Cryptanalysis of A Multi-use CCA-secure Proxy Re-encryption Scheme

Yi Cai¹ ² and Deyu Qi¹

¹School of Computer Science and Engineering, South China University of Technology, Guangzhou 510641, P. R. China
²School of Computer Engineering, Guangzhou College of South China University of Technology, Guangzhou 510800, P. R. China
caiyibbx@gmail.com

Abstract

Proxy re-encryption (PRE) can be classified as single-hop PRE and multi-use PRE according to the times which the ciphertext is transformed. Finding a unidirectional, multi-use, and chosen ciphertext attack (CCA) secure PRE is presented as an open problem by Canetti et al. Wang claimed to resolve this problem by proposing the first multi-use CCA-secure unidirectional PRE scheme. But Zhang et al. have shown their proposal is not CCA-secure by giving concrete attacks. In this paper, we propose an improvement for Wang-Multi-Use-PRE to resist Zhang’s attack. But we also found that different with single-hop PRE, multi-use PRE without randomize encrypt key in its re-encryption algorithm could be vulnerable to attack. According to this principle we find a new type attack to multi-use CCA-secure PRE named proxy bypass attack. Then we give concrete attacks on Wang-Multi-Use-PRE scheme. This attack is also effective for other multi-use scheme.

Keywords: Proxy re-encryption, multi-use, chosen ciphertext security

1. Introduction


Recently, Wang, et al., [12, 13] claimed to answer this problem by proposing the first multi-use CCA-secure unidirectional PRE scheme. But in INCoS 2012, Zhang, et al., [18-20] have shown their proposal is not CCA-secure in the corresponding security models. In this article, we argue that Zhang’s attack is easy to fix. But there exists another attack to Wang-Multi-Use-PRE in regard to its multi-use property. This attack is also universally effective for other multi-use scheme.

Proxy re-encryption has been coming up for eight years now [3]. In 2006 year Ateniese et al. [2] proposed the first unidirectional proxy re-encryption scheme, but it is only CPA-secure.
Since then a number of PRE schemes with different properties have been proposed in the literature (e.g., [1, 2, 4, 6, 8, 9, 14–17]). In ACNS2007, Green [7] introduced an Identity-Based Proxy Re-Encryption (IB- PRE) scheme and defined the multi-use property. A multi-use scheme permits the proxy (or proxies) to perform multiple re-encryptions on a single ciphertext, e.g., re-encrypts from A to B, then re-encrypts the result from B to C, etc. They brought out a notion of “encryption level”, as an implicit property of a ciphertext. After that, there have been some papers [5, 10, 12, 13,21] proposed multi-use PRE schemes.

In this article, we propose an improvement for Wang-Multi-Use-PRE to resist Zhang’s attack. And then we introduce an attack named proxy bypass attack. The attack is against the PRE schemes that are extended from single-hop to multi-use and without randomize encrypt key in their re-encrypt algorithms. It’s effective to Wang-Multi-Use-PRE and also effective for some multi-use schemes such as case scheme [5, 13].

The rest of this paper is organized as follows. We review the framework of Multi-Use-PRE and Wang-Multi-Use-PRE scheme in Section 2. In Section 3, we describe Zhang’s attack and propose an improvement for Wang-PRE scheme to resist Zhang’s attack. And in Section 4, we propose a new general attack named proxy bypass attack, then give a concrete attack on Wang-multi-use PRE scheme and discuss the flaw exist in other schemes. Finally we conclude this paper in Section 5.

2. Framework of Multi-Use-PRE Scheme

In this section, we review the definition and security notions for Multi-Use-PRE as formalized by Zhang [18].

2.1. Multi-Use-PRE Scheme Definition

A multi-use unidirectional proxy re-encryption scheme is a tuple of algorithms (Key-Gen, ReKeyGen, Encrypt, ReEncrypt, Decrypt):

- KeyGen(1^k)→(pk, sk): Taking as input a security parameter k, this algorithm returns a public key pk and a secret key sk.
- ReKeyGen(sk_i, pk_j)→r_{ki}→j: Given user i’s secret key sk_i and user j’s public key pk_j, this algorithm outputs a re-encryption key r_{ki}→j.
- Encrypt(pk, m)→C_{pk}^{(1)}: Given a public key pk and a message m ∈ {0, 1}^n, this algorithm outputs a first-level ciphertext C_{pk}^{(1)}.
- ReEncrypt(r_{ki}→j, C_{pk}^{(l)})→C_{pkj}^{(l+1)} (l ≥ 1): Given a re-encryption key r_{ki}→j, a l-level ciphertext C_{pk}^{(l)} under pk_i, outputs a (l + 1)-level ciphertext C_{pkj}^{(l+1)} under pk_j or ⊥.
- Decrypt(sk_i, C_{pki})→m: Given a l-level ciphertext C_{pki} under pk_i, outputs a message m or ⊥.

2.2. CCA-security of Multi-Use-PRE Scheme

Before defining the CCA-security of Multi-Use-PRE Scheme, we introduce a notion challenge derivative which is defined by [11]. The definition of challenge derivative is as follows:

1. (pk^*, C^*) is a challenge derivative of itself.
2. If (pk_i, C_{pki}^{(l)}) is a challenge derivative, A has issued the query O_{REEnc}(pk_i, pk_j, C_{pki}^{(l)}) to receive a new ciphertext C_{pkj}^{(l+1)}, then (pk_j, C_{pkj}^{(l+1)}) is a challenge derivative.
3. If \((pki, C_{pki}^{(l)})\) is a challenge derivative, \(A\) has issued the query \(O_{RE-EXT}(pk_i, pk_j)\) to obtain a re-encryption key \(rk_{i,j}\), and a \((l + 1)\)-level ciphertext \(C_{pkj}^{(l+1)} = \text{ReEncrypt}(rk_{i,j}, C_{pki}^{(l)})\) is a challenge derivative.

Now, we review the CCA-secure definition of multi-use unidirectional PRE scheme. Notice that the definition is in the static corruption model, where the adversary should decide the corrupted users before the game starts.

A multi-use unidirectional PRE scheme is CCA-secure if the advantage of any PPT adversary \(A\) in the following game played between a challenger \(C\) and \(A\) is negligible in the security parameter \(k\).

- Setup Phase: The challenger takes a security parameter \(k\) and generates the public parameters.
- Phase 1: \(A\) issues queries \(q_1, \ldots, q_m\) adaptively, where \(q_i\) is one of the following oracles queries:
  - \(O_{KeyGen}(l)\): \(C\) runs algorithm \(\text{KeyGen}(1^k)\) to generate a key pair \((pk_i, sk_i)\), outputs \(pk_i\).
  - \(O_{EXT}(pk_i)\): \(C\) outputs user \(i\)'s private key \(sk_i\) corresponding to public key \(pk\).
  - \(O_{RE-EXT}(pk_i, pk_j)\): \(C\) runs algorithm \(\text{ReKeyGen}(sk_i, pk_j) \rightarrow rk_{i,j}\) to generate a re-encryption key \(rk_{i,j}\), where \(sk_i\) is the secret key corresponding to \(pk\).
  - \(O_{RE-Ext}(pk_i, pk_j, C^{(l)})\): \(C\) returns \(C_{pkj}^{(l+1)} = \text{ReEncrypt}(rk_{i,j}, C_{pki}^{(l)})\).
  - \(O_{Dec}(pk_i, C_{pki}^{(l)})\): \(C\) returns \(\text{Decrypt}(sk_i, C_{pki}^{(l+1)})\), where \(sk_i\) is the secret key corresponding to \(pk\).
- Challenge: Once \(A\) decides that Phase 1 is over, it outputs two equal-length plaintexts \(m_0, m_1 \in M\) and a public key \(pk^*\) corresponding to an uncorrupted private key, on which it wishes to be challenged. Moreover, \(A\) is restricted to the choice of \(pk^*\) that trivial decryption is not possible using keys extracted in Phase 1, e.g., by using re-encryption keys to transform \(C_{pki}\) into \(C_{pki}\) for which \(A\) holds a decryption key. The challenger picks a random bit \(b \in \{0, 1\}\) and computes the challenge ciphertext \(C^* = \text{Encrypt}(m_b, \text{params, } pk^*)\). It sends \(C^*\) as the challenge to \(A\).
- Phase 2: \(A\) issues queries \(q_{m+1}, \ldots, q_n\) adaptively, where \(q_i\) is one of the following oracles queries:
  - \(O_{EXT}(pk_i)\): \(C\) responds as in Phase 1 except a challenge derivative \((pk_b, C_{pki}^{(l)})\) exists.
  - \(O_{RE-EXT}(pk_a, pk_j)\): \(C\) responds as in Phase 1 except a challenge derivative \((pk_a, C_{pki}^{(l)})\) exists and \(A\) has issued \(O_{EXT}(pk_j)\).
  - \(O_{RE-Ext}(pk_i, pk_j, C_{pki}^{(l)})\): \(C\) responds as in Phase 1 except a challenge derivative \((pk_j, C_{pki}^{(l)})\) exists and \(A\) has issued \(O_{EXT}(pk_j)\).
  - \(O_{Dec}(pk_b, C_{pki}^{(l)})\): If \((pk_b, C_{pki}^{(l)})\) is not a challenge derivative, \(C\) responds as in Phase 1.
  - Guess: Finally, \(A\) outputs a guess \(b' \in \{0, 1\}\). \(A\) wins if \(b' = b\).

We define adversary \(A\)'s advantage in attacking PRE as
\[
\text{Adv}_A = |Pr[b = b'] - 1/2|
\]

### 2.3 Reviews of Wang-Multi-Use-PRE scheme

Here we review a PRE scheme proposed by Wang et al. in CCS’09 poster session. Let \(1^k\) be the security parameter, \((q, g, G_1, G_T, e)\) be generated by a bilinear group generator with input \(1^k\), and \(\text{Sig} = (G, \mathcal{S}, V)\) be a strongly unforgeable signature scheme. Let \(g_1, h_1, h_2,\) and \(h_3\) be four random elements in \(G_1\). Furthermore, let \(H_1: \{0, 1\}^* \rightarrow Z_q^*\) and \(H_2: G_T \rightarrow G_1\) be two one-way, collision-resistant cryptographic hash functions. The public parameters are: par
= (q, g, g₁, h₁, h₂, h₃, G₁, G₂, e, Sig, H₁, H₂). Wang-Multi-Use-PRE scheme consists of the following five algorithms (KeyGen, Enc, RKGen, ReEnc, Dec):

- KeyGen(1ⁿ) → (pk, sk): Selects x ∈ ℤₚ*, set pk = gˣ, and sk = x.
- Enc(pk, m) → C(pk)[i]: Given m ∈ G₂, selects r ∈ ℤₚ*, outputs a first-level ciphertext
  \[ C(pk)[i] = (c₁[1], c₁[2], c₁[3]) = (gˣ, m • e(g, pk)ᵢ), (h₁, h₂ R⁻¹₁(8rᵢ[s]), h₃) x ḥ⁻¹ₑ(i). \]
- ReKeyGen(skᵢ, pkᵢ) → rkᵢ→j (i ≠ j): To generate a re-encryption key from pkᵢ to pkⱼ for pkᵢ’s proxy Pᵢ, do the following:
  1. Select rᵢj ∈ ℤₚ*, Kᵢj ∈ ℤₚ, Tᵢj ∈ ℤₚ.
     \[ (h₁ ḥˡᵢ(8rᵢ[s]), h₂ ḥˡᵢ(8rᵢ[s]) Rᵢ[j], h₃) x ḥᵢ j, Rᵢ[j] = Hᵢ(Kᵢj) • gᵢ x ḥᵢ j. \] where svkᵢ is a publicly available verification key of pkᵢ’s proxy Pᵢ.
  3. Output \[ rkᵢ→j = (Rᵢ[j], Rᵢ[j], Rᵢ[j], Rᵢ[j], Rᵢ[j]). \] The re-encryption key is sent to Pᵢ via a secure channel.
- ReEncrpt(rkᵢ→j, C(pk)[i]) → C(pk)[i+l] (i ≠ j, l ≥ 1):
  1. To re-encrypt a First-level ciphertext denoted by \[ C(pk)[i], \] do:
     (a) Parse \[ C(pk)[i] \] as (cᵢ[1], cᵢ[2], cᵢ[3]), and \[ rkᵢ→j \] as \[ (Rᵢ[j], Rᵢ[j], Rᵢ[j], Rᵢ[j], Rᵢ[j]). \]
     (b) Check if \[ e(gᵢ, cᵢ[3]) = e(cᵢ[1], h₁ ḥᵢ j cᵢ[1], h₂ ḥᵢ j cᵢ[1], h₃) \] and \[ e(gᵢ, Rᵢ[j]) = e(Rᵢ[j]), \]
        \[ (h₁ ḥᵢ j cᵢ[1], h₂ ḥᵢ j cᵢ[1], h₃) x ḥᵢ j. \] If either of them fails, return \( ⊥ \).
     (c) Compute \[ C = (cᵢ'[1], cᵢ'[2], cᵢ'[3], cᵢ'[4], cᵢ'[5], cᵢ'[6]), \] where \[ cᵢ'[1] = cᵢ[1], cᵢ'[2] = cᵢ[1] • e(cᵢ[j], Rᵢ[j]), \]
     (d) Let \[ Pᵢ \] be pkᵢ’s proxy, and \[ svkᵢ \] be the signing key of \[ Pᵢ \] corresponding to \[ Pᵢ \]’s verification key \[ Rᵢ[j] \].
     (e) Run the signing algorithm \[ S(sskᵢ, C) \], to generate a signature on the ciphertext tuple \[ (cᵢ'[1], cᵢ'[2], cᵢ'[3], cᵢ'[4], cᵢ'[5], cᵢ'[6]) \], and denote the signature as \[ Sᵢ[j]. \]
     (f) Output the ciphertext \[ Cᵢ[l] = \langle C, Sᵢ[j] \rangle. \]
  2. To re-encrypt a \( l \)-level ciphertext \[ Cᵢ[l] \]:
     (a) Parse \[ Cᵢ[l] \] as \[ (cᵢ[1, 1, 1], cᵢ[1, 1, 2], cᵢ[1, 2, 3], cᵢ[1, 2, 4], cᵢ[1, 2, 5], cᵢ[1, 2, 6]) \], and \[ rkᵢ→j \] as \[ (Rᵢ[j], Rᵢ[j], Rᵢ[j], Rᵢ[j], Rᵢ[j], Rᵢ[j]). \]
     (b) Check if \[ e(gᵢ, cᵢ[1, 2]) = e(cᵢ[1, 1], h₁ ḥᵢ j cᵢ[1, 1], h₂ ḥᵢ j cᵢ[1, 1], h₃) \] and \[ e(gᵢ, Rᵢ[j]) = e(Rᵢ[j]), \]
        \[ (h₁ ḥᵢ j cᵢ[1, 1], h₂ ḥᵢ j cᵢ[1, 1], h₃) x ḥᵢ j. \] If either of them fails, return \( ⊥ \).
     (c) For \( ∀ k ∈ [2, l] \), check \[ V(c[k, 3], c[k, 5], c[k], c[k], c[k], c[k]) = 1. \] Whenever one of the m fails, return \( ⊥ \). Otherwise, do the following:
     (d) Compute \[ C = (cᵢ'[1, 1, 1], cᵢ'[1, 1, 2], cᵢ'[1, 2, 3], cᵢ'[1, 2, 4], cᵢ'[1, 2, 5], cᵢ'[1, 2, 6]), \] where \[ cᵢ'[1, 2, 2] = e(cᵢ[1, 2], Rᵢ[j]), cᵢ'[1, 1, 1] = Rᵢ[j], cᵢ'[1, 2, 4] = Rᵢ[j], cᵢ'[1, 1, 2] = Rᵢ[j], cᵢ'[1, 2, 5] = Rᵢ[j], cᵢ'[1, 2, 6] = Rᵢ[j], \] and all other elements remain unchanged.
     (e) Let \[ Pᵢ \] be \[ pkᵢ \]’s proxy, and \[ svkᵢ \] be the signing key of \[ Pᵢ \] corresponding to \[ Pᵢ \]’s verification key \[ Rᵢ[j] \].
     (f) Run the signing algorithm \[ S(sskᵢ, C) \] to generate a signature on the ciphertext tuple \[ (cᵢ'[1, 1, 1], cᵢ'[1, 1, 2], cᵢ'[1, 2, 3], cᵢ'[1, 2, 4], cᵢ'[1, 2, 5], cᵢ'[1, 2, 6]) \], and denote the signature as \[ Sᵢ[l]. \]
     (g) Output the ciphertext \[ Cᵢ[l+1] = \langle C, Sᵢ[l] \rangle. \]
• Decrypt$(sk_p, C_{pk}) \rightarrow m$: If $C_{pk}^{(i)}$ cannot be parsed as $(c_{1,1}, c_{1,2}, c_{1,3})$ for a first-level ciphertext, or $(c_{1,1}, \cdots, c_{l,1}, c_{l,2}, c_{l,3}, c_{l,4}, c_{l,5})$ for an $l^{th}$-level ciphertext $(l > 1)$, then return ⊥. Otherwise, continue the following process:

1. For a First-level ciphertext,
   (a) Verify that $e(g, c_{1,3}) = e(c_{1,1}, h^r_{i(c_{1,1})} h^{r_{i(c_{1,1})}}_{h_{2(c_{1,1})}} h_{3(c_{1,1})})$. If not, return ⊥.
   (b) Otherwise, compute $m \leftarrow c_{1,2}g^{e(c_{1,1}, g^{sk_p})}$.
   (c) Output $m$.
2. For an $l^{th}$-level $(l > 1)$ ciphertext $(l > 1)$,
   (a) Check if $e(g, c_{1,3}) = e(c_{1,1}, h^r_{i(c_{1,1})} h^{r_{i(c_{1,1})}}_{h_{2(c_{1,1})}} h_{3(c_{1,1})})$. If not, return ⊥. Otherwise, do the following:
   (b) For $\forall k \in \{2, l\}$, check $V(c_{k,3}, c_{k,5}, (c_{1,1}, \cdots, c_{k,3}, c_{k,4})) = 1$. Whenever one of them fails, return ⊥. Otherwise, do the following:
   (c) Compute $K_{i,4} \leftarrow c_{1,2}g^{e(c_{1,1}, g^{sk_p})}$.
   (d) For $i$ from $l-2$ down to 1, compute $K_{i,3} \leftarrow c_{i+1,2}g^{e(c_{i+1,1}, H_{2}^{(K_{i+1})})}$.
   (e) Compute $m(\rightarrow c_{1,2}g^{e(c_{1,1}, H_{2}^{(K_{i})})}$.
   (f) Output $m$.

3. An Improvement For Wang-Multi-Use-PRE to Resist Zhang’s Attack

Jindan Zhang and Xu An Wang [18] show Wang-Multi-Use-PRE is not CCA-secure in the corresponding security models in INCoS 2012. In this section, we describe Zhang’s attack and propose an improvement for Wang-PRE to resist Zhang’s attack.

According to Zhang’s attack, let $C' = C_{s}^{(1)} = (C_{s}^{(1,1)}, C_{s}^{(1,2)}, C_{s}^{(1,3)}) = (g^{r_{i,b}}, m_{b}, e(g, pk_{j}^{\gamma})^{r_{i,b}}, (h_{1(c_{1,1})}^{r_{i(c_{1,1})}} h_{2(c_{1,1})}^{r_{i(c_{1,1})}} h_{3(c_{1,1})})^{r_{i,b}})$, an adversary $A$ can malformed it’s $l^{th}$-level ciphertext $C_{j}^{(a)}$ for $pk_{j}$ as below:

1. Issue $O_{RE-EXT}(pk_{j}^{a}, pk_{j})$, get $r_{k,\gamma}$.
2. Choose $t \in R_{Z_{p}^{*}}$, compute $C_{j}^{(2)(1)} = C_{s}^{(1)(1,2)}g_{t}^{t} = g^{r_{i,t}}$.
3. Compute $C_{j}^{(2)(1)} = C_{s}^{(1,2)}e(g_{t}, pk_{j}^{a})e(C_{s}^{(1,3)}g_{t}, r_{k,\gamma}^{-1}) = m_{b}e(g^{r_{i,t}, H_{2}^{(K_{i})}})$.
4. Compute other part of $C_{j}^{(2)}$ as ReEncrypt algorithm.
5. Issue $O_{DEC}(pk_{j}^{a}, C_{j}^{(2)})$, get plaintext $m_{b}$.

Because adversary $A$ can crack the proxy $P_{j}$, get the re-encryption key $r_{k,\gamma}$, and the signing key $ssk_{p}$ corresponding to $P_{j}$’s verification key $K_{j}^{(0)}$. Then it can sign the ciphertext as proxy $P_{j}$, so the challenger cannot distinguish the malformed re-encryption from normal one. Since $C_{j}^{(2)} \neq O_{RE-Enc}(pk_{j}^{a}, pk_{j}, C_{s}^{(1)})$, challenger $C$ responds the query $O_{DEC}(pk_{j}^{a}, C_{j}^{(2)})$, and outputs plaintext $m$ to adversary $A$.

But this problem can be solved simply by adding source $c_{1,2}$ in re-encryption ciphertext, and checking $e(g, c_{1,3}) = e(c_{1,1}, h_{1(c_{1,1})}^{r_{i(c_{1,1})}} h_{2(c_{1,1})}^{r_{i(c_{1,1})}} h_{3(c_{1,1})})$. Before decrypting. The algorithm is modified as follows:

• In the section of the origin Algorithm: ReEncrypt$(rk_{i,\gamma}, C_{pk_{j}}^{(i)}) \rightarrow C_{pk_{j}}^{(i+1)} (i \neq j, l \geq 1)$: 1. To re-encrypt a First-level ciphertext $C_{i}^{(0)}$, denoted by $C_{i}^{(1)}$, on the (c) step: add source $c_{1,2}$ into the ciphertext. Compute $C = (c_{i,1}', c_{i,2}', c_{i,2}'', c_{i,3}', c_{i,2}', c_{i,2}', c_{i,2}', c_{i,2}')$, where $c_{i,2}'' = c_{i,2}'$, other values computed like in the origin Algorithm.
2. To re-encrypt an $l^{th}$-level $(l > 1)$ ciphertext $C_{i}^{(0)}$, on the (d) step: compute $C = (c_{i,1}', \cdots, c_{i,1}', c_{i,2}', c_{i,2}', c_{i,2}', c_{i,2}', c_{i,2}, c_{i,2})$, where $c_{i,2}'' = c_{i,2}$.

• In the section of the origin Algorithm: Decrypt$(sk_{p}, C_{pk_{j}}^{(i)}) \rightarrow m$:
2. For a \(l^{th}-level(l>1)\) ciphertext(\(l>1\)) , change the (a) step as: For \(\forall k \in [1, l]\), check if 
\(e(g, c_{sk}) = e(e(c_{pk}, h_1^{H_{c_1(R_1 \ldots R_{k-1})}}h_2^{H_{c_1(R_1 \ldots R_{k-1})}}), h_3)\). If not, return \(\perp\). Otherwise do the following steps in the origin Algorithm.

4. Proxy Bypass Attack

The PRE scheme allows a proxy to transform a ciphertext encrypted under A’s public key into ciphertext that can be decrypted by B’s secret key. Most of PRE schemes replace Alice’s encrypt factor by the temp security value \(\chi\), then encrypt \(\chi\) with B’s public key. B could get \(\chi\) by decrypt encrypted(\(\chi\)) with its private key, then decrypt re-encrypted ciphertext with \(\chi\). We argue that with a view to chosen ciphertext security, such method may be suitable for single-hop schemes but it cannot fit for multi-use schemes.

The multi-use schemes allow re-encrypting from A to B, then re-encrypt the result from B to C and go on. Once C decrypts a re-encrypt ciphertext from A to B to C, it can get all the temp security values and decrypt other re-encrypted ciphertexts from A to B without proxy. We name this attack as proxy bypass attack. Wang-multi-use PRE scheme has such flaw and is vulnerable to the proxy bypass attack eventhough it has been improved by us.

We give a concrete attack of proxy bypass on Wang-multi-use PRE scheme and then discuss this flaw. This flaw also exists in some other multi-use scheme.

4.1 The Proxy Bypass Attack To Wang-Multi-Use-PRE

In this subsection, we indicate how a polynomial time adversary \(A\) can break the chosen ciphertext security of wang-multi-use-PRE scheme. Adversary \(A\) works as follows:

1. In Setup Phase, adversary \(A\) obtains the public parameters params from the challenger \(C\).
2. In Phase 1 , adversary \(A\) issues queries below:
   (a) \(O_{\text{KeyGen}}(i_A) : A\) gets A’s public key \(pk_A\).
   (b) \(O_{\text{KeyGen}}(i_B) : A\) gets B’s public key \(pk_B\).
   (c) \(O_{\text{Ext}}(pk_B) : A\) gets B’s private key \(sk_B\).
3. In Challenge Phase, adversary \(A\) returns a challenged public key \(pk^* = g^{x_B}\) and two equal-length plaintexts \(m_0, m_1\). Then challenger \(C\) picks \(b \in \{0, 1\}\), sets the challenge ciphertext to \(C_{\ast,b}^{(1)} = \text{Encrypt}(pk^*, m_b)\), and gives \(C^* = C_{\ast,b}^{(1)} = (C_{\ast,b}^{(1)}(1), C_{\ast,b}^{(1)}(2), C_{\ast,b}^{(1)}(3))\) to \(A\). According to the proposed scheme, \(C_{\ast,b}^{(1)}\) is of the following form: \(C_{\ast,b}^{(1)}(1) = g^{e_{b,1}}\), \(C_{\ast,b}^{(1)}(2) = m_b \cdot e(g_1, pk_A)^x_b\), and \(C_{\ast,b}^{(1)}(3) = (h_1^{H_{c_1(R_1 \ldots R_{k-1})}}h_2^{H_{c_1(R_1 \ldots R_{k-1})}}h_3)^{x_b}\).
4. In Phase 2, adversary \(A\) performs the following steps:
   (a) Adversary \(A\) selects a plaintext \(m_2\), computes \(C_{\ast,2}^{(1)} = \text{Encrypt}(pk^*, m_2) = (g^{x_2}, m_2 \cdot e(g_1, pk_A)^{x_2})\).
   (b) Adversary \(A\) issues queries \(O_{\text{RE-Enc}}(pk_A, C_{\ast,2}^{(1)})\), gets \(C_{A,2}^{(2)} = (g^{x_2}, m_2 \cdot e(g^{x_2}, H_2(K_{OA}), h_1^{H_{c_1(R_1 \ldots R_{k-1})}}h_2^{H_{c_1(R_1 \ldots R_{k-1})}}h_3)^{y_2}, R_1^{(A)}, R_2^{(A)} = e(g_1, pk_A)^{x_A}, R_3^{(A)}, R_4^{(A)}, S_{A,2}^{(1)}, R_1^{(AB)}, R_2^{(AB)} = e(g_1, pk_A)^{x_{AB}}, R_3^{(AB)}, R_4^{(AB)}, S_{AB,2}^{(2)}\)).
   (c) Adversary \(A\) issues query \(O_{\text{RE-Enc}}(pk_B, C_{A,2}^{(2)})\), gets \(C_{R,2}^{(3)} = (g^{x_2}, m_2 \cdot e(g^{x_2}, H_2(K_{OA}), h_1^{H_{c_1(R_1 \ldots R_{k-1})}}h_2^{H_{c_1(R_1 \ldots R_{k-1})}}h_3)^{y_2}, R_1^{(A)}, K_{OA} \cdot e(g^{x_A}, H_2(K_{AB})), R_3^{(A)}, R_4^{(A)}, S_{A,2}^{(1)}, R_1^{(AB)}, R_2^{(AB)} = e(g_1, pk_B)^{x_{AB}}, R_3^{(AB)}, R_4^{(AB)}, S_{AB,2}^{(2)}\)). While \(A\) has the private key \(sk_B\), but because of \(C_{\ast,2}^{(1)} \neq C_{\ast,b}^{(1)}\), so according to the security model, it is legal for \(A\) to issue this query.
(d) $\mathcal{A}$ computes $K_{AB} \leftarrow C_{B,2}^{(3,2)} \text{e}(C_{B,2}^{(3,3)}, g_1^{sk_B})$, then computes $K_{sA} \leftarrow C_{B,2}^{(3,2,2)} \text{e}(C_{B,2}^{(3,2,1)}, H_2(K_{AB}))$.

(e) $\mathcal{A}$ issues query $\mathcal{O}_\text{RE-Enc}(pk_A, C_{AB}^{(1)})$, gets $C_{AB}^{(2)} = (g^{r_{sA}}, m_A \cdot \text{e}(g^{r_{sA}}, H_2(K_{sA})), (h_1, h_2, h_3, h_4) r_{sA}, R_1^{(*)} = g^{r_{sA}}, R_2^{(*)} = K_{sA} \cdot \text{e}(g_{\ell_A}, pk_A)^{r_{sA}}, R_3^{(*)}, R_4^{(*)}, R_{A,B}^{(*)})$. Note that since $\mathcal{A}$ doesn’t have the private key $sk_A$, according to the security model, it is legal for $\mathcal{A}$ to issue this query.

(f) With $K_{sA}$, $\mathcal{A}$ computes $m_b \leftarrow C_{A,b}^{(2,1,2)} \text{e}(C_{A,b}^{(2,1,1)}, H_2(K_{sA}))$.

5. Guess: With $m_b$, adversary $\mathcal{A}$ can certainly win the game.

4.2 Discussions

In previous subsection, we have indicated that wang-multi-use-PRE is insecure. The flaw exists in the ReEncrypt period. It converts a ciphertext encrypted by $pk_i$ to a ciphertext encrypted by temp value $K_{ij}$, which is chosen by ReKeyGen and remains unchanged in every ReEncrypt time. In single-hop scheme, Once $\mathcal{A}$ gets the private key of $j$, it cannot get the re-encryption key $rk_{i,j}$ or the re-encryption ciphertext of $C^*$ for $j$. So $\mathcal{A}$ cannot get either temp value $K_j$ or the ciphertext encrypted by $K_j$. But to a multi-use one, as long as $\mathcal{A}$ doesn’t have a path from $C^*_{\ell_1}$ to $C^*_{\ell_n}$, $\mathcal{A}$ can get all the temp values $K_{ij}$ in the path. So $\mathcal{A}$ has both temp value $K_{ij}$ and the ciphertext encrypted by $K_{ij}$, which is chosen by ReKeyGen and remains unchanged in every ReEncrypt time. In single-hop scheme, Once $\mathcal{A}$ gets the private key of $j$, it cannot get the re-encryption key $rk_{i,j}$ or the re-encryption ciphertext of $C^*$ for $j$. So $\mathcal{A}$ cannot get either temp value $K_j$ or the ciphertext encrypted by $K_j$. But to a multi-use one, as long as $\mathcal{A}$ doesn’t have a path from $C^*_{\ell_1}$ to $C^*_{\ell_n}$, $\mathcal{A}$ can get all the temp values $K_{ij}$ in the path. So $\mathcal{A}$ has both temp value $K_{ij}$ and the ciphertext encrypted by $K_{ij}$, which is chosen by ReKeyGen and remains unchanged in every ReEncrypt time. In single-hop scheme, Once $\mathcal{A}$ gets the private key of $j$, it cannot get the re-encryption key $rk_{i,j}$ or the re-encryption ciphertext of $C^*$ for $j$. So $\mathcal{A}$ cannot get either temp value $K_j$ or the ciphertext encrypted by $K_j$. But to a multi-use one, as long as $\mathcal{A}$ doesn’t have a path from $C^*_{\ell_1}$ to $C^*_{\ell_n}$, $\mathcal{A}$ can get all the temp values $K_{ij}$ in the path. So $\mathcal{A}$ has both temp value $K_{ij}$ and the ciphertext encrypted by $K_{ij}$, which is chosen by ReKeyGen and remains unchanged in every ReEncrypt time. In single-hop scheme, Once $\mathcal{A}$ gets the private key of $j$, it cannot get the re-encryption key $rk_{i,j}$ or the re-encryption ciphertext of $C^*$ for $j$. So $\mathcal{A}$ cannot get either temp value $K_j$ or the ciphertext encrypted by $K_j$. But to a multi-use one, as long as $\mathcal{A}$ doesn’t have a path from $C^*_{\ell_1}$ to $C^*_{\ell_n}$, $\mathcal{A}$ can get all the temp values $K_{ij}$ in the path. So $\mathcal{A}$ has both temp value $K_{ij}$ and the ciphertext encrypted by $K_{ij}$, which is chosen by ReKeyGen and remains unchanged in every ReEncrypt time. In single-hop scheme, Once $\mathcal{A}$ gets the private key of $j$, it cannot get the re-encryption key $rk_{i,j}$ or the re-encryption ciphertext of $C^*$ for $j$. So $\mathcal{A}$ cannot get either temp value $K_j$ or the ciphertext encrypted by $K_j$. But to a multi-use one, as long as $\mathcal{A}$ doesn’t have a path from $C^*_{\ell_1}$ to $C^*_{\ell_n}$, $\mathcal{A}$ can get all the temp values $K_{ij}$ in the path. So $\mathcal{A}$ has both temp value $K_{ij}$ and the ciphertext encrypted by $K_{ij}$, which is chosen by ReKeyGen and remains unchanged in every ReEncrypt time. In single-hop scheme, Once $\mathcal{A}$ gets the private key of $j$, it cannot get the re-encryption key $rk_{i,j}$ or the re-encryption ciphertext of $C^*$ for $j$. So $\mathcal{A}$ cannot get either temp value $K_j$ or the ciphertext encrypted by $K_j$. But to a multi-use one, as long as $\mathcal{A}$ doesn’t have a path from $C^*_{\ell_1}$ to $C^*_{\ell_n}$, $\mathcal{A}$ can get all the temp values $K_{ij}$ in the path. So $\mathcal{A}$ has both temp value $K_{ij}$ and the ciphertext encrypted by $K_{ij}$, which is chosen by ReKeyGen and remains unchanged in every ReEncrypt time. In single-hop scheme, Once $\mathcal{A}$ gets the private key of $j$, it cannot get the re-encryption key $rk_{i,j}$ or the re-encryption ciphertext of $C^*$ for $j$. So $\mathcal{A}$ cannot get either temp value $K_j$ or the ciphertext encrypted by $K_j$. But to a multi-use one, as long as $\mathcal{A}$ doesn’t have a path from $C^*_{\ell_1}$ to $C^*_{\ell_n}$, $\mathcal{A}$ can get all the temp values $K_{ij}$ in the path. So $\mathcal{A}$ has both temp value $K_{ij}$ and the ciphertext encrypted by $K_{ij}$, which is chosen by ReKeyGen and remains unchanged in every ReEncrypt time.

5. Conclusions

Wang, et al., [12, 13] proposed a unidirectional and multi-use IND-CCA2 secure PRE scheme, claim that they solve the problem of constructing a unidirectional and multi-use IND-CCA2 secure PRE scheme [4, 7]. But Zhang, et al., [18, 19] have shown their proposal is not IND-CCA-secure by giving a concrete attack.

We first propose an improvement for Wang’s Multi-Use-PRE to resist Zhang’s attack. Zhang uses the proxy, cracks the proxy key, and gets the corresponding signing key. Then it can perform the re-encryption ciphertext and cheats the Decrypt Oracle to decrypt the malformed ciphertext. This article fixes this problem by enhancing the public verifiability of the re-encrypted ciphertext. So the challenger can distinguish the malformed re-encryption ciphertext from normal one.

Secondly, we underline the differences between single-hop scheme and multi-use scheme, and present proxy bypass attack against multi-use IND-CCA secure PRE. Most multi-use PRE schemes do not randomize encrypt key in re-encrypt algorithm. Their IND-CCA security can be broken by this attack. We prove this flaw by attacking wang-multi-use-PRE.

Since Wang’s Multi-Use-PRE is not IND-CCA-secure, the problem of finding a multi-use IND-CCA-secure PRE scheme deserves more efforts to study carefully. We should find an effective solution to avoid the proxy bypass attack in the next stage. Another problem is that Wang’s proposal scheme features a linear ciphertext size. How to design more efficient scheme to reduce ciphertext size explosion is worth of study.
References


Authors

Yi Cai, received her M.S. degree from South China University of Technology, Guangzhou, China, in 2007. She is currently a Ph.D. Candidate in the School of Computer Science and Engineering at the South China University of Technology. Her research interests include information security and distributed computing.

Deyu Qi, is the PhD Candidate Supervisor at the College of Computer Science and Engineering, South China University of Technology. His research interests include software architecture, information security and distributed computing.