pr[C \xleftarrow{r} C_k: D^c(\mathcal{A}(Obf(C), aux), aux) = 1]$$
$$-\Pr[C \xleftarrow{r} C_k: D^C(Sim^C(1^n, aux), aux) = 1)]| < negl(l)$$

We let $\perp$ denote an error message, and $||$ denote string concatenation. Let U be the set of users, denoted by $U = \{1, \ldots, n\}$. Denote the set of uncorrupted users by U_h and set of corrupted users by U_e, such that $U_h \cup U_e = U$. The proxy re-encryption (PRE) [21, 24] and the CCA notion of PRE are defined as follows.

Definition 2 (A single-hop, unidirectional PRE): A single-hop, unidirectional proxy re-encryption is comprised of the following algorithms:

1. ***Setup***(1^l): On input a security parameter 1^l, this algorithm generates the public parameter **Param**. Note that the public parameter **Param** is used in the other algorithms and we take it as input implicitly.
2. ***KeyGen***(i)***:*** On input the parameter **Param**, this algorithm is run by user i to generate public and private key pairs (pk_i, sk_i).
3. ***ReKeyGen***(pk_i, sk_i, pk_j): On input a secret key sk_i and public key pk_j, this algorithm is run by user i to output a re-encryption key $rk_{i\rightarrow j}$, which can be used to re-encrypt a first form ciphertext of user i to a second form which can decrypted by user j.
4. ***Enc***(pk_i, m): Taking public key pk_i and a message m as inputs, this algorithm is run to encrypt a message m to a first form cipertext CT.
5. ***ReEnc***$(rk_{i\rightarrow j}, CT)$: On input a re-encryption key $rk_{i\rightarrow j}$, and a ciphertext CT, this algorithm is run to re-encrypt a first form ciphertext CT to a second form CT'.
6. ***Dec***$(sk_{i(j)}, CT(CT'))$: On input a secret key $sk_{i(j)}$ and a cihpertext CT or CT', this algorithm is run to decrypt a first form ciphertext CT (or a second form ciphertext CT'), and outputs the message m if succeeds or an error $\perp$ if fails.

Correctness: A single-hop, unidirectional PRE is correct if: at random select m from the message space, $\forall i, j \in U$, $pk_i, sk_i \leftarrow \text{KeyGen}(i)$, $pk_j, sk_j \leftarrow \text{KeyGen}(j)$, and $rk_{i\rightarrow j} \leftarrow \text{ReKeyGen}(sk_i, pk_j)$, we have $\text{Dec}(sk_j, \text{ReEnc}(rk_{i\rightarrow j}, Enc(pk_i, m))) = m$.

We use the terms $O_{\text{KeyGen()}}$ and $O_{\text{Dec()}}$ to denote the oracle access to the **KeyGen** algorithm and **Dec** algorithm, respectively.

Definition 3 (CCA-Security): Let $\mathcal{A}$ be a PPT adversary, $\mathcal{C}$ be a challenger. The CCA security is defined by the following game with a negligible advantage.

l. Key generation $O_{\text{KevGen}}(1^l)$: $(pk_i, sk_i) \leftarrow O_{\text{KevGen}}(1^l)$.
2. $(m_0, m_1, i, aux) \leftarrow \mathcal{A}^{O_{Dec}(\cdot)}(l)$, $m_0, m_1 \in M$ s.t. $|m_0| = |m_1|$.
3. $CT \leftarrow \text{Enc}(pk_i, m_\lambda)$, where $\lambda \xleftarrow{r} \{0, 1\}$.
4. $\lambda \leftarrow \mathcal{A}^{O_{Dec}(\cdot)}(\text{CT}, aux)$, s.t. $O_{\text{Dec}}(\cdot) \neq O_{\text{Dec}}(\text{CT})$.
5. Challenge: $\mathcal{A}$ outputs (pk_i, m_0, m_1), and sends to $\mathcal{C}$, where pk_i is the challenge key. $\mathcal{C}$ at random selects $\lambda \leftarrow \{0, 1\}$, and runs $CT^* \leftarrow \text{Enc}(pk_i, m_\lambda)$, then returns CT^* to $\mathcal{A}$.
6. Guess: Finally, $\mathcal{A}$ outputs $\lambda' \leftarrow \{0, 1\}$, and $\mathcal{A}$wins the game when $\lambda' = \lambda$.

Definition 4 (Anonymity): Let $\mathcal{C}$ maintain a re-encryption key list storing all $rk_{i\to j}$ with the form $(i\to j, \{rk_{i\to j}, \text{query}\})$. Let $i\to j$ be the search key to identify each pair of users. If $rk_{i\to j}$ has been queried by oracle $O_{ReKeyGen}$, then set $query = 1$, otherwise set $query = 0$. Initially, the list is $\{rk_{i\to j}, \text{query}\}$ empty for the search key (i, j). The anonymous game is defined as follows:

1. $O_{\text{KeyGen}}(1^l)$: same as in **Definition 3**.
2. $O_{\text{ReKeyGen}}(pk_i, pk_j)$: $\mathcal{C} \leftarrow (i \to j, \{rk_{i\to j}, \text{query}\})$ and works as follows:
   (1) If $\{rk_{i\to j}, \text{query}\}$ is null or all $rk_{i\to j}$ have been queried, sets $rk_{i\to j} \leftarrow \text{ReKenGen}(sk_i, pk_j)$, adds $(rk_{i\to j}, query = 1)$ to the list $(rk_{i\to j}, query)$, and the returns $rk_{i\to j}$.
   (2) Otherwise, sets $rk_{i\to j} \xleftarrow{r} \{(rk_{i\to j}, query = 0)\}$, updates $query = 1$ for $rk_{i\to j}$, and returns $rk_{i\to j}$ to $\mathcal{A}$.
3. $O_{\text{ReEnc}}(pk_i, pk_j, CT)$: $\mathcal{C} \leftarrow (i \to j, \{rk_{i\to j}, query\})$ and works as follows:
4. $O_{\text{Dec}}(pk_i, CT)$: performs $\{m, \perp\} \leftarrow \text{Dec}(sk_i, CT)$, and returns the result to $\mathcal{A}$.
5. Challenge $(i^*, j^*) \leftarrow \mathcal{A}$, $\mathcal{C} \leftarrow (i^*, j^*, \{rk_{i^*\to j^*}, query\})$, works as follows:
   (1) If $i^* \in U_e$ or $j^* \in U_e$, $\mathcal{C}$ aborts.
   (2) If the list $\{(rk_{i^*\to j^*}, query)\}$ is null or all $rk_{i^*\to j^*}$ have been queried, compute $rk_{i^*\to j^*} \leftarrow \text{ReKenGen}\{(sk_i, pk_j)\}$ to the set $\{(rk_{i^*\to j^*}, query)\}$.
   (3) Otherwise sets $rk_{i^*\to j^*} \xleftarrow{r} \{(rk_{i^*\to j^*}, query = 0)\}$, and updates $query = 1$ for $rk_{i^*\to j^*}$. $\mathcal{C}$ at random selects $\lambda \xleftarrow{r} \{0, 1\}$, if $\lambda = 0$, sets $\mathcal{A} \leftarrow rk_{i^*\to j^*}$, otherwise returns a re-encryption key selected from the re-encryption key space uniformly.
6. Guess: Finally, $\mathcal{A}$ output a guess $\lambda' \in \{0, 1\}$, and $\mathcal{A}$ wins the game if $\lambda' = \lambda$.

3. CCA-SECURE ENCRYPTION AND RE-ENCRYPTION FUNCTION

Like in [10], we give the global parameters of bilinear group generator, some secure hash functions and secure pseudorandom generators that will be used in our scheme. The encryption scheme Π can output two types of ciphertexts. Also, we use a flag β to assign the ciphertext form, when $\beta = 0$, the ciphertext has first form, and when $\beta = 1$, the ciphertext has second form. The first form of ciphertext can be re-encrypted into the second form under some auxiliary information, but the second form of ciphertext cannot be

converted to the first form (*i.e.*, unidirectional). We prove that the encryption scheme Π achieves CCA security. In this section, we first give the global parameters of bilinear group generator, and then propose the encryption scheme and the re-encryption scheme.

3.1 Cryptographic Assumptions

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

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

We will use the following security assumption.

Definition 5 (Decisional Bilinear Diffie-Hellman (DBDH)): $\forall x, y \xleftarrow{r} \mathbb{Z}_q$ and $Q \xleftarrow{r}$ $\mathbb{G}_T$, given (g, g^x, g^y, g^z, Q) for any PPT algorithm A, the following two distributions are computationally indistinguishable:

$$\{g, g^x, g^y, g^z, g^{xyz}\} \approx \{g, g^x, g^y, g^z, Q\}.$$

Definition 6 (Squared Decisional Bilinear Diffie-Hellman (SDBDH)): $\forall x, y \xleftarrow{r} \mathbb{Z}_q$ and $Q \xleftarrow{r} \mathbb{G}_T$, given the tuple (g, g^x, g^y, Q) for any PPT algorithm A, the following two distributions are computationally indistinguishable:

$$\{g, g^x, g^y, e(g, g)^{x^2y}\} \approx \{g, g^x, g^y, Q\}.$$

3.2 CCA-Secure Encryption Scheme Π

Under the idea in [19], we give the extensional two-form CCA secure encryption as follows:

Setup(1^ℓ): Given the security parameter ℓ, this algorithm obtains two secondary security parameters ℓ_1, ℓ_2 and instantiates H_0, H_1, H_2, H_3, H_4, F_1, F_2, and the bilinear map $(q, g, \mathbb{G}, \mathbb{G}_T, e)$, where q is a ℓ_1-bit prime and the message $m\in\{0, 1\}^{\ell_2}$. In addition, let $g_1 \xleftarrow{r} \mathbb{G}$ and set $Param = \{q, g, g_1, \mathbb{G}, \mathbb{G}_T, e\}$

KeyGen(1^ℓ, Param): At random select $x \xleftarrow{r} \mathbb{Z}_q$, and set $sk = x$ and $pk = g^x$.

Enc(pk, β, m):

1. Parse $pk = g^x$.
2. Choose $R \xleftarrow{r} \mathbb{G}_T$ randomly, and compute $r = H_1(m, R)$.
3. If $\beta = 0$, output ciphertext $(C_0, C_1, C_2, C_3, C_4, C_5) = (0, g^r, R\cdot e(g^x, H_0(g^x))^r, m\oplus F_1(R),$ $g_1^r, H_2(C_0, C_1, C_2, C_3, C_4)^r)$.
4. If $\beta = 1$, at random select s, t, $w \xleftarrow{r} \mathbb{Z}_q$, let $R_1 = g^{H_4(s)}\cdot g_1^t$, $R_2 = g^t$, $R_3 = g^w$, $R_4 = e(g^x, g^x)^{tw}$, $R_5 = e(g^x, g)^{tw}$, and let $T_1 = g^r$, $T_2 = R\cdot e(g^r, g^{x\cdot H_4(s)})$, $T_3 = m\oplus F_1(R)$, $T_4 = g_1^r$, and set $\Gamma = T_1\|T_2\|T_3\|T_4\|R_1\|R_2\|R_3$, random select $R' \xleftarrow{r} \mathbb{G}_T$, let $r' = H_3(\Gamma, R')$, output the ciphertext $(C'_0, C'_1, C'_2, C'_3, C'_4, C'_5) = (1, R_5^{r'}, R'\cdot R_4^{r'}, \Gamma\oplus F_2(R'), \phi, \phi)$.

Dec(sk, CT):
1. At first parse $sk = x$.
2. If $\beta = 0$, check the equation $e(g, C_5) \neq e(C_1, H_2(C_0, C_1, C_2, C_3, C_4)$ holds, then output $\perp$. Otherwise let $R = C_2/e(g^r, H_0(g^x))^x$, and compute $m = (C_3 \oplus F_1(R))$.
3. If $\beta = 1$, compute $R' = C'_2/C'^x_1$, let $\Gamma = C'_3 \oplus F_2(R')$, and parse $\Gamma = T_1||T_2||T_3||T_4||R_1||R_2||R_3$. If $e(R_2, R_3)^{xH_3(\Gamma, R')} \neq C'_1$ holds, then return $\perp$. Otherwise set $R = T_2(\frac{e(R_2, T_4)}{r(R_1, T_1)})^x$, and output $m = T_3 \oplus F_1(R)$.

Fig. 1. Two-form CCA-secure encryption scheme Π.

Theorem 1: Assume that the DBDH assumption holds the scheme is CCA secure in the random oracle model for the first form ciphertext, and the SDBDH assumption holds, then the second ciphertext is CCA secure.

3.3 Re-encryption Functionality

Let (pk_1, sk_1) and (pk_2, sk_2) be two key-pairs which are generated by KeyGen algorithm, the re-encryption functionality of performing re-encryption from pk_1 to pk_2 is described as follows.

$\text{ReEnc}_{1\to 2}$
Input: $CT = [C_0, C_1, C_2, C_3, C_4, C_5]$ or special symbol denoted by **keys**
Constants: $sk_1 = x$, $pk_1 = g^x$, $pk_2 = g^y$, $R' \xleftarrow{r} \mathbb{G}_T$, and let $r = H_1(m, R)$
1. If input = **keys**, output (pk_1, sk_2)
2. If $C_0 = 1$, output $\perp$
3. Else
   – Compute $R = C_2/e(g^r, H_0(g^x))^x$
   – Output = $(C_3 \oplus F_1(R))$
   – Encrypt m as same as the Step 4 in encryption of Π
Output: (C'_0, C'_1, C'_2, C'_3, C'_4, C'_5)

Fig. 2. Functionality of $\text{ReEnc}_{1\to 2}$.

4. OBFUSCATOR FOR CCA-SECURE RE-ENCRYPTION FUNCTION

4.1 Algorithm for CCA-Secure Re-encryption Functionality

Let $pk_1 = g^x$, $pk_2 = g^y$, and let C denote the re-encryption circuit C_{sk_1,pk_2}, and let $R_{sk_1,pk_2} \leftarrow Obf(C)$ be an obfuscated version of C. The obfuscator for re-encryption circuits is described as follows:

Algorithm **Obf**, on input a circuit $C_{sk_1,pk_2} \in C_k$
1. Read $sk_1 = x$, $pk_2 = g^y$, from the description of C_{sk_1,pk_2}
2. At random select s, t, $w \xleftarrow{r} Z_q$, generates the re-encryption key $rk_{1\to 2} = (rk_1, rk_2, rk_3, rk_4, rk_5, rk_6)$ as $(H_0(g^x)^{-x} \cdot (g^y)^{H_4(s\cdot x)}, g^{H_4(s\cdot x)} \cdot g_1^t, g^t, g^w, e(g^y, g^y)^{tw}, e(g^y, g)^{tw})$
3. Construct and output an obfuscated circuit R_{sk_1,pk_2} that contains the values pk_1, pk_2,

$rk_{1\to 2}$ as follows:
– On input **keys**, output $(pk_1, pk_2) = (g^x, g^y)$
– On input 6-tuple $[C_0, C_1, C_2, C_3, C_4, C_5]$
(a) At first check $e(g, C_5) = e(C_1, H_2(C_1, C_2, C_3, C_4))$. If the equation does not hold, it aborts.
(b) Otherwise, let
$T_1 = C_1 = g^r$, $T_1 = C_1 = g^r$
$T_2 = C_2 \cdot e(C_1, rk_1) = R \cdot e(g^r, (g^y)^{H_4(s\|x)})$,
$T_3 = C_3 = m \oplus F_1(R)$,
$T_4 = C_4 = g_1^r$, and
set $\Gamma = T_1\|T_2\|T_3\|T_4\|rk_2\|rk_3\|rk_4$.
(c) At random select $R' \xleftarrow{r} \mathbb{G}_T$, let $r' = H_3(\Gamma, R')$.
(d) Compute $C'_1 = rk_6^{r'}$, $C'_2 = R' \cdot rk_5^{r'}$, $C'_3 = \Gamma \oplus F_2(R')$.
(e) Set $C'_0 = 1$, $C'_4 = \phi$, $C'_5 = \phi$.
(f) Output the tuple $[C'_0, C'_1, C'_2, C'_3, C'_4, C'_5]$.

Fig. 3. Obfuscator for re-encryption circuits for Π.

4.2 Analysis

Theorem 2 (*Preserving Functionality*): The proposed re-encryption obfuscator *Obf* satisfies the preserving functionality.

***Proof*:** For any circuit $C \in C_k$ and let $R_{sk_1,pk_2} \in Obf(C)$. For any possible input, the output distributions of C and R_{sk_1,pk_2} are statistically close, we consider the following three classes of inputs.

First, for any message $m \in \{0, 1\}^{\ell_2}$, observe that

$$\begin{aligned}&\mathrm{Enc}(pk_1, 0, m) = [C_0, C_1, C_2, C_3, C_4, C_5] \\ &= [0, g^r, R \cdot e(g^x, H_0(g^x))^r, m \oplus F_1(R), g_1^r, H_2(C_1, C_2, C_3, C_4)]\end{aligned}$$

For a randomly chosen, when such a ciphertext is fed as input, the circuit outputs

$$\begin{aligned}&[1, rk_6^{r'}, R' \cdot rk_5^{r'}, \Gamma \oplus F_2(R'), \phi, \phi] \\ &= [1, e(g^y, g)^{twr'}, R' \cdot e(g^y, g^y)^{twr'}, \Gamma \oplus F_2(R'), \phi, \phi]\end{aligned}$$

where s, t, w, $r' \xleftarrow{r} \mathbb{Z}_q$ randomly. The output is identically distributed to the output of $\mathrm{Enc}(pk_2, 1, m)$. Secondly, the same holds for all $m \in \{0, 1\}^{\ell_2}$. Finally, for **keys** and illegal inputs, the outputs are also identical.

Theorem 3 (*Anonymity*): Assume that SDBDH assumption holds, the obfuscated re-encryption scheme **Obf** achieves the anonymity for re-encryption keys in **Definition 4** in the random oracle model.

The proof of Theorem 2 is similar with [19] and we omit it here.

Theorem 4: (*Polynomial Slowdown*) The proposed obfuscator **Obf** has the performance of polynomial slowdown.

***Proof*:** This property is obviously. The obfuscated circuit computes a few bilinear maps and hash functions compared with the original unobfuscated algorithm.

Theorem 5 (*Virtual Black-box Security*): The proposed obfuscator satisfies the average-cage virtual black-box security.

***Proof*:** In order to hold the virtual black-box property, we consider an adversary who outputs the code of the obfuscated circuit $Obf(C)$. Thus, we must construct a simulator $Sim^C(1^l, aux)$ such that the distinguisher D^C takes as input an obfuscated circuit and auxiliary input aux, we require

$$|\Pr[D^C(Obf(C), aux)] = 1 - \Pr[D^C(Sim^C(1^l, aux), aux)] = 1| < negl(l)$$

The simulator $Sim^C(1^l, aux)$ definition as follows:

1. Query the oracle C on **keys** to obtain pk_1, pk_2.
2. Sample $rk_1^{r'}$, $rk_2^{r'} \xleftarrow{r} \mathbb{G}$.
3. As in Step (3) of the **Obf** algorithm, create and output a circuit R'_{sk_1,pk_2} using the values (pk_1, pk_2, rk'_1, rk'_2, rk_3, rk_4, rk_5, rk_6).

We devise two experiments Nice(D^C, aux) and Junk(D^C, aux) such that outputs of $D^C(Obf(C), aux)$ and $D^C(Sim^C(1^l, aux), aux)$ are identically distributed to Nice(D^C, aux) and Junk(D^C, aux), respectively.

Nice(D^C, aux)	**Junk**(D^C, aux)
$q, \mathbb{G}, H_0, H_1, H_2, H_3, H_4 \xleftarrow{r} BG(1^l)$	$q, \mathbb{G}, H_0, H_1, H_2, H_3, H_4 \xleftarrow{r} BG(1^l)$
$x, y \xleftarrow{r} \mathbb{G}$	$x, y \xleftarrow{r} \mathbb{G}$
$pk_1 \leftarrow g^x, pk_2 \leftarrow g^y$	$pk_1 \leftarrow g^x, pk_2 \leftarrow g^y$
$rk_1 \leftarrow H_0(g^x)^{-x} \cdot (g^y)^{H_4(s \cdot x)}$	$rk_1' \xleftarrow{r} \mathbb{G}$
$rk_2 \leftarrow g^{H_4(s \cdot x)} \cdot g_1^t$	$rk_2 \xleftarrow{r} \mathbb{G}$
$b \leftarrow D^C(pk_1, pk_2, rk_1, rk_2, rk_3, rk_4, rk_5, rk_6)$	$b \leftarrow D^C(pk_1, pk_2, rk_1', rk_2', rk_3, rk_4, rk_5, rk_6)$
Output b	Output b

We can prove that, in case the distinguisher D having oracle access to C_{sk_1,pk_2} for the keys sk_1 and pk_2 that are generated in the above experiments, there have the identical distribution, and thus the distinguisher D obtaining the knowledge of the obfuscated circuit can be simulated only knowledge of auxiliary input. This indicates that the obfuscator satisfies the average-case virtual black-box property.

We now show that the obfuscator **Obf** satisfies the average-case virtual black-box property. The proof techniques used here are similar to [2].

Lemma 1: Under the secure hash functions and SDBDH assumption, for all PPT distinguishers D and auxiliary input aux, the following two distributions are statistically close.

Proposition 1: Under the secure hash functions, $Nice^1_{l.aux} \approx Junk^1_{l.aux}$ where

Nice^1: Proceed as **Nice** except that the output is (pk_1, pk_2, rk_1, rk_2, rk_3, rk_4, rk_5, rk_6).
Junk^1: Proceed as **Junk** except that the output is (pk_1, pk_2, rk_1, rk_2, rk_3, rk_4, rk_5, rk_6).

If there exists a distinguisher D which distinguishes $Nice^1$ from $Junk^1$ with advantage ε, then there exists a distinguisher D' which solves the resistance of hash functions.

We now extend proposition 1 by providing the distinguisher with an oracle which returns a 6-tuple of random values. On input the tuple [0, C_1, C_2, C_3, C_4, C_5], at first check whether C_1, C_4, C_5 belong to $\mathbb{G}$, C_2 belong to $\mathbb{G}_T$ and C_3 belong to $\{0, 1\}^{\ell_2}$. If succeeds, then output [0, C'_1, C'_2, C'_3, ϕ, ϕ] where C'_1, C'_2 are chosen uniformly and independently from group $\mathbb{G}$, $C'_3 \leftarrow \{0, 1\}^{6\ell_1+\ell_2}$. Otherwise, output $\perp$. Intuitively, oracle $\mathcal{R}$ outputs only random values and thus should not help the distinguisher.

Proposition 2: Under the secure hash functions, $Nice^2_{l.aux} \approx Junk^2_{l.aux}$ where

Nice^2: Same as Nice($D^{\mathcal{R}}$, *aux*),
Junk^2: Same as Junk($D^{\mathcal{R}}$, *aux*).

***Proof*:** The oracle $\mathcal{R}$ can be perfectly simulated without any auxiliary information. Thus, for any $D^{\mathcal{R}}$, there exists another non-oracle distinguisher D', D' runs D internally by giving its own input to D. The output distribution of D' is identical to D. Applying Proposition 1, we thus have that for all distinguisher $D^{\mathcal{R}}$, $Nice^2_{l.aux} \approx Junk^2_{l.aux}$.

Proposition 3: For any p.p.t distinguisher D^C, let

$$\alpha(l, aux) = \mathrm{Avd}(Nice(D^C, aux), Junk(D^C, aux))$$
$$\beta(l, aux) = \mathrm{Avd}(Nice(D^{\mathcal{R}}, aux), Junk(D^{\mathcal{R}}, aux))$$

There exists a p.p.t algorithm $\mathcal{A}$ which distinguishes between the two distributions of the SDBDH problem with advantage at least $\frac{1}{2}|\alpha(l, aux) - \beta(l, aux)|$

***Proof*:** We let $\alpha = \alpha(l, aux)$, $\beta = \beta(l, aux)$. Input to $\mathcal{A}$ is a SDBDH instance $\Delta = (g, g^a, g^b, Q)$ and *aux*, and:

1. $\mathcal{A}$ samples a challenge bit $\mu \xleftarrow{r} \{0, 1\}$ to pick whether to run Nice or Junk.
2. $\mathcal{A}$ selects integers a, b, c, s, $w \xleftarrow{r} \mathbb{Z}_q$ and group elements rk'_1, rk'_2, rk'_3, rk'_4, rk'_5, rk'_6 $\xleftarrow{r} \mathbb{G}$.
3. $\mathcal{A}$ sets $pk_1 = g^a$, $pk_2 = g^{xb}$ then computes a valid re-encryption tuple (rk_1, rk_2, rk_3, rk_4, rk_5, rk_6) by

$$rk_1 \leftarrow H_0(g^a)^{-a} \cdot (g^{xb})^{H_4(s \cdot a)},$$
$$rk_2 \leftarrow g^{H_4(s \cdot a)} \cdot g^{yc},$$
$$rk_3 \leftarrow g^y,$$
$$rk_4 \leftarrow g^w,$$

$rk_5 \leftarrow Q^{b^2w}$,
$rk_6 \leftarrow e(g^{xb}, g^y)^w$.

4. If $\mu = 1$, then $\mathcal{A}$ runs $D^{\mathcal{O}}(pk_1, pk_2, rk_1, rk_2, rk_3, rk_4, rk_5, rk_6, aux)$ where $\mathcal{O}$ is defined below. If $\mu = 0$, then $\mathcal{A}$ runs $D^{\mathcal{O}}(pk_1, pk_2, rk'_1, rk'_2, rk'_3, rk'_4, rk'_5, rk'_6, aux)$.
   When D queries the oracle $\mathcal{O}$ on input $[0, C_1, C_2, C_3, C_4, C_5]$, $\mathcal{A}$ responds as follows:
   (a) Check the equation $e(g, C_5) = e(C_1, H_2(C_0, C_1, C_2, C_3, C_4)$. If the equation does not hold, it aborts.
   (b) Otherwise let $T_1 = C_1 = g^r$, $T_2 = C_2 \cdot e(C_1, rk_1) = R \cdot e(g^r, (g^y)^{H_4(s \cdot x)})$, $T_3 = C_3 = m \oplus F_1(R)$, and $T_4 = C_4 = g_1^4$, and set $\Gamma = T_1||T_2||T_3||T_4||rk_2||rk_3||rk_4$.
   (c) At random select $R' \xleftarrow{r} \mathbb{G}_T$, let $r' = H_3(\Gamma, R')$.
   (d) Compute $C'_1, = rk_6^{r'}$, $C'_2 = R' \cdot rk_2^{r'}$, $C'_3 = \Gamma \oplus F_2(R')$, and set $C'_0 = 1$, $C'_4 = \phi$, $C'_5 = \phi$.
   (e) Respond with the tuple $[1, C'_1, C'_2, C'_3, C'_4, C'_5]$.
5. Finally, D outputs $\mu' \in \{0, 1\}$. If $\mu = \mu'$, $\mathcal{A}$ outputs 1 (*i.e.* it confirms that $Q = e(g, g)^{x^2y}$). Else if $\mu \neq \mu'$, then $\mathcal{A}$ outputs 0 (*i.e.* it confirm that $Q \neq e(g, g)^{x^2y}$).

Claim: When $Q = e(g, g)^{x^2y}$ (*i.e.* Δ is an SDBDH instance), then $\Pr[\mathcal{A}(\Delta) = 1] = \frac{1}{2} + \alpha(l, aux)/2$.

***Proof of Claim*:** When $Q = e(g, g)^{x^2y}$, then $\mathcal{A}$ simulates Nice^C or Junk^C toward the algorithm D. Since the re-encryption tuple rk_1, rk_2, rk_3, rk_4, rk_5, rk_6 is a valid re-encryption tuple for $pk_1 \rightarrow pk_2$, the input parameters to D in Step 4 are identically distributed to the inputs to D in either experiment Nice or Junk, and the response to an oracle query on **keys** is also identically distributed. Note that the remains is to show that the responses $\mathcal{A}$ provides to oracle queries on the tuples $[0, C_1, C_2, C_3, C_4, C_5]$ are also identically distributed. Since $Q = e(g, g)^{x^2y}$ and rk_1, rk_2, rk_3, rk_4, rk_5, rk_6 are valid re-encryption tuple. Hence, the probability that $\mathcal{A}$ outputs 1 is $\Pr[\mathcal{A}(\Delta) = 2] = \frac{1}{2} + \alpha/2$.

Claim: If Q is randomly chosen value, then $\Pr[\mathcal{A}(\Delta) = 0] = \frac{1}{2} + \beta(l, aux)/2$.

***Proof*:** This proof is almost identical to the previous one. We must show that responses to the oracle return three randomly selected group elements. We denote by ω, χ, γ, μ, ν the values such that $C_1 = g^\omega$, $C_4 = g^\chi$, $C_5 = g^\gamma$, $C_2 = e(g^\mu, g^\nu)$, $C_3 \leftarrow \{0, 1\}^{\ell_2}$, and by e the value such $pk_2 = g^e$. Finally the elements returned by the oracle are

$$C'_1 = e(g^{xb}, g^y)^{wr'}, \; C'_2 = R' \cdot Q^{b^2wr'},$$
$$C'_3 = \Gamma \oplus F_2(R') = g^\omega || e(g^\mu, g^\nu) \cdot e(g^\omega, H_0(g^a)^{-a} \cdot (g^{xb})^{H_4(s \cdot a)})$$
$$||C_3||g^\chi||g^{H_4(s \cdot a)} \cdot g^{vc}||g^y||g^w$$

Since r', c, s, w are chosen uniformly at random from $\mathbb{Z}_q$, then C'_1, C'_2, C'_3 will also be independent on every invocation of the oracle. This is because C'_1, C'_2, C'_3 can be viewed as a generator of a group or as an element of group which has been mutiplied by a random group element. Hence, the probability that $\mathcal{A}$ outputs 0 is $\Pr[\mathcal{A}(\Delta) = 0] = \frac{1}{2} + \beta/2$.

Thus, $\mathcal{A}$ distinguishes the two distributions of the SDBDH problem with advantage at least $\frac{1}{2}|\alpha(l, aux) - \beta(l, aux)|$.

***Proof of Lemma* 1:** We know from proposition 2 that $\beta(l, aux)$ is negligible. By the SDBDH problem and Proposition 3, we can infer that $|\alpha(l, aux)-\beta(l, aux)|$ is negligible, and therefore, $\alpha(l, aux)$ is also negligible. Hence, the obfuscator **Obf** satisfies the average case secure virtual black box property and this concludes the proof of the Lemma.

5. CONCLUSION

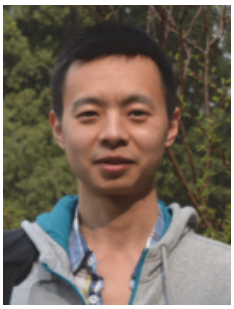
We designed a program obfuscator to implement the secure re-encryption functionality in the presence of possible secret revealing. We used two types of ciphertexts to ensure that it could not be multiple-hop and bidirectional re-encryption when the obfuscator is run on the (untrusted) server. The obfuscator algorithm achieves the single-hop and unidirectional property and obtains the function property of anonymous notion in virtual black-box security. The obfuscator for re-encryption algorithm can be executed on an untrusted server, without revealing the sensitive information such as the user's secret key and the plaintext of the message during the perform.

REFERENCES

1. B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang, "On the (im)possibility of obfuscating programs," *Journal of the ACM*, Vol. 59, 2012, pp. 1-48.
2. S. Hohenberger, G. N. Rothblum, A. Shelat, and V. Vaikuntanathan, "Securely obfuscating re-encryption," *Theory of Cryptography*, LNCS, 2007, pp. 233-252.
3. V. Balachandran and S. Emmanuel, "Software protection with obfuscation and encryption," *Information Security Practice and Experience*, LNCS, 2013, pp. 309-320.
4. J. Shao, P. Liu, and Y. Zhou, "Achieving key privacy without losing CCA security in proxy re-encryption," *Journal of Systems and Software*, Vol. 85, 2012, pp. 655-665.
5. R. Canetti, S. Goldwasser, and O. Poburinnaya, "Adaptively secure two-party computation from indistinguishability obfuscation," *Theory of Cryptography*, LNCS, 2015, pp. 557-585.
6. A. Sahai and B. Waters, "How to use indistinguishability obfuscation," in *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, 2014, pp. 475-484.
7. J. Shen, H. Tan, J. Wang, and S. Lee, "A novel routing protocol providing good Transmission reliability in underwater sensor networks," *Journal of Internet Technology*, Vol. 16, 2015, pp. 171-178.
8. A. Srinivasan and C. P. Rangan, "Efficiently obfuscating re-encryption program under DDH assumption," *Cryptology ePrint Archive*, Report 2015/822, 2015.
9. Z. Xia, X. Wang, L. Zhang, Z. Qin, X. Sun, and K. Ren, "A privacy-preserving and copy-deterrence content-based image retrieval scheme in cloud computing," *IEEE Transactions on Information Forensics and Security*, DOI:10.1109/TIFS.2016.2590944, 2016.

10. M. Zhang, B. Chen, and H. Shen, "Program obfuscator for privacy-carrying unidirectional one-hop re-encryption," *Algorithms and Architectures for Parallel Processing*, LNCS, 2015, pp. 133-142.
11. J. Zimmerman, "How to obfuscate programs directly," *Advances in Cryptology – EUROCRYPT*, LNCS, 2015, pp. 439-467.
12. R. Cheng and F. Zhang, "Secure obfuscation of conditional re-encryption with keyword search," in *Proceedings of the 5th International Conference on Intelligent Networking and Collaborative Systems*, 2013, pp. 30-37.
13. Z. Fu, X. Wu, C. Guan, X. Sun, and K. Ren, "Towards efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement," *IEEE Transactions on Information Forensics and Security*, DOI: 10.1109/TIFS.2016.2596138, 2016.
14. S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters, "Candidate indistinguishability obfuscation and functional encryption for all circuits," in *Proceedings of the 54th IEEE Annual Symposium on Foundations of Computer Science*, 2013, pp. 40-49.
15. S. Goldwasser and G. N. Rothblum, "On best-possible obfuscation," *Theory of Cryptography*, LNCS, 2007, pp. 194-213.
16. S. Hada, "Zero-knowledge and code obfuscation," *Advances in Cryptology – ASIACRYPT*, LNCS, 2000, pp. 443-457.
17. M. Zhang, T. Nishide, B. Yang, and T. Takagi, "Anonymous encryption with partial-order subset delegation functionality," *Provable Security*, LNCS, 2011, pp. 154-169.
18. M. Zhang and T. Takagi, "Efficient constructions of anonymous multireceiver encryption protocol and their deployment in group e-mail systems with privacy preservation," *IEEE Systems Journal*, Vol. 7, 2013, pp. 410-419.
19. Q. Zheng, W. Zhu, J. Zhu, and X. Zhang, "Improved anonymous proxy re-encryption with CCA security," in *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security*, 2014, pp. 249-258.
20. S. Hada, "Secure obfuscation for encrypted signatures," *Advances in Cryptology – EUROCRYPT*, LNCS, 2010, pp. 92-112.
21. G. Ateniese, K. Benson, and S. Hohenberger, "Key-private proxy re-encryption," *Topics in Cryptology – CT-RSA*, LNCS, pp. 279-294.
22. B. Libert and D. Vergnaud, "Unidirectional chosen-ciphertext secure proxy re-encryption," *Public Key Cryptography*, LNCS, 2008, pp. 360-379.
23. J. Shao, P. Liu, G. Wei, and Y. Ling, "Anonymous proxy re-encryption," *Security and Communication Networks*, Vol. 5, 2011, pp. 439-449.
24. R. Canetti and S. Hohenberger, "Chosen-ciphertext secure proxy re-encryption," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, 2007, pp. 185-194.

Mingwu Zhang is a Professor at School of Computer Sciences, Hubei University of Technology. From August 2010 to August 2012, Dr. Zhang has been a JSPS Postdoctoral Fellow of Japan Society of Promotion Sciences at Institute of Mathematics for Industry in Kyushu University, Japan, and From June 2015 to June 2016, he has been a senior visiting scholar at University of Wollongong, Australia. His research interests include cryptography technology for network and data security, secure computation, and privacy preservation, *etc.*



Yudi Zhang is a master student at School of Computer Sciences, Hubei University of Technology. His research interest focuses on applied cryptography and program obfuscation techniques.

Hua Shen is an Associate Professor at School of Computer Sciences, Hubei University of Technology. Her research focuses on security protocol designs and applications in open networks.

Chunming Tang is a Professor at School of Mathematics and Information Sciences, Guangzhou University. His research interests include cryptography and applications, security in cloud computing *etc.*

Lein Harn received the B.Sc. degree in Electrical Engineering from National Taiwan University in 1977, the M.S. degree in Electrical Engineering from the State University of New York, Stony Brook, in 1980, and the Ph.D. degree in Electrical Engineering from the University of Minnesota in 1984. He joined the Department of Electrical and Computer Engineering, University of Missouri-Columbia, as an Assistant Professor in 1984. In 1986, he moved to the Computer Science and Telecommunication Program at the University of Missouri-Kansas City (UMKC). His research interests are cryptography, network security, and wireless communication security. In 2015, he was appointed as a Chu-Tian Researcher with the School of Computer Science and Technology, Hubei University of Technology, China.