Comments on Shao-Cao's Unidirectional Proxy Re-Encryption Scheme from PKC 2009

Xi Zhang, Min-Rong Chen, Xia Li

Abstract

In Eurocrypt'98, Blaze, Bleumer and Strauss [4] introduced a primitive named proxy re-encryption (PRE), in which a semi-trusted proxy can convert - without seeing the plaintext - a ciphertext originally intended for Alice into an encryption of the same message intended for Bob. PRE systems can be categorized into *bidirectional* PRE, in which the proxy can transform from Alice to Bob and vice versa, and *unidirectional* PRE, in which the proxy cannot transform ciphertexts in the opposite direction. How to construct a PRE scheme secure against chosen-ciphertext attack (CCA) without pairings is left as an open problem in ACM CCS'07 by Canetti and Hohenberger [7]. In CANS'08, Deng *et al.* [8] successfully proposed a CCA-secure bidirectional PRE scheme without pairings. In PKC'09, Shao and Cao [10] proposed a unidirectional PRE without pairings, and claimed that their scheme is CCA-secure. They compared their scheme with Libert-Vergnaud's pairing-based unidirectional PRE scheme from PKC'08, and wanted to indicate that their scheme gains advantages over Libert-Vergnaud's scheme. However, Weng *et al.* [13] recently pointed out that Shao-Cao's scheme is not CCA-secure by giving a concrete chosen-ciphertext attack, and they also presented a more efficient CCA-secure unidirectional PRE scheme without parings. In this paper, we further point out that, Shao-Cao's comparison between their scheme and Libert-Vergnaud's scheme is unfair, since Shao-Cao's scheme is even *not* secure against chosen-plaintext attack (CPA) in Libert-Vergnaud's security model.

Keywords. Proxy re-encryption, chosen-ciphertext attack, chosen-plaintext attack.

1 Introduction

The concept of proxy re-encryption (PRE) is initially introduced by Blaze, Bleumer and Strauss [4] in Eurocrypt'98. In a PRE system, a proxy is able to transform a ciphertext originally intended for Alice into an encryption of the same message intended for Bob. The proxy, however, cannot learn anything about the messages encrypted under either key. According to the direction of transformation, PRE can be categorized into *bidirectional* PRE and *unidirectional* PRE. In bidirectional PRE, the proxy can transform from Alice to Bob and vice versa. In contrast, the proxy in unidirectional PRE cannot transform ciphertexts in the opposite direction. According to another criterion, PRE systems can also be classified into *multi-hop* PRE, in which the ciphertexts can be transformed from Alice to Bob and then to Charlie and so on, and *single-hop* PRE, in which the ciphertexts can only be transformed once[1].

Blaze *et al.* [4] proposed the first bidirectional PRE scheme in 1998. In 2005, Ateniese *et al.* [2,3] presented unidirectional PRE schemes based on bilinear pairings. All of these schemes are only secure against chosen-plaintext attack (CPA). However, applications often require security against chosen-ciphertext attacks (CCA). So, in ACM CCS'07, Canetti and Hohenberger [7] presented the first CCA-secure bidirectional PRE scheme. In PKC'08 Libert and Vergnaud [9] presented a (single-hop) unidirectional PRE scheme with replayable chosen-ciphertext (RCCA) security [13], where a harmless mauling of the challenge ciphertext is tolerated.

[1] In [2, 3, 9], for a single-hop unidirectional PRE scheme, the original ciphertext is named *second level ciphertext*, and the transformed ciphertext is called *first level ciphertext*. Through out this paper, we also use these notations.

The CCA/RCCA-secure PRE schemes in [7, 9] rely on bilinear pairings. It is well-known that, compared with standard operations such as modular exponentiation in finite fields, the pairing computation is a costly expensive operation. It would be desirable for cryptosystems to be constructed without bilinear pairings, especially in computation resource limited settings. In view of this, Canetti and Hohenberger [7] left an important open problem in ACM CCS'07, i.e., how to construct a CCA-secure PRE scheme without pairings.

In CANS'08, Deng *et al.* [8] proposed the first CCA-secure (single-hop) bidirectional PRE scheme without pairings. Subsequently, in PKC'09 Shao and Cao [10] proposed a single-hop unidirectional PRE without pairings, and claimed that their scheme is CCA-secure. They compared their scheme with Libert-Vergnaud's unidirectional PRE scheme in [9], and wanted to indicate that their scheme gains advantages over Libert-Vergnaud's scheme. However, Weng *et al.* [13] recently pointed out that Shao-Cao's scheme is not CCA-secure by giving a concrete chosen-ciphertext attack. Weng *et al.* [13] also presented a more efficient CCA-secure unidirectional PRE scheme without parings.

In this paper, we further point out that, Shao-Cao's comparison between their scheme and Libert-Vergnaud's scheme is unfair, since Shao-Cao's scheme is even *not* secure against chosen-plaintext attack (CPA) in Libert-Vergnaud's security model.

Organization. The rest of the paper is organized as follows. In Section 2, we review the definition and security notions for PRE systems as defined by Libert and Vergnaud [9]. In Section 3, we review Shao-Cao's unidirectional PRE scheme, and then point out that Shao-Cao's scheme is even not CPA-secure under Libert-Vergnaud security model [9]. Finally, Section 4 concludes the paper.

2 Framework of Unidirectional Proxy Re-Encryption

2.1 Definition of Unidirectional Proxy Re-Encryption

In this subsection, we review the definition of (single-hop) unidirectional PRE as defined in [9], with slight modifications for an easy description of Shao-Cao's unidirectional PRE scheme to be reviewed later.

A (single hop) unidirectional PRE scheme is defined by the following six algorithms:

- $\texttt{Global-Setup}(\lambda)$: On input of a security parameter λ, this setup algorithm outputs the global parameters *param* to be used by all parties in the scheme. To lighten notations, we do not explicitly write *param* as the input for the rest of algorithms.
- $\texttt{KeyGen}(\lambda)$: On input of the security parameter λ, all parties use this randomize algorithm to generate a public/private key pair (pk, sk).
- $\texttt{ReKeyGen}(sk_i, pk_j)$: On input of user i's private key sk_i and user j's public key pk_j, this algorithm outputs a re-encryption key $rk_{i\rightarrow j}$, which can be used to translate second level ciphertexts for i into first level ciphertexts for j.
- $\texttt{Enc}(pk, m)$: On input of a receiver's public key pk and a plaintext m, this algorithm outputs a first level ciphertext that can be re-encrypted into a first level one intended for another party.
- $\texttt{ReEnc}(rk_{i\rightarrow j}, \mathsf{CT}_i)$: On input of a re-encryption key $rk_{i\rightarrow j}$ and a second level ciphertext CT_i under public key pk_i, this algorithm outputs a first level ciphertext CT_j intended for user j.
- $\texttt{Dec}(sk, \mathsf{CT})$: On input a private key sk and a ciphertext CT, this algorithm outputs a plaintext m or the error symbol $\perp$ if CT is invalid.

Correctness requires that, for any plaintext m chosen from the plaintext space, and any couple of private/public key pairs (sk_i, pk_i) and (sk_j, pk_j) generated by $\texttt{KeyGen}$, the following conditions

should hold:

$$\texttt{Dec}(sk_i, \texttt{Enc}(pk_i, m)) = m,$$
$$\texttt{Dec}(sk_j, \texttt{ReEnc}(\texttt{ReKeyGen}(sk_i, pk_j), \texttt{Enc}(pk_i, m))) = m.$$

2.2 Security Notions for Unidirectional Proxy Re-Encryption

In this subsection, we review the security notions for single-hop unidirectional PREs as defined in [9],with slight modifications for the sake of an easy explanation the attack against Shao-Cao's unidirectional PRE scheme. We remark that the security notions given in this subsection are in fact equivalent to those defined by Libert and Vergnaud [9].

The definition defined in [9] requires the adversary commit ahead of time which public key she wants to challenge. In addition, it only considers the replayable chosen-ciphertext security, where the attacker is notably disallowed to ask for a decryption of a re-randomized version of the challenge ciphertext.

Security of Second level ciphertexts. In this notion, the challenger provides the adversary with a second level ciphertext. The RCCA-security at the second level for a unidirectional PRE scheme Π can be defined via the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:

Setup. $\mathcal{A}$ first commits that he wants to challenge the target user i^*'s public key. Then $\mathcal{C}$ runs algorithm $\mathsf{GlobalSetup}$ to generate the public parameters $param$, and returns $param$ to $\mathcal{A}$.

Phase 1. $\mathcal{A}$ issues queries $q_1, \cdots, q_m$ where query q_i is one of the following:

- *Uncorrupted key generation oracle* $\mathcal{O}_u(i)$: $\mathcal{C}$ first runs algorithm KeyGen to obtain a public/private key pair (pk_i, sk_i), and then sends pk_i to $\mathcal{A}$. Here it is required that pk_{i^*} is generated by this oracle.
- *Corrupted key generation oracle* $\mathcal{O}_u(j)$: $\mathcal{C}$ first runs algorithm KeyGen to obtain a public/private key pair (pk_j, sk_j), and then gives (pk_j, sk_j) to $\mathcal{A}$.
- *Re-encryption key generation oracle* $\mathcal{O}_{rk}(pk_i, pk_j)$: $\mathcal{C}$ first runs $\texttt{ReKeyGen}(sk_i, pk_j)$ to generate a re-encryption key $rk_{i\to j}$, where sk_i is the private key with respect to pk_i. Then $\mathcal{C}$ returns $rk_{i\to j}$ to $\mathcal{A}$. Here it is required that $\mathcal{A}$ cannot issue the query $\mathcal{O}_{rk}(pk_{i^*}, pk_j)$ if pk_j is generated by $\mathcal{O}_c$.
- *Re-encryption oracle* $\mathcal{O}_{re}(pk_i, pk_j, \mathsf{CT}_i)$: $\mathcal{C}$ first runs algorithm $\texttt{ReKeyGen}$ to generate the re-encryption key $rk_{i\to j}$. Then it runs $\texttt{ReEnc}(rk_{i\to j}, \mathsf{CT}_i)$ to obtain a first level ciphertext CT_j, which is returned to $\mathcal{A}$.
- *Decryption oracle* $\mathcal{O}_d(pk, \mathsf{CT})$: Challenger $\mathcal{C}$ returns the result of $\mathsf{Decrypt}(sk, \mathsf{CT})$ to $\mathcal{A}$, where sk is the private key with respect to pk.

For the last three oracles, it is required that pk_i, pk_j and pk are all generated beforehand by $\mathcal{O}_u$ or $\mathcal{O}_c$.

Challenge. Once $\mathcal{A}$ decides that Phase 1 is over, it outputs two equal-length plaintexts m_0 and m_1 chosen from the plaintext space. Challenger $\mathcal{C}$ flips a random coin $\delta \in \{0, 1\}$, and encrypts m_δ under public key pk_{i^*} to generate a *second* level ciphertext CT^*. $\mathcal{C}$ then returns CT^* as the challenge ciphertext to $\mathcal{A}$.

Phase 2. $\mathcal{A}$ issues additional queries $q_{m+1}, \cdots, q_{max}$ where each of the queries is one of the following:

- *Public key generation query* $\mathcal{O}_u(i)$: The same as in Phase 1.
- *Secret key query* $\mathcal{O}_c(j)$: The same as in Phase 1.
- *Re-encryption key generation query* $\mathcal{O}_{rk}(pk_i, pk_j)$: The same as in Phase 1.

- *Re-encryption oracle* $\mathcal{O}_{re}(pk_i, pk_j, \mathsf{CT}_i)$: Challenger $\mathcal{C}$ responds as in Phase 1. Here it is required that, if pk_j is generated by $\mathcal{O}_c$, then $(pk_i, \mathsf{CT}_i) \neq (pk_{i^*}, \mathsf{CT}^*)$.
- *Decryption oracle* $\mathcal{O}_d(pk, \mathsf{CT})$: Challenger $\mathcal{C}$ responds as in Phase 1. Here it is required that, (pk, CT) can not be a derivative of $(pk_{i^*}, \mathsf{CT}^*)$. Here *derivative* of $(pk_{i^*}, \mathsf{CT}^*)$ is defined as follows:
  - $(pk_{i^*}, \mathsf{CT}^*)$ is a derivative of itself;
  - If pk is generated by $\mathcal{O}_u$ and $\mathtt{Dec}(sk, C) \in \{m_0, m_1\}$, then (pk, CT) is a derivative of $(pk_{i^*}, \mathsf{CT}^*)$.

Guess. Finally, $\mathcal{A}$ outputs a guess $\delta' \in \{0, 1\}$.

We define $\mathcal{A}$'s *advantage* in attacking scheme Π's RCCA-security at the second level as $\left|\Pr[\delta' = \delta] - \frac{1}{2}\right|$, where the probability is taken over the random coins consumed by the challenger and the adversary. If for any probabilistic polynomial time (PPT) adversary $\mathcal{A}$ defined as above, his advantage is negligible, then we say that the unidirectional PRE scheme is RCCA-secure at the second level.

Security of First Level Ciphertext. The above definition provides the adversary with a *second* level ciphertext in the challenge phase. Libert and Vergnaud [9] also defined a complementary definition of security by providing the adversary with a *first* level ciphertext in the challenge phase. Note that, in a single hop unidirectional PRE scheme, since the first level ciphertext cannot be further re-encrypted, $\mathcal{A}$ is allowed to obtain *any* re-encryption keys. Furthermore, given these re-encryption keys, $\mathcal{A}$ can re-encrypt ciphertexts by himself, and hence there is no need to provide the re-encryption oracle $\mathcal{O}_{re}$ for him. Concretely, the RCCA-security at the first level for a single-hop unidirectional PRE scheme Π can be defined via the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:

Setup. $\mathcal{A}$ first commits that he wants to challenge the target user i^*'s public key. Then $\mathcal{C}$ runs algorithm $\mathsf{GlobalSetup}$ to generate the public parameters $param$, and returns $param$ to $\mathcal{A}$.

Phase 1. $\mathcal{A}$ issues queries $q_1, \cdots, q_m$ where query q_i is one of the following:

- *Uncorrupted key generation oracle* $\mathcal{O}_u(i)$: $\mathcal{C}$ first runs algorithm KeyGen to obtain a public/private key pair (pk_i, sk_i), and then sends pk_i to $\mathcal{A}$. Here it is required that pk_{i^*} is generated by this oracle.
- *Corrupted key generation oracle* $\mathcal{O}_u(j)$: $\mathcal{C}$ first runs algorithm KeyGen to obtain a public/private key pair (pk_j, sk_j), and then gives (pk_j, sk_j) to $\mathcal{A}$.
- *Re-encryption key generation oracle* $\mathcal{O}_{rk}(pk_i, pk_j)$: $\mathcal{C}$ first runs $\mathtt{ReKeyGen}(sk_i, pk_j)$ to generate a re-encryption key $rk_{i \to j}$, where sk_i is the private key with respect to pk_i. Then $\mathcal{C}$ returns $rk_{i \to j}$ to $\mathcal{A}$. Note that, $\mathcal{A}$ is allowed to query *any* re-encryption key generation queries, even including $\mathcal{O}_{rk}(pk_{i^*}, pk_j)$ when pk_j is generated by $\mathcal{O}_c$.
- *Decryption oracle* $\mathcal{O}_d(pk, \mathsf{CT})$: Challenger $\mathcal{C}$ returns the result of $\mathsf{Decrypt}(sk, \mathsf{CT})$ to $\mathcal{A}$, where sk is the private key with respect to pk.

For the last three oracles, it is required that pk_i, pk_j and pk are all generated beforehand by $\mathcal{O}_u$ or $\mathcal{O}_c$.

Challenge. Once $\mathcal{A}$ decides that Phase 1 is over, it outputs two equal-length plaintexts m_0 and m_1 chosen from the plaintext space. Challenger $\mathcal{C}$ flips a random coin $\delta \in \{0, 1\}$, and encrypts m_δ under public key pk_{i^*} to generate a *first* level ciphertext CT^*. $\mathcal{C}$ then returns CT^* as the challenge ciphertext to $\mathcal{A}$.

Phase 2. $\mathcal{A}$ issues additional queries $q_{m+1}, \cdots, q_{max}$ where each of the queries is one of the following:

- *Uncorrupted key generation oracle* $\mathcal{O}_u(i)$: The same as in Phase 1.
- *Corrupted key generation oracle* $\mathcal{O}_c(j)$: The same as in Phase 1.
- *Re-encryption key generation oracle* $\mathcal{O}_{rk}(pk_i, pk_j)$: The same as in Phase 1.
- *Decryption oracle* $\mathcal{O}_d(pk, \mathsf{CT})$: Challenger $\mathcal{C}$ responds as in Phase 1. Here it is required that, $\mathcal{A}$ cannot issue $\mathcal{O}_d(pk_{i^*}, \mathsf{CT}')$ such that $\mathtt{Dec}(sk_{i^*}, \mathsf{CT}') \in \{m_0, m_1\}$.

Guess. Finally, $\mathcal{A}$ outputs a guess $\delta' \in \{0, 1\}$.

We define $\mathcal{A}$'s *advantage* in attacking scheme Π's RCCA-security at the first level as $\left|\Pr[\delta' = \delta] - \frac{1}{2}\right|$, where the probability is taken over the random coins consumed by the challenger and the adversary. If for any PPT adversary $\mathcal{A}$ defined as above, his advantage is negligible, then we say that the single-hop unidirectional PRE scheme is RCCA-secure at the first level. As usual, we can define the chosen-plaintext security (CPA) at the first level for an unidirectional PRE scheme by not providing the decryption oracle for adversary $\mathcal{A}$.

Master Secret Security. In [2, 3, 9], another security notion, named master secret security, is defined for unidirectional PRE. This security notion captures the intuition that, even if the dishonest proxy colludes with the delegatee, it is still impossible for them to derive the delegator's private key in full. Formally, we say that a unidirectional PRE scheme has master secret security, if for any PPT adversary $\mathcal{A}$, the following probability is negligible in the security parameter λ

$$\Pr\left[\gamma = sk_{i^*} \;\middle|\; \begin{array}{c} (pk_{i^*}, sk_{i^*}) \leftarrow \mathtt{KeyGen}(\lambda), \{(pk_x, sk_x) \leftarrow \mathtt{KeyGen}(\lambda)\}, \\ \{rk_{i^*\to x} \leftarrow \mathtt{ReKeyGen}(sk_{i^*}, pk_x)\}, \{rk_{x\to i^*} \leftarrow \mathtt{ReKeyGen}(sk_x, pk_{i^*})\}, \\ \gamma \leftarrow \mathcal{A}(pk_{i^*}, \{(pk_x, sk_x)\}, \{rk_{i^*\to x}\}, \{rk_{x\to i^*}\}). \end{array}\right].$$

As argued in [9], for single-hop unidirectional PRE, the master secret security is implied by the RCCA-security at the first level.

3 Analysis of Shao-Cao's Unidirectional PRE Scheme

3.1 Review of Shao-Cao's Scheme

Before reviewing Shao-Cao's scheme, we review the following non-interactive zero-knowledge proof of knowledge, named signature of knowledge of equality of two discrete logarithms, which is used in Shao-Cao's scheme.

Signature of Knowledge [1, 6, 12]. Let $y_1, y_2, g, h \in \mathbb{G}$ where $\mathbb{G}$ is a cyclic group of quadratic residues modulo N^2 (N is a safe-prime modulus), and H be a hash function such that $H : \{0,1\}^* \to \{0,1\}^k$ (k is the security parameter). A pair (c, s), verifying $c = H(y_1 \| y_2 \| g \| h \| g^s y_1^c \| h^s y_2^c \| m)$ is a signature of knowledge of the discrete logarithm of both $y_1 = g^x$ w.r.t. base g and $y_2 = h^x$ w.r.t. base h, on a message $m \in \{0,1\}^*$.

Note that the party with the secret x is able to compute the signature, provided that $x = \log_g y_1 = \log_h y_2$, by choosing a random $t \in \{0, \cdots, 2^{|N^2|+k} - 1\}$, and then computing (c, s) as

$$c = H(y_1 \| y_2 \| g \| h \| g^t \| h^t \| m), \quad s = t - cx.$$

In [10], they denoted $\mathtt{SoK.Gen}(y_1, y_2, g, h, m)$ as the generation of the proof.

Now, we begin to review Shao-Cao's scheme [10] as below, only with minor notational differences.

- `Global-Setup`: Choose three hash functions H_1, H_2 and H_3 where $H_1 : \{0,1\} \to \{0,1\}^{k_1}$, $H_2 : \{0,1\} \to \{0,1\}^n$, and $H_3 : \{0,1\} \to \{0,1\}^{k_2}$. Here k_1 and k_2 are security parameters, and n is the bit-length of messages to be encrypted. The public parameters are $param = (H_1, H_2, H_3, n, k_1, k_2)$.

- `KeyGen`: First choose a safe prime modulus $N = pq$, where $p = 2p' + 1, q = 2q' + 1$, p, p', q, q' are primes. Next, choose three random numbers $\alpha \in \mathbb{Z}_{N^2}^*, a, b \in [1, pp'qq']$ and a hash function H such that $H : \{0,1\}^* \to \mathbb{Z}_{N^2}$. Furthermore, set $g_0 = \alpha^2 \bmod N^2$, $g_1 = g_0^a \bmod N^2$, and $g_2 = g_0^b \bmod N^2$. The public key for this user is $pk = (H, N, g_0, g_1, g_2)$, the "weak" secret key is $wsk = (a, b)$, and the long term secret key is $sk = (p, q, p', q')$.

  Note that either the long term secret key or the weak secret key can be used to decrypt ciphertexts, but both the long term secret key and the weak secret key are required to produce a re-encryption key. Note also that in the following description, the elements from the key of user X contain an additional subscript X, e.g. $pk_X = (H_X(\cdot), N_X, g_{X0}, g_{X1}, g_{X2})$.

- `ReKeyGen`: On input a public key $pk_Y = (H_Y, N_Y, g_{Y0}, g_{Y1}, g_{Y2})$, a "weak" secret key $wsk_X = a_X$, and a long term secret key $sk_X = (p_X, q_X, p'_X, q'_X)$, it outputs the re-encryption key $rk_{X\to Y} = (rk^{(1)}_{X\to Y}, rk^{(2)}_{X\to Y})$, where $rk^{(1)}_{X\to Y} = (\dot{A}, \dot{B}, \dot{C})$, as follows:

  1. Randomly pick $\dot{\sigma} \in \mathbb{Z}_{N_Y}$ and $\dot{\beta} \in \{0,1\}^{k_1}$.
  2. Compute $rk^{(2)}_{X\to Y} = a_X - \dot{\beta} \bmod (p_X q_X p'_X q'_X)$.
  3. Compute $r_{X\to Y} = H_Y(\dot{\sigma}\|\dot{\beta})$ and then

  $$\dot{A} = (g_{Y0})^{r_{X\to Y}} \bmod (N_Y)^2, \quad \dot{B} = (g_{Y2})^{r_{X\to Y}} \cdot (1 + \dot{\sigma} N_Y) \bmod (N_Y)^2, \quad \dot{C} = H_1(\dot{\sigma}) \oplus \dot{\beta}.$$

  Remark 1. When the proxy colludes with the delegatee (i.e., given the re-encryption key $rk_{X\to Y}$ and the delegatee's secret key), she can recover the delegator's "weak" secret key a_X, although she cannot compugte the long term secret key sk_X.

- `Enc`: On input a public key $pk = (H, N, g_0, g_1, g_2)$ and a message $m \in \{0,1\}^n$, the sender acts as follows:

  1. Randomly pick $\sigma \in \mathbb{Z}_N$ and compute $r = H(\sigma\|m)$.
  2. Compute $A = (g_0)^r \bmod N^2$, $B = (g_1)^r \cdot (1 + \sigma N) \bmod N^2$, $C = H_2(\sigma) \oplus m$, and $D = (g_2)^r \bmod N^2$.
  3. Run $(c, s) \leftarrow \mathtt{SoK.Gen}(A, D, g_0, g_2, (B, C))$, where the underlying hash function is H_3.
  4. Output the second level ciphertext $\mathsf{CT} = (A, B, C, D, c, s)$.

- `ReEnc`: On input a re-encryption key $rk_{X\to Y} = (rk^{(1)}_{X\to Y}, rk^{(2)}_{X\to Y})$ and a ciphertext $\mathsf{CT} = (A, B, C, D, c, s)$ under key $pk_X = (H_X, N_X, g_{X0}, g_{X1}, g_{X2})$, this algorithms first checks whether $c = H_3(A\|D\|g_{X0}\|g_{X2}\|(g_{X0})^s A^c\|(g_{X2})^s D^c\|(B\|C))$ holds. If not, return $\perp$; otherwise, compute $A' = A^{rk^{(2)}_{X\to Y}} = (g_{X0})^{r(a_X - \dot{\beta})}$, and output the first level ciphertext

$$\mathsf{CT}_Y = (pk_X, A, A', B, C, rk^{(1)}_{X\to Y}) = (pk_X, A, A', B, C, \dot{A}, \dot{B}, \dot{C}).$$

  Remark 2. Shao and Cao did not explicitly include the public key pk_X in the first level ciphertext, and their comparison with Libert-Vergnaud's scheme did not count this ciphertext overhead. However, the delegator's public key should indeed be included in the first level ciphertext. Otherwise, as shown in their decryption algorithm, without knowing the delegator's public key, the delegatee is unable to decrypt the first level ciphertexts. Taking into account this ciphertext overhead, $pk_X = (H_X, N_X, g_{X0}, g_{X1}, g_{X2})$, the comparison made by Shao and Cao is unfair, and their scheme should be more *inefficient* than Libert-Vergnaud's scheme.

- `Dec`: On input a private key and ciphertext CT, this algorithm acts according to cases:

  – If CT is a second level ciphertext in the form $\mathsf{CT} = (A, B, C, D, c, s)$, it works as follows:

1. Check whether $c = H_3(A\|D\|g_0\|g_2\|(g_0)^s A^c\|(g_2)^s D^c\|(B\|C))$ holds. If not, return $\perp$, else,
   * if the secret key is the "weak" secret key a, compute $\sigma = \frac{B/(A^a) - 1 \bmod N^2}{N}$.
   * if the secret key is the long term secret key (p, q, p', q'), compute

$$\sigma = \frac{(B/g_0^{w_1})^{2p'q'} - 1 \bmod N^2}{N} \cdot \pi \bmod N,$$

   where w_1 is computed as that in [5], and π is the inverse of $2p'q' \bmod N$.
2. Compute $m = C \oplus H_2(\sigma)$.
3. If $B = (g_1)^{H(\sigma\|m)} \cdot (1 + \sigma N) \bmod N^2$ holds, return m; else return $\perp$.

– If CT is a first level ciphertext in the form $\mathsf{CT} = (pk_X, A, A', B, C, \dot{A}, \dot{B}, \dot{C})$ re-encrypted from pk_X to pk_Y:

1. Compute $\dot{\sigma}$ according to the following situations:
   * if the secret key is the "weak" secret key b, compute $\dot{\sigma} = \frac{\dot{B}/(\dot{A}^b) - 1 \bmod N_Y^2}{N_Y}$.
   * if sk is the long term secret key (p, q, p', q'), compute

$$\dot{\sigma} = \frac{(\dot{B}/g_{Y0}^{w_1})^{2p'q'} - 1 \bmod N_Y^2}{N_Y} \cdot \pi \bmod N_Y,$$

   where w_1 is computed as that in [5], and π is the inverse of $2p'q' \bmod N_Y$.
2. Compute $\dot{\beta} = \dot{C} \oplus H_1(\dot{\sigma})$.
3. Check whether $\dot{B} = (g_{Y1})^{H_Y(\dot{\sigma}\|\dot{\beta})} \cdot (1 + \dot{\sigma} N_Y) \bmod N_Y^2$ holds. If not, return $\perp$. (Up to this point, only the decryptor's public key, $(H_Y, N_Y, g_{Y0}, g_{Y1}, g_{Y2})$, is used. Afterward, only the delegator's public key, $(H_X, N_X, g_{X0}, g_{X1}, g_{X2})$, will be used.)
4. Compute $\sigma = \frac{B/(A' \cdot A^{\dot{\beta}}) - 1 \bmod N_X^2}{N_X}$, and $m = C \oplus H_2(\sigma)$.
5. If $B = (g_{X1})^{H_X(\sigma\|m)} \cdot (1 + \sigma N_X) \bmod N_X^2$ holds, return m; else return $\perp$.

3.2 On the Security of Shao-Cao's Scheme at the First level

In this subsection, we will show that Shao-Cao's scheme fails to satisfy the first level cipertext security, even in the CPA sense. Below, we present an adversary $\mathcal{A}$ against Shao-Cao's scheme with *non-negligible* advantage. Adversary $\mathcal{A}$ works as follows:

1. In the **Setup** phase, $\mathcal{A}$ first commits that he wants to challenge the target user Y^*'s public key. Then she is given the public parameters $param = (H_1, H_2, H_3, n, k_1, k_2)$.

2. In Phase 1, $\mathcal{A}$ issues a uncorrupted key generation query $\mathcal{O}_u$ to obtain the target user's public key pk_{Y^*}.

3. In Challenge Phase, $\mathcal{A}$ returns two equal-length plaintexts m_0 and m_1 to the challenger. Then she is given a first level challenge ciphertext CT^* encrypted for m_δ under pk_{Y^*} re-encrypted from another public key pk_X, where δ is the random bit chosen by the challenger. Recall that the delegator's public key is included in the first level ciphertext. Suppose the challenger ciphertext $\mathsf{CT}^* = (pk_X, A, A', B, C, \dot{A}, \dot{B}, \dot{C})$ is as below:

$$pk_X = (H_X, N_X, g_{X0}, g_{X1}, g_{X2}),$$
$$A = (g_{X0})^r \bmod N_X^2, B = (g_{X1})^r \cdot (1 + \sigma N_X) \bmod N_X^2, C = H_2(\sigma) \oplus m_\delta, A' = (g_{X0})^{r(a_X - \dot{\beta})},$$
$$\dot{A} = (g_{Y^*0})^{r_{X \to Y^*}} \bmod (N_{Y^*})^2, \quad \dot{B} = (g_{Y^*2})^{r_{X \to Y^*}} \cdot (1 + \dot{\sigma} N_{Y^*}) \bmod (N_{Y^*})^2, \quad \dot{C} = H_1(\dot{\sigma}) \oplus \dot{\beta}.$$

   where $r = H(\sigma\|m_\delta)$, and δ is the random bit chosen by the challenger.

4. In Phase 2, $\mathcal{A}$ acts as follows:

(a) Issue an corrupted key generation query to obtain another user Z's public/private key pair (pk_Z, sk_Z). Here $Z \neq Y^*, X$.

(b) Issue a re-encryption key generation query $\mathcal{O}_{rk}(pk_X, pk_Z)$ to get the re-encryption key $rk_{X\to Z}$. Recall that according the game for the first level ciphertext security, $\mathcal{A}$ is allowed to issue *any* re-encryption key generation query.

(c) As mentioned in Remark 1, using the re-encryption key $rk_{X\to Z}$ and the delegatee's secret key sk_Z, adversary $\mathcal{A}$ is able to recover the delegator's "weak" secret key a_X. Now, using a_X, $\mathcal{A}$ is able decrypt the ciphertext (A, B, C) to obtain the underlying plaintext m_δ as below:

$$\sigma = \frac{B/(A^{a_X}) - 1 \bmod N_X^2}{N_X}, \quad m_\delta = C \oplus H_2(\sigma).$$

5. Finally, with m_δ, adversary $\mathcal{A}$ can certainly know the bit δ chosen by the challenger, and thus break the first level ciphertext security of Shao-Cao's scheme.

Remark 3. The above adversary does not issue any decryption queries. Thus it means that Shao-Cao's scheme *fails* to achieve the CPA-security at the first level, not to mention the RCCA-security.

Remark 4. The above adversary is considered in the non-adaptive corruption model. In the fully adaptive corruption model, this adversary can certainly succeed, since this model provides the adversary more powers.

Remark 5. We notice that, Weng *et al.* [13] uploaded (on 05-May-2009 09:05:55 UTC) their paper on the eprint website presenting a chosen-ciphertext attack against Shao-Cao's scheme and mentioning measures to fix. Subsequently, Shao and Cao [11] modified (on 07-May-2009 22:36:35 UTC) their scheme on the eprint website according to these measures without mentioning this. However, we remark that, the above adversary can also be applied to Shao-Cao's modified scheme. Thus their modified scheme is still *not* CPA-secure at the first level.

4 Conclusions

We pointed out that Shao-Cao's comparisons between their unidirectional PRE scheme and Libert-Vergnaud's scheme is unfair, since Shao-Cao's scheme is even not CPA-secure in Libert-Vergnaud's security model.

References

[1] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *CRYPTO*, pages 255–270, 2000.

[2] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage. In *NDSS*. The Internet Society, 2005.

[3] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur.*, 9(1):1–30, 2006.

[4] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible Protocols and Atomic Proxy Cryptography. In *EUROCRYPT*, pages 127–144, 1998.

[5] Emmanuel Bresson, Dario Catalano, and David Pointcheval. A Simple Public-Key Cryptosystem with a Double Trapdoor Decryption Mechanism and Its Applications. In Chi-Sung Laih, editor, *ASIACRYPT*, volume 2894 of *Lecture Notes in Computer Science*, pages 37–54. Springer, 2003.

[6] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups (extended abstract). In Burton S. Kaliski Jr., editor, *CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 410–424. Springer, 1997.

[7] Ran Canetti and Susan Hohenberger. Chosen-Siphertext Cecure Proxy Re-Encryption. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *ACM Conference on Computer and Communications Security*, pages 185–194. ACM, 2007.

[8] Robert H. Deng, Jian Weng, Shengli Liu, and Kefei Chen. Chosen-Ciphertext Secure Proxy Re-encryption without Pairings. In Matthew K. Franklin, Lucas Chi Kwong Hui, and Duncan S. Wong, editors, *CANS*, volume 5339 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2008.

[9] Benoît Libert and Damien Vergnaud. Unidirectional Chosen-Ciphertext Secure Proxy Re-encryption. In Ronald Cramer, editor, *Public Key Cryptography*, volume 4939 of *Lecture Notes in Computer Science*, pages 360–379. Springer, 2008.

[10] Jun Shao and Zhenfu Cao. CCA-Secure Proxy Re-encryption without Pairings. In Stanislaw Jarecki and Gene Tsudik, editors, *Public Key Cryptography*, volume 5443 of *Lecture Notes in Computer Science*, pages 357–376. Springer, 2009.

[11] Jun Shao and Zhenfu Cao. Cca-secure proxy re-encryption without pairings. Cryptology ePrint Archive, Report 2009/164, 2009. http://eprint.iacr.org/. Improved version of [10]. Version 2 and 3.

[12] Victor Shoup. Practical threshold signatures. In *EUROCRYPT*, pages 207–220, 2000.

[13] Jian Weng, Sherman S.M. Chow, Yanjiang Yang, and Robert H. Deng. Efficient unidirectional proxy re-encryption. Cryptology ePrint Archive, Report 2009/189, 2009. http://eprint.iacr.org.