On DDos Attack against Proxy in Re-encryption and Re-signature

Xu an Wang

Key Laboratory of Information and Network Security
Engneering College of Chinese Armed Police Force, P.R. China
wangxahq@yahoo.com.cn

Abstract. In 1998, Blaze, Bleumer, and Strauss proposed new kind of cryptographic primitives called proxy re-encryption and proxy re-signature[BBS98]. In proxy re-encryption, a proxy can transform a ciphertext computated under Alice's public key into one that can be opened under Bob's decryption key. In proxy re-signature, a proxy can transform a signature computated under Alice's secret key into one that can be verified by Bob's public key. In 2005, Ateniese et al proposed a few new re-encryption schemes and discussed its several potential applications especially in the secure distributed storage[AFGH05]. In 2006, they proposed another few re-signature schemes and also discussed its several potential applications[AH06]. They predicated that re-encryption and re-signature will play an important role in our life. Since then, researchers are sparked to give new lights to this area. Many excellent schemes have been proposed. In this paper, we introduce a new attack- DDos attack against proxy in the proxy re-cryptography. Although this attack can also be implemented against other cryptographic primitives, the danger caused by it in proxy re-cryptography seems more serious. We revisit the current literature, paying attention on their resisting DDos attack ability. We suggest a solution to decline the impact of DDos attacking. Also we give a new efficient re-encryption scheme which can achieve CCA2 secure based on Cramer-Shoup encryption scheme and prove its security. We point out this is the most efficient proxy re-encryption schemes for the proxy which can achieve CCA2 secure in the literature. At last we give our conclusions with hoping researchers give more attention on this attack.

1 Introduction

The concept of proxy re-cryptography dates back to the work of Blaze, Bleumer, and Strauss in 1998. The goal of proxy re-encryptiohn is to securely enable the re-encryption of ciphertexts from one key to another, without relying on trusted parties.Simirlarly, the goal of proxy re-signature is to securely enable the signature signed by one to transform to be another signature on the same message signed by another without relying on trusted parties.In 2005, Ateniese et al proposed a few new re-encryption schemes and discussed its several potential applications especially in the secure distributed storage [AFGH05]. In

2006, they proposed another few re-signature schemes and also discussed its several potential applications[AH06]. They predicated that re-encryption and re-signature will play an important role in our life. Since then, researchers are sparked to give new lights to this area.Many excellent schemes have been proposed,especially,the IEEE P1363.3 standard working group is establishing the standard for re-encryption, which will certainly give new power on researching in re-cryptography.

Generally speaking, we can split the research area into two parts-re-encryption part and re-signature part.In re-encryption area,besides the pioneering work[BBS98] [AFGH05],researchers have done some good work on achieving CCA2 re-encryption such as [CH07,LV08a],some good work on achieving re-encryption in identity-based settings [GA07] or between identity-based setting and PKI setting[M07] [P1363.3/06],some good work on achieving re-encryption without random oracle [CT07,SXC08],some good work on other aspects of re-encryption[HRSV07,LV08b] [LV08c,CAP08,KR08,MAL07,MA08].In re-signature area,some good work also has been done [AH06,SCWL07].

[**Motivation**]Undoubtedly security is the most important thing when we consider on re-cryptography, but we can not ignore the practice of this new cryptographic primitive either. But in the current literature, very little attention has been paid on efficiency, especially on the efficiency of proxy. Because in most circumstances the semi-trusted proxies are relative scarce equipments,we are certain that they are easily suffering from the DDos attacking, especially when the proxy must implement lots of computation. We can find the proof in the [AFGH05]:

> *Our server is able to sustain 100 re-encryptions/sec until reaching about 1,000 outstanding requests. The server coped with up to 10,000 outstanding re-encryption requests, but quickly spiraled downwards thereafter.*

So we must count the DDos attack in re-cryptography, at least decrease the dangerous caused by this attack.

[**Our Contribution**]In this paper, we pay attention to the efficiency of the proxy. The workload of proxy can be divided into two parts: one part for checking the initial-signature or initial-ciphertext validity and the other part for re-encryption or re-signature. We revisit the literature on re-encryption and re-signature, especially pay attention on their ability of resisting DDos attacking, we propose a solution to decrease the dangerous of DDos attack on proxy, Also we give a new efficient re-encryption scheme which can achieve CCA2 secure based on Cramer-Shoup encryption scheme and prove its security. We point out this is the most efficient proxy re-encryption schemes for the proxy which can achieve CCA2 secure in the literature.

We organize our paper as following:In section 2,we revisit the current literature on proxy re-encryption,especially on the efficiency of proxy, and give some comparisons to the existing schemes; In section 3, we revisit the current

literature on proxy re-signature,especially on the efficiency of proxy,and also give some comparisons to the existing schemes; In section 4,we propose a solution to decrease the dangerous caused by DDos attack;In section 5, we give a new efficient re-encryption scheme which can achieve CCA2 secure based on Cramer-Shoup encryption scheme and prove its security;In section 6, we give our concluding remarks.

2 Revisit Re-encryption Schemes

In this section, we revisit most known re-encryption schemes in the literature, including traditional public key encryption environment , identity-based re-encryption environment and hybrid encryption environment.

2.1 Re-encryption in Public Key Certificates Setting

[**AFGH05 Scheme**] This Elgamal based scheme operates on bilinear parings in group G_1, G_2, G_T .

1. **Key Generation** (KG)**:** A user A's key pair is of the form $pk_a = (z^{a_1}, g_2^a), sk_a = (a_1, a_2)$.
2. **Re-Encryption Key Generation** (RG)**:** A user A delegates to B by publishing the re-encryption key $rk_{AB} = g^{a_1 b_2} \in G_1$, computed from B's public key.
3. **First-Level Encryption** (E_1)**:** To encrypt a message $m \in G_2$ under pk_a in such a way that it can only be decrypted by the holder of sk_a, output $c_{a,1} = (Z^{a_1 k}, mZ^k)$.
4. **Second-Level Encryption** (E_2)**:** To encrypt a message $m \in G_2$ under pk_a in such a way that it can be decrypted by A and her delegatees, output $c_{a,r} = (g^k, mZ^{a_1 k})$.
5. **Re-Encryption** (R)**:** Any one can change a second-level ciphertext for A into a first-level ciphertext for B with $rk_{AB} = g^{a_1 b_2}$. From $c_{a,r} = (g^k, mZ^{a_1 k})$, compute $e(g^k, g^{a_1 b_2}) = Z^{b_2 a_1 k}$ and publish $c_{b,2} = (Z^{b_2 a_1 k}, mZ^{a_1 k}) = (Z^{b_2 k'}, mZ^{k'})$.
6. **Decryption** (D_1, D_2)**:** To decrypt a first-level ciphertext $c_a = (\alpha, \beta)$ with secret key $sk = a$, compute $m = \beta / \alpha^{1/a}$. To decrypt a second-level ciphertext $c_a = (\alpha, \beta)$ with secret key $sk = a$, compute $m = \beta / e(\alpha, g)^{1/a}$.

In this scheme the delegation key need not be private. The scheme does not require initial ciphertext checking, but the proxy needs one pairing to do the re-encryption.This scheme has the property of containing two-level ciphertexts, some additional computation must be operated to distinguish the first level and second level.This scheme is not CCA2 secure.

we conclude this section with Table 1, which shows the efficiency of the proxy and the schemes' ability of resisting DDos attack in public key certificate setting(Description of some schemes list in the appendix).

Scheme	Security	Re-enc key be protected	Checking	Re-encryption
BBS98	CPA	Yes	No	1exp
AFGH05	CPA	No	No	1pairing
CH07	CCA2	Yes	1verifying and 4pairing	1exp
LV08a	CCA2	No	1verifying and 2pairing	3pairing
LV08b	CPA	No	No	(n+1)exp and 2pairing
KR08**	WCCA2	Yes	No	1exp
MAL07@	CPA	Yes	No	2pairing
MA08@@	CPA	No	No	3pairing

* This scheme is designed for tracing the malicious proxy.
** This is a RSA-TBOS signcryption with proxy re-encryption scheme.
@ This scheme is a group based proxy re-encryption scheme and we think it is CPA secure although the authors claimed it to be CCA2 secure.
@@ This scheme is a group based proxy re-encryption scheme and we think it is CPA secure although the authors claimed it to be CCA2 secure.
⋆ Our scheme can be seen in section 5,which is the most efficient among the CCA2 secure schemes.

Table 1. Efficiency of the proxy in the public key certificate setting

2.2 Re-encryption in Identity Based Setting

[**M07b Scheme**] There are five entities involved in an identity-based proxy re-encryption system, delegator, proxy, delegatee, PKG and Re-encryption Key Generator, RKG.In this system, each of delegator and delegatee is an IBE user. The RKG generates re-encryption keys and sets them into the proxy via secure channel.

- The underlying IBE system (BB-IBE system):
  1. $\mathbf{SetUp_{IBE}}(\mathbf{k})$.Given a security parameter k, select a random generator $g \in G$ and random elements $g_2, h \in G$. Pick a random $\alpha \in Z_p^*$. Set $g_1 = g$,$mk = g_2^\alpha$, and $parms = (g, g_1, g_2, h)$. Let mk be the master-secret key and let $parms$ be the public parameters.
  2. $\mathbf{KeyGen_{IBE}}(\mathbf{mk}, \mathbf{parms}, \mathbf{ID})$. Given $mk = g_2^\alpha$ and ID with $parms$, pick a random $u \in Z_p^*$. Set $sk_{ID} = (d_0, d_1) = (g_2^\alpha (g_1^{ID} h)^u, g^u)$.
  3. $\mathbf{Enc_{IBE}}(\mathbf{ID}, \mathbf{parms}, \mathbf{M})$. To encrypt a message $M \in G_1$ under the public key $ID \in Z_p^*$, pick a random $r \in Z_p^*$ and compute $C_{ID} = (g^r, (g_1^{ID} h)^r, Me(g_1, g_2)^r) \in G^2 \times G_1$.
  4. $\mathbf{Dec_{IBE}}(\mathbf{sk_{ID}}, \mathbf{parms}, \mathbf{C_{ID}})$. Given ciphertext $C_{ID} = (C_1, C_2, C_3)$ and the secret key $sk_{ID} = (d_0, d_1)$ with $prams$, compute $M = C_3 e(d_1, C_2)/e(d_0, C_1)$.
- The delegation system:
  1. $\mathbf{EGen}(sk_{ID}, parms)$. Given $sk_{ID} = (d_0, d_1) = (g_2^\alpha (g_1^{ID} h)^u, g^u)$ with $parms$, set $e_{ID} = d_1$.
  2. $KeyGen_{RKG}(mk, parms)$. Given $mk =$ with $parms$, set $sk_R = .$
  3. $\mathbf{KeyGen}_{PRO}(\mathbf{sk}_R, e_{ID'}, parms, ID, ID')$. Given $sk_R = \alpha$, $e'_{ID} = g^{u'}$ with $parms$, set $rk_{ID \to ID'} = (ID \to ID', g^{u'\alpha})$.

4. **ReEnc**$(rk_{ID\to ID'}, parms, C_{ID}, ID, ID')$. Given the delegator's identity ID, the delegatee's identity ID', $rk_{IDID'} = (ID \to ID', g^{u'\alpha})$,$C_{ID} = (C_1, C_2, C_3)$ with $parms$, re-encrypt the ciphertext C_{ID} into $C_{ID'}$ as follows. $C_{ID'} = (C'_1, C'_2, C'_3) = (C_1, C_2, C_3 e(C_1^{ID'-ID}, g^{u'})) \in G^2 \times G_1$.
5. **Check**$(parms, C_{ID}, ID)$. Given the delegator's identity ID and $C_{ID} = (C_1, C_2, C_3)$ with $parms$, compute $v_0 = e(C_1, g_1^{ID}h)$ and $v_1 = e(C_2, g)$. If $v_0 = v_1$ then output 1. Otherwise output 0.

In this scheme, the proxy key needs not be private,checking the initial ciphertext needs two pairing computation and re-encryption needs one pairing computation.This scheme is CPA secure and is being standardized by IEEE P1363.3 Workgroup.

2.3 Re-encryption From CBE to IBE

[M07a Scheme] The hybrid proxy re-encryption system involving the ElGamal-type CBE system and the BB-IBE system.

- The underlying IBE system (BB-IBE system):
  1. $\mathbf{SetUp_{IBE}}(\mathbf{k})$.Given a security parameter k, select a random generator $g \in G$ and random elements $g_2, h \in G$. Pick a random $\alpha \in Z_p^*$. Set $g_1 = g$,$mk = g_2^\alpha$, and $parms = (g, g_1, g_2, h)$. Let mk be the master-secret key and let $parms$ be the public parameters.
  2. $\mathbf{KeyGen_{IBE}}(\mathbf{mk}, \mathbf{parms}, \mathbf{ID})$. Given $mk = g_2^\alpha$ and ID with $parms$, pick a random $u \in Z_p^*$. Set $sk_{ID} = (d_0, d_1) = (g_2^\alpha (g_1^{ID}h)^u, g^u)$.
  3. $\mathbf{Enc_{IBE}}(\mathbf{ID}, \mathbf{parms}, \mathbf{M})$. To encrypt a message $M \in G_1$ under the public key $ID \in Z_p^*$, pick a random $r \in Z_p^*$ and compute $C_{ID} = (g^r, (g_1^{ID}h)^r, Me(g_1, g_2)^r) \in G^2 \times G_1$.
  4. $\mathbf{Dec_{IBE}}(\mathbf{sk_{ID}}, \mathbf{parms}, \mathbf{C_{ID}})$. Given ciphertext $C_{ID} = (C_1, C_2, C_3)$ and the secret key $sk_{ID} = (d_0, d_1)$ with $prams$, compute $M = C_3 e(d_1, C_2)/e(d_0, C_1)$.
- The underlying CBE system (ElGamal-type CBE system):
  1. $\mathbf{KeyGen_{CBE}}(\mathbf{k}, \mathbf{parms})$. Given a security parameter k and $parms$, pick a random $\beta, \theta, \delta \in Z_p$. Set $g_3 = g^\theta$,$g_4 = g_1^\beta$ and $g_5 = h^\delta$. The public key is $pk = (g_3, g_4, g_5)$. The secret random key is $sk = (\beta, \theta, \delta)$.
  2. $\mathbf{Enc_{CBE}}(\mathbf{pk}, \mathbf{parms}, \mathbf{M})$. Given $pk = (g_3, g_4, g_5)$ and a message M with $parms$, pick a random $r \in Z_p^*$ and compute $C_{PK} = (g_3^r, g_4^r, g_5^r, Me(g_1, g_2)^r \in G^3 \times G_1$.
  3. $\mathbf{Dec_{CBE}}(\mathbf{sk}, \mathbf{parms}, \mathbf{C_{PK}})$. Given $C_{PK} = (C_1, C_2, C_3, C_4)$ and the secret key $sk = (\beta, \theta, \delta)$ with $parms$, compute $M = C4/e(C_2^{1/\beta}, g_2)$.
- The delegation system:
  1. $\mathbf{EGen}(\mathbf{sk_{ID}}, \mathbf{parms})$. Given $sk_{ID} = (d_0, d_1) = (g_2^\alpha (g_1^{ID}h)^u, g^u)$ for ID with $parms$, set $e_{ID} = d_1 = g^u$.
  2. $\mathbf{KeyGen_{PRO}}(\mathbf{sk}, \mathbf{e_{ID}}, \mathbf{parms})$. Given $sk = (\beta, \theta, \delta)$ and $e_{ID} = g^u$ for ID with $parms$, set $rk_{ID} = (\beta, g^{u/\theta}, \delta)$.

3. **ReEnc**($\mathbf{rk_{ID}}, \mathbf{parms}, \mathbf{C_{PK}}, \mathbf{ID}$). Given a CBE ciphertext $C_{PK} = (C_1, C_2, C_3, C_4)$, the re-encryption key $rk_{ID} = (\beta, g^{u/\theta}, \delta)$ and ID with $parms$, re-encrypt the ciphertext CPKinto CIDas follows. $C_{ID} = (C_1', C_2', C_3') = (C_1^{1/\theta}, C_3^{1/\delta}, C_4 e(g^{u/\theta}, C_2^{ID})) \in G^2 \times G_1$.
4. **Check**($\mathbf{parms}, \mathbf{C_{PK}}, \mathbf{pk}$). Given $C_{PK} = (C_1, C_2, C_3, C_4)$ and $pk = (g_3, g_4, g_5)$ with $parms$, set $v_1 = e(C_1, g_4)$, $v_2 = e(C_2, g_3)$, $v_3 = e(C_2, g^5)$ and $v_4 = e(C_3, g^4)$. If $v_1 = v_2$ and $v_3 = v_4$ then output 1, otherwise output 0.

In this scheme, the proxy key need be private, checking the initial ciphertext requires four pairing computation and re-encryption requires three exponentiation and one pairing computation.This scheme is IND-ID-CPA secure and is being standardized by IEEEP1363.3 workgroup.

we conclude the above two sections with Table 2, which shows the efficiency of the proxy and the schemes' ability of resisting DDos attack in identity based setting(Description of some schemes list in the appendix).

Scheme	**Security**	**Re-enc key be protected**	**Checking**	**Re-encryption**
MA06	ID-CCA2	No	2pairing	2pairing
M07b	ID-CPA	No	2pairing	1pairing
P1363.3/06a**	ID-CPA	No	No	1pairing
CT07*	ID-CPA	No	1verifying	(n+2)exp
SXC08	ID-CCA2	No	1verifying	(2n+1)exp and 1signing
M07a	ID-CPA	Yes	4paring	3exp and 1pairing
P1363.3/06b***	ID-CPA	Yes	4paring	2exp and 2paring

* This scheme is cryptanalysis by [SXC08], which its original authors claimed to be IND-ID-CCA2 secure. But In [SXC08] it is proved to be IND-ID-CPA secure.
** This scheme is designed for the identity based setting.
*** This scheme is designed for transformation from CBE to IBE.

Table 2. Efficiency of the proxy in the identity based and hybrid setting

3 Revisit Re-signature Schemes

[**AH06a Scheme**] This scheme based on the BLS short signature scheme.It requires a bilinear map $e : G_1 \times G_1 \rightarrow G_2$ operates over two groups G_1, G_2 of prime order q. The global parameters are (e, q, G_1, G_2, g, H), where g generates G_1 and H is a hash function from arbitrary strings to elements in G_1.

1. **KeyGeneration**(**KeyGen**): On input the security parameter 1^k, select a random $a \in Z_q$, and output the key pair $pk = g^a$ and $sk = a$.
2. **Re** $-$ **SignatureKeyGeneration**(**ReKey**): On input two secret keys $sk_A = a$, $sk_B = b$ (the public keys are not required for this algorithm), output the resignature key $rk_{A \rightarrow B} = b/a(modq)$.

3. **Sign**(**Sign**): On input a secret key $sk = a$ and a message m, output $\delta = H(m)^a$.
4. $\mathbf{Re-Sign}$(**ReSign**):On input a re-signature key $rk_{A\to B}$, a public key pk_A, a signature δ, and a message m, check that $Verify(pk_A, m, \delta) = 1$. If δ does not verify, output $\perp$; otherwise, output $\delta^{'=\delta^{rk_{A\to B}}}$.
5. **Verify**(**Verify**): On input a public key pk_A, a message m, and a purported signature δ, output 1 if $e(\delta, g) = e(H(m), pk_A)$ and 0 otherwise.

In this scheme, the proxy key needs to be private, checking the initial signature requires one verifying ,that is , two pairing computation. Re-signature requires one exponentiation computation.This scheme is EU-CMA secure.

we conclude this sections with Table 3, which shows the efficiency of the proxy and the signature schemes' ability of resisting DDos attack.

Scheme	**Security**	**Re-sig key be protected**	**Checking**	**Re-signature**
BBS98	EU-CMA	Yes	(3k)exp	(k)multi
AH06a*	EU-CMA	Yes	2pairing	1exp
AH06b**	EU-CMA	No	2pairing	1exp
AH06c***	EU-CMA	Yes	1verifying and 2pairing	2exp and 1signing
SCWL07$^{@}_{a}$	EU-CMA	Yes	3pairing	2exp
SCWL07$^{@@}_{b}$	EU-CMA	Yes	4paring	2exp
LV08c$^{\circledcirc}_{a}$	EU-CMA	No	2pairing or 4pairing*	3exp or 1exp*
LV08c$^{\circledast}_{b}$	EU-CMA	No	(2l+2)pairing	(2l+3)exp

* This scheme is a bidirectional proxy re-signature scheme.
** This scheme is uni-directional single-use scheme with public re-signature key.
*** This scheme is uni-directional single-use scheme with private re-signature key.
@ This scheme is a multi-use bidirectional scheme.
@@ This scheme is an ID based multi-use bidirectional scheme.
⊙ This scheme is a single-hop scheme.
⋆ This means the forwarding computation is for level 1 signature and the latter computation is for level 2 signature.
⊛ This scheme is a multi-hop scheme.

Table 3. Efficiency of the proxy in the identity based and hybrid setting for re-signature

4 A Solution to Decrease the Danger Caused by DDos Attack

Generally speaking,the workload of proxy can be divided into two parts, one part for the ciphertext or signature validity checking, one part for the re-encrypting or re-signing.We note that the validity checking can be done without any secret, but the re-encrypting or re-signing needs some secret to be done or at least needs to be done at some semi-trusted nodes. So we can separate the workload of the

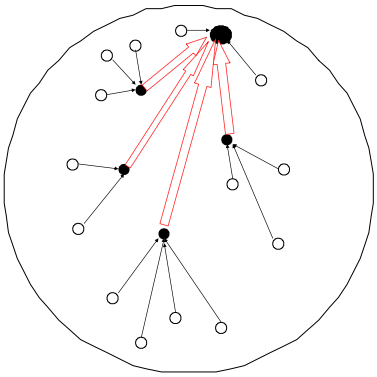

Fig. 1. A distribution of multi-proxies in re-encryption or re-signature setting

proxy into two parts. We introduce some aided proxies in the system, which can be some equipments having lots of computation power.But we do not plant any private value in it,all we need is these equipments can be trusted to honestly do the checking work.Furthermore, we denotes the original proxies as main proxies, which also must be powerful. It must do all the work of re-encryption or re-signature, and it must answer the re-encrypt or re-sign Oracle's queries.

We also note the re-encryption and re-signature schemes can be divided by two kinds - one kind of proxy key must be private and the other need not - by the proxy key being private or not.In the case of proxy key being public, the difference between main proxies and aided proxies can be disappeared, that is to say, all the proxies run in the peer to peer model. In the case of proxy key being private, only the main proxies can do the re-encryption or re-signature work.

We express our idea in Firgure 1. In the firgure, the white nodes denote the end user;the small black nodes denote the aided proxy which only do the checking work;the big black node denotes the main proxy which do the checking and re-encryption or re-signature work; the thin black arrows denote the communication from the end user to the aided proxy or main proxy;the big red arrows denote the communication from the aided proxy to the main proxy which happens only after the initial ciphertext or signature passing the checking.

But we note that DDos attacking can not be avoided, all we can do is just decreasing the damage caused by it. So we remark our solution just tries best to resist the DDos attacking instead of avoiding it.

5 A New Efficient Re-encryption Scheme Based on Cramer-Shoup Encryption and Its Security Proof

5.1 New Efficient Re-encryption Scheme Based on Cramer-Shoup Encryption

[**Basic Cramer-Shoup Encryption**] The scheme assume a group G of prime order q, where q is large. It also assume that cleartext messages are (or can be encoded as) elements of G. It also use a universal one-way family of hash functions that map long bit strings to elements of Z_q.

1. **KeyGeneration**. The key generation algorithm runs as follows. Random elements $g_1, g_2 \in G$ are chosen, and random elements $x_1, x_2, y_1, y_2, z \in Z_q$ are also chosen. Next, the group elements $c = g_1^{x_1} g_2^{x_2}, d = g_1^{y_1} g_2^{y_2}, h = g_1^z$ are computed. Next, a hash function H is chosen from the family of universal one-way hash functions. The public key is (g_1, g_2, c, d, h, H), and the private key is (x_1, x_2, y_1, y_2, z).
2. **Encryption**. Given a message $m \in G$, the encryption algorithm runs as follows. First, it chooses $r \in Z_q$ at random. Then it computes $u_1 = g_1^r, u_2 = g_2^r, e = h^r m, \alpha = H(u_1, u_2, e), v = c^r d^{r\alpha}$. The ciphertext is (u_1, u_2, e, v).
3. **Decryption**. Given a ciphertext (u_1, u_2, e, v), the decryption algorithm runs as follows. It first computes $\alpha = H(u_1, u_2, e)$, and tests if $u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha} = v$. If this condition does not hold, the decryption algorithm outputs "reject"; otherwise, it outputs $m = e/u_1^z$.

[**New Efficient Re-encryption Scheme**] This scheme shares the same *parameters* and *KeyGeneration* algorithm with Cramer-Shoup scheme.The delegator's public key is $pk_A = (g_1, g_2, c, d, h, H)$, his private key is $sk_A = (x_1, x_2, y_1, y_2, z)$; The delegatee's public key is $pk_B = (g_1, g_2, c', d', h', H)$,his private key is $sk_B = (x_1', x_2', y_1', y_2', z')$.

1. $\mathbf{Re-KeyGeneration}(\mathbf{ReKeyGen})$ On input $sk_A = (x_1, x_2, y_1, y_2, z)$, $sk_B = (x_1', x_2', y_1', y_2', z')$,output the re-signature key $rk_{A\to B} = (kx_1, kx_2, ky_1, ky_2, k'x_1', k'x_2', k'y_1', k'y_2', z/z')$ where $k, k' \in Z_q^*$.The delegator preserves k for ciphertext-transformation purpose and the delegatee preserves k' for decryption purpose.
2. **Encryption**. Given a message $m \in G$, the encryption algorithm runs as follows. First, it chooses $r \in Z_q$ at random. Then it computes $u_1 = g_1^r, u_2 = g_2^r, e = h^r m, \alpha = H(u_1, u_2, e), v = c^r d^{r\alpha}$. The ciphertext is (u_1, u_2, e, v).
3. $\mathbf{Ciphertext-transformation}(\mathbf{CTran})$ On input initial ciphertext (u_1, u_2, e, v), the delegator transforms it to be (u_1, u_2, e, v') where $v' = v^k$.
4. $\mathbf{Re-encryption}(\mathbf{ReEnc})$ The proxy first verifies ciphertext (u_1, u_2, e, v')'s validity. if $u_1^{kx_1+ky_1\alpha} u_2^{kx_2+ky_2\alpha} \neq v'$ where $\alpha = H(u_1, u_2, e)$, then return "Reject",else computes $u_1' = {u_1}^{z/z'}, u_2' = {u_2}^{z/z'}, e' = e, \alpha' = H(u_1', u_2', e'), v'' = {u_1}^{k'x_1'z/z'} {u_2}^{k'x_2'z/z'} {u_1}^{k'y_1'\alpha'z/z'} {u_2}^{k'y_2'\alpha'z/z'}$.

5. **Decryption$_1$(Dec$_1$)**. Given a re-encryption ciphertext (u_1', u_2', e', v''), the decryption algorithm runs as follows. It first computes $\alpha' = H(u_1', u_2', e')$, and tests if $(u_1')^{k'(x_1'+y_1'\alpha')}(u_2')^{k'(x_2'+y_2'\alpha')} = v''$. If this condition does not hold, the decryption algorithm outputs "reject"; otherwise, it outputs $m = e'/(u_1')^{z'}$.
6. **Decryption$_2$(Dec$_2$)**. Given a normal ciphertext (u_1, u_2, e, v), the decryption algorithm runs as follows. It first computes $\alpha = H(u_1, u_2, e)$, and tests if $(u_1)^{(x_1+y_1\alpha)}(u_2)^{(x_2+y_2\alpha)} = v$. If this condition does not hold, the decryption algorithm outputs "reject"; otherwise, it outputs $m = e/(u_1)^z$.

First we show our re-encryption ciphertext is a valid Cramer-Shoup ciphertext.

$$
\begin{aligned}
u_1' &= {u_1}^{z/z'} = g_1^{rz/z'} = g_1^{r'} \\
u_2' &= {u_2}^{z/z'} = g_2^{rz/z'} = g_2^{r'} \\
e' &= e = h^r m = g_1^{zr} m = (g_1^{z'})^{rz/z'} = (h')^{rz/z'} = (h')^{r'} \\
\alpha' &= H(u_1', u_2', e') \\
v'' &= {u_1}^{k'x_1'z/z'} {u_2}^{k'x_2'z/z'} {u_1}^{k'y_1'\alpha'z/z'} {u_2}^{k'y_2'\alpha'z/z'} \\
&= {g_1}^{rk'x_1'z/z'} {g_2}^{rk'x_2'z/z'} {g_1}^{rk'y_1'\alpha'z/z'} {g_2}^{rk'y_2'\alpha'z/z'} \\
&= {g_1}^{x_1'rk'z/z'} {g_2}^{x_2'rk'z/z'} {g_1}^{y_1'rk'z/z'\alpha'} {g_2}^{y_2'rk'z/z'\alpha'} \\
&= (c'^{k'})^{rz/z'} (d'^{k'})^{rz/z'\alpha'} \\
&= (c'^{k'})^{r'} (d'^{k'})^{r'\alpha'} \\
&= (c'')^{r'} (d'')^{r'\alpha}
\end{aligned}
$$

Next we note that,in our scheme, the delegator must preserve k,it has the Ciphertext-transformation algorithm which operated by delegator by k,and the delegatee must preserve k' for validating the re-encryption ciphertext. Everything we do like that is just for CCA2 security, which we will explain in the next section.And we believe that ciphertext-transformation algorithm is a reasonable step for re-encryption, after all, re-encryption is possible only after the delegator agrees this.The additional cost is that delegator and delegatee must preserve k and k' for every re-key generation.This is a shortcoming which we hope it can be improved in the future.

At last, we compare our scheme with other CCA2 secure scheme.First, the proxy key in our scheme must be private, the initial ciphertext checking requires two exponentiation computation and the re-encryption requires six exponentiation computation.Because our scheme needs no pairing(one pairing almost equals six exponentiation), so our scheme's proxy is most efficient so far, which can be seen in Table 4.

5.2 Security Analysis

First we give our security model for our new proxy re-encryption scheme,we follow the models in [CH07,AH06].Then we prove our scheme in this model.

Definition 1. (Bidirectional PRE-CCA game) *Let k be the security parameter. Let A be an oracleTM, representing the adversary. The game consists of an execution of A with the following oracles, which can be invoked multiple times in any order, subject to the constraints below:*

- **Uncorrupted keygeneration O_{keygen}:** *Obtain a new key pair as $(pk, sk) \leftarrow KeyGen(1^k)$. A is given pk .*
- **Corrupted key generation$O_{corkeygen}$:** *Obtain a new key pair as $(pk, sk) \leftarrow KeyGen(1^k)$.$A$ is given pk , sk.*
- **Re-encryption key generation** $O_{rekeygen}$*:On input (pk, pk') by the adversary, where pk, pk' were generated before by $KeyGen$, return the re-encryption key $rk_{pk \leftrightarrow pk'} = ReKeyGen(sk, sk')$ where sk , sk' are the secret keys that correspond to pk, pk'. We require that either both pk and pk' are corrupted, or alternatively both are uncorrupted. We do not allow for re-encryption key generation queries between a corrupted and an un- corrupted key. (This represents the restriction that the identities of parties whose security is compromised should be fixed in advance.)*
- **Encryptionoracle**. *Given a message $m \in G$,the output ciphertext is (u_1, u_2, e, v).*
- **Ciphertext-transformation oracle O_{ctra}:** *On input ciphertext (u_1, u_2, e, v), the delegator transforms it to be (u_1, u_2, e, v') where $v' = v^k$.*
- **Challenge oracle:** *This oracle can be queried only once. On input (pk^*, m_0, m_1), where pk^* is called the challenge key, the oracle chooses a bit $b \leftarrow \{0, 1\}$ and returns the challenge ciphertext $C^* = Enc(pk^*, m_b)$. (As we note later, the challenge key must be uncorrupted for A to win).*
- **Re-encryption O_{renc}:** *On input (pk, pk', C), where pk,pk' were generated before by $KeyGen$, if pk' is corrupted, then return a special symbol $\perp$ which is not in the domains of messages or ciphertexts. Else, return the re-encrypted ciphertext $C' = ReEnc(ReKeyGen(sk, sk'), C)$.*
- **Decryption oracle O_{dec_1}:** *On input a re-encryption ciphertext (pk, C), if pk was not generated before by $KeyGen$, then return a special symbol $\perp$ which is not in the domain D of messages. Else, return $Dec_1(sk, C)$.*
- **Decryption oracle O_{dec_2}:** *On input a normal ciphertext (pk, C), if pk was not generated before by $KeyGen$, then return a special symbol $\perp$ which is not in the domain D of messages. Else, return $Dec_2(sk, C)$.*
- **Decision oracle**: *This oracle can also be queried only once. On input b': If $b' = b$ and the challenge key pk^* is not corrupted, then output 1; else output 0.*

We say that A wins the $PRE-CCA$ game with advantage ϵ if the probability, over the random choices of A and the oracles, that the decision oracle is invoked and outputs 1, is at least $1/2 + \epsilon$.

Internal and External Security. Our security model protects users from two types of attacks: those launched from parties outside the system (External Security), and those launched from parties inside the system, such as the proxy, another delegation partner, or some collusion between them (Internal Security). We now provide both intuition and a formalization of these security notions.

External Security: Our first security notion protects a user from adversaries outside the system (i.e., excluding the proxy and any delegation partners).

Definition 2. *A PRE scheme is chosen-ciphertext secure if the probability*

$$Pr[(pk^*, sk^*) \leftarrow O(\lambda), \{(pk_x, sk_x) \leftarrow O_{corkeygen}(\lambda)\}, \{(pk_h, sk_h) \leftarrow O_{keygen}(\lambda)\},$$
$$\{R_{\star h} \leftarrow O_{rekeygen(sk^*, pk_h)}\}, \{R_{h\star} \leftarrow O_{rekeygen(sk_h, pk^*)}\},$$
$$\{R_{hx} \leftarrow O_{rekeygen}(sk_h, pk_x)\}, \{R_{xh} \leftarrow O_{rekeygen}(sk_x, pk_h)\},$$
$$\{R_{hh'} \leftarrow O_{rekeygen(sk_h, pk_{h'})}\}, \{R_{xx'} \leftarrow O_{rekeygen}(sk_x, pk_x)\},$$
$$(m_0, m_1, St) \leftarrow A^{O_{dec_1}, O_{dec_2}, O_{renc}, O_{ctra}}(pk^\star, \{(pk_x, sk_x)\}, \{pk_h\}, \{R_{x\star}\}, \{R_{h\star}\},$$
$$\{R_{\star h}\}, \{R_{xh}\}, \{R_{hx}\}, \{R_{hh'}\}, \{R_{xx'}\}),$$
$$d^\star \xleftarrow{R} \{0,1\}, C^\star = Enc_2(m_{d\star}, pk^\star), d' \leftarrow A^{O_{renc}, O_{ctra}}_{O_{dec_1}, O_{dec_2}}(C^\star, St) : d' = d^\star]$$

is negligibly close to 1/2 for any PPT adversary A. In our notation, St is a state information maintained by A while (pk, sk) is the target user's key pair generated by the challenger that also chooses other keys for corrupt and honest parties. For other honest parties, keys are subscripted by h or h'and we subscript corrupt keys by x or x'. The adversary is given access to all re-encryption keys but those that would allow re-encrypting from the target user to a corrupt one. In the game, A is said to have advantage ϵ if this probability, taken over random choices of A and all oracles, is at least $1/2 + \epsilon$.

Internal Security: Our second security notion protects a user, as much as possible, when they are fooled into trusting a rogue proxy and/or delegation partner (who may be colluding).

- **Limited Proxy**: If the delegator and the delegatee are both honest, then the proxy cannot decrypt the ciphertext or not even distinguish two ciphertexts. We will show our scheme is CPA secure for the proxy.
- **Delegatee Security**: Because in our scheme, the proxy key is private, then If the delegator and proxy collude, the delegatee is no longer safe.
- **Delegator Security**: As the same reason above,the delegator is no longer safe as the delegator and proxy collude in our scheme.

After we set the model for our scheme , we prove our scheme's security.We give two theorems as following:

Theorem 1. *Our re-encryption scheme is secure against adaptive chosen ciphertext attack for the external adversaries assuming that (1) the hash function H is chosen from a universal one-way family, and (2) the Diffe-Hellman decision problem is hard in the group G.*

Proof. The intuition is that the initial ciphertext is a Cramer-Shoup ciphertext,nobody can get help from the re-encryption oracle by querying the oracle

with "invalid ciphertext".Also our re-encryption ciphertext is a Cramer-Shoup ciphertext, nobody can get help either.The proxy key in our scheme is of the form $rk_{A\to B} = (kx_1, kx_2, ky_1, ky_2, k'x'_1, k'x'_2, k'y'_1, k'y'_2, z/z')$,any external adversary can not distinguish it with $sk_A = (x_1, x_2, y_1, y_2, z)$, $sk_B = (x'_1, x'_2, y'_1, y'_2, z')$, thus the adversary can not get any help information from re-Keygeneration Oracle.

We give our scheme's formal proof as following.Assume the external adversaries' algorithm B breaking the IND-CCA2 property of the scheme ,we use B to construct algorithm A distinguish a four tuple (g_1, g_2, u_1, u_2) from G is a DDH tuple or not. Oralce queries from B are handled by A as following:

- **Query to Key Generation Oracle:** If user i is corrupted ,A randomly chooses $sk_i = (x_{1i}, x_{2i}, y_{1i}, y_{2i}, z_i) \in Z_q$ at random,also chooses a hash function H at random and computes $pk_i = (g_1, g_2, c_i = g_1^{x_{1i}} g_2^{x_{2i}}, d_i = g_1^{y_{1i}} g_2^{y_{2i}}, h_i = g_1^{z_i}, H)$,return pk_i and sk_i to the adversary. If the user j is uncorrupted, A randomly chooses $sk_j = (x_{1j}, x_{2j}, y_{1j}, y_{2j}, z_{1j}, z_{2j}) \in Z_q$ at random, also chooses a hash function H at random and computes $pk_j = (g_1, g_2, c_j = g_1^{x_1} g_2^{x_2}, d_j = g_1^{y_1} g_2^{y_2}, h_j = g_1^{z_1} g_2^{z_2}, H)$,returns pk_j to the adversary.That is ,there is a $KeGen$ list of form$(corrupted, i, sk_i = (x_{1i}, x_{2i}, y_{1i}, y_{2i}, z_i)$, $pk_i = (g_1, g_2, c_i = g_1^{x_{1i}} g_2^{x_{2i}}, d_i = g_1^{y_{1i}} g_2^{y_{2i}}, h_i = g_1^{z_i}, H)$ or $(uncorrupted, j, sk_j = (x_{1j}, x_{2j}, y_{1j}, y_{2j}, z_{1j}, z_{2j}), pk_j = (g_1, g_2, c_j = g_1^{x_1} g_2^{x_2}, d_j = g_1^{y_1} g_2^{y_2}, h_j = g_1^{z_1} g_2^{z_2}, H)$.
- **Query to Re-Keygeneration Oracle:**On input (i, j) to $O_{rekeygen}$, if one of i and j is uncorrupted and the other is corrupted, then this call is illegal(So we just consider uncorrupted users in re-keygeneration , ciphertext-transformation , re-encryption and decryption oracle queries). Otherwise, A randomly choose $k_i, k_j, z \in Z_q^*$ and outputs the re-encryption key $rk_{i\to j} = (k_i x_{1i}, k_i x_{2i}, k_i y_{1i}, k_i y_{2i}, k_j x_{1j}, k_j x_{2j}, k_j y_{1j}, k_j y_{2j}, z)$. That is ,there is a $ReEnc$ $KeyGen$ list of form $(uncorrupted, i, k_i, uncorrupted, j, k_j, z)$
- **Query to Encryption Oracle:**. Given a message $m \in G$, the encryption algorithm runs as follows. First, it chooses $r \in Z_q$ at random. Then it computes $u_1 = g_1^r, u_2 = g_2^r, e = u_1^{z_1} u_2^{z_2} m, \alpha = H(u_1, u_2, e), v = c^r d^{r\alpha}$. The ciphertext is (u_1, u_2, e, v).
- **Query to Ciphertext-Transformation Oracle:**On input (u_1, u_2, e, v) from user i to user j,if user i and j are uncorrupted, B search in the $ReEncKeyGen$ list and if finding an item including i and j,then compute $v' = v^{k_i}$,and outputs (u_1, u_2, e, v'), else first run the querying to Re-Keygeneration Oracle.
- **Query to the Re-encryption Oracle:**On input(u_1, u_2, e, v') from user i to user j, search the $ReEncKeyGen$ list and if finding an item including i and j,then B first verifies ciphertext (u_1, u_2, e, v')'s validity. if $u_1^{k_i x_{1i} + k_i y_{1i}\alpha} u_2^{k_i x_{2i} + k_i y_{2i}\alpha} \neq v'$ where $\alpha = H(u_1, u_2, e)$, then return "Reject",else computes $u'_1 = {u_1}^z, u'_2 = {u_2}^z, e' = e, \alpha' = H(u'_1, u'_2, e'), v'' = {u_1}^{k_j x_{1j} z} {u_2}^{k_j x_{2j} z} {u_1}^{k_j y_{1j} \alpha' z} {u_2}^{k_j y_{2j} \alpha' z}$ output (u'_1, u'_2, e', v'').If not finding such item,first run the querying to Re-Keygeneration Oracle and querying to Ciphertext-Transformation Oracle.
- **Query to the Decryptionlevel_1 Oracle:**Level 1 cipertext is the re-encryption ciphertext. On input (u'_1, u'_2, e', v'') from user i to j , B first search $ReEncKeyGen$

list and if finding an item including i and j,then B first verifies ciphertext (u_1', u_2', e', v'')'s validity. if $u_1'^{k_j x_{1j}+k_j y_{1j}\alpha} u_2'^{k_j x_{2j}+k_j y_{2j}\alpha'} \neq v''$ where $\alpha' = H(u_1', u_2', e')$, then returns "Reject",else computes $m = e'/(u_1'^{z_1} u_2'^{z_2})^{1/z}$ output m.If not finding such items, first run the querying to Re-Keygeneration Oracle, querying to Ciphertext-Transformation Oracle and Re-Encryption Oracle.

- **Query to the Decryptionlevel_2 Oracle:** Level 2 ciphertext is the normal ciphertext.On input $c = (u_1, u_2, e, v)$ to user j, B first computes $\alpha = H(u_1, u_2, e)$, and search the $KeyGen$ list if finding an item including j and *uncorrupted* then tests if $u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha} = v$. If this condition does not hold, the decryption algorithm outputs "reject"; otherwise, it outputs $m = e/u_1^{z_1} u_2^{z_2}$, if finding an item including j and *corrupted*, run $dec_2(sk_j, c)$. Else first run qurey to Key Generation Oracle.

Next we show our oracle simulation is perfect.

- **Key Generation Oracle Simulation:**For corrupted users, the simulated output $(sk_i = (x_{1i}, x_{2i}, y_{1i}, y_{2i}, z_i), pk_i = (g_1, g_2, c_i = g_1^{x_{1i}} g_2^{x_{2i}}, d_i = g_1^{y_{1i}} g_2^{y_{2i}}, h_i = g_1^{z_i}, H)$ is an identical distribution to the real distribution of real private key and public key.For uncorrupted users,assuming $g_2 = g_1^w$,as the same technique used in Cramer-Shoup encryption,the simulated output$(sk_j = (x_{1j}, x_{2j}, y_{1j}, y_{2j}, z_{1j}, z_{2j}), pk_j = (g_1, g_2, c_j = g_1^{x_1} g_2^{x_2}, d_j = g_1^{y_1} g_2^{y_2}, h_j = g_1^{z_1} g_2^{z_2}, H)$ also is an identical distribution to the real distribution of real private key and public key for $h_j = g_1^{z_1} g_2^{z_2} = g_1^{z_1+wz_2} = g_1^{z'}$ .So this is a perfect simulation.
- **Re-Keygeneration Oracle Simulation:** When user i query the oracle with input i, j and users i or j is corrupted, we call this query illegal and output $\perp$.If users i and j are uncorrupted, the simulated output is $rk_{i\to j} = (k_i x_{1i}, k_i x_{2i}, k_i y_{1i}, k_i y_{2i}, k_j x_{1j}, k_j x_{2j}, k_j y_{1j}, k_j y_{2j}, z)$ which is indistinguishable with $rk_{A\to B} = (kx_1, kx_2, ky_1, ky_2, k'x_1', k'x_2', k'y_1', k'y_2', z/z')$.So this is also a perfect simulation.
- **Encryption Oracle Simulation:**.In the real encryption,Given a message $m \in G$, the encryption algorithm runs as follows. First, it chooses $r \in Z_q$ at random. Then it computes $u_1 = g_1^r, u_2 = g_2^r, e = h^r m, \alpha = H(u_1, u_2, e), v = c^r d^{r\alpha}$. The ciphertext is (u_1, u_2, e, v). In our simulation, Given a message $m \in G$, the encryption algorithm runs as follows. First, it chooses $r \in Z_q$ at random. Then it computes $u_1 = g_1^r, u_2 = g_2^r, e = u_1^{z_1} u_2^{z_2} m, \alpha = H(u_1, u_2, e), v = c^r d^{r\alpha}$. The ciphertext is (u_1, u_2, e, v).From Cramer-Shoup encryption,if (g_1, g_2, u_1, u_2) is a DDH tuple,the two ciphertexts can not be distinguished.So this is also a perfect simulation.
- **Ciphertext-Transformation Oracle Simulation:**In the real cipertext-transformation,On input initial ciphertext (u_1, u_2, e, v), the delegator transforms it to be (u_1, u_2, e, v') where $v' = v^k$.While in our oracle simulation,On input (u_1, u_2, e, v) ,if user i and user j are uncorrupted, it computes $v' = v^{k_i}$,and outputs (u_1, u_2, e, v^{k_i}),this is indistinguishable from the real outputs (u_1, u_2, e, v^k).If users i or j is corrupted,it output $\perp$. So this is also a perfect simulation.

- **Re-encryption Oracle Simulation:**In the real re-encryption,The proxy first verifies ciphertext (u_1, u_2, e, v')'s validity. if $u_1^{kx_1+ky_1\alpha} u_2^{kx_2+ky_2\alpha} \neq v'$ where $\alpha = H(u_1, u_2, e)$, then return "Reject",else computes $u_1' = {u_1}^{z/z'}, u_2' = {u_2}^{z/z'}, e' = e, \alpha' = H(u_1', u_2', e'), v'' = {u_1}^{k'x_1'z/z'} {u_2}^{k'x_2'z/z'} {u_1}^{k'y_1'\alpha'z/z'} {u_2}^{k'y_2'\alpha'z/z'}$. In our simulation, On input(u_1, u_2, e, v') B first verifies ciphertext (u_1, u_2, e, v')'s validity. if $u_1^{k_i x_{1i}+k_i y_{1i}\alpha} u_2^{k_i x_{2i}+k_i y_{2i}\alpha} \neq v'$ where $\alpha = H(u_1, u_2, e)$, then return "Reject",else computes $u_1' = {u_1}^z, u_2' = {u_2}^z, e' = e, \alpha' = H(u_1', u_2', e'), v'' = {u_1}^{k_j x_{1j} z} {u_2}^{k_j x_{2j} z} {u_1}^{k_j y_{1j} \alpha' z} {u_2}^{k_j y_{2j} \alpha' z}$ output (u_1', u_2', e', v''). In lemma l, we will show anybody cannot construct valid (u_1, u_2, e, v') by himself with (g_1, g_2, u_1, u_2) being not a DDH tuple, as the same technique used in Cramer-Shoup Encryption. Thus the real output and simulated output are indistinguishable.So this is also a perfect simulation.
- **Decryptionlevel$_1$ Oracle Simulation:**In the real Decryption Dec_1,Given a re-encryption ciphertext (u_1', u_2', e', v''), the decryption algorithm runs as follows. It first computes $\alpha' = H(u_1', u_2', e')$, and tests if $(u_1')^{k'(x_1'+y_1'\alpha')} (u_2')^{k'(x_2'+y_2'\alpha')} = v''$. If this condition does not hold, the decryption algorithm outputs "reject"; otherwise, it outputs $m = e'/(u_1')^{z'}$. In our simulation, On input (u_1', u_2', e', v'') from user i to j ,B B first verifies ciphertext (u_1', u_2', e', v'')'s validity. if $u_1'^{k_j x_{1j}+k_j y_{1j}\alpha} u_2'^{k_j x_{2j}+k_j y_{2j}\alpha'} \neq v''$ where $\alpha' = H(u_1', u_2', e')$, then returns "Reject",else computes $m = e'/(u_1'^{z_1} u_2'^{z_2})^{1/z}$ output m.As in Cramer-Shoup encryption, if (g_1, g_2, u_1, u_2) is a DDH tuple.Our simulated decryption is a perfect decryption.In lemma l, we will show anybody cannot construct valid (u_1', u_2', e', v'') by himself with (g_1, g_2, u_1, u_2) being not a DDH tuple.Thus ,anybody can not distinguish the real decryption and the simulated decryption.So this is also a perfect simulation.
- **Decryptionlevel$_2$ Oracle Simulation:**In the real Decryption Dec_2,a normal ciphertext (u_1, u_2, e, v), the decryption algorithm runs as follows. It first computes $\alpha = H(u_1, u_2, e)$, and tests if $(u_1)^{(x_1+y_1\alpha)} (u_2)^{(x_2+y_2\alpha)} = v$. If this condition does not hold, the decryption algorithm outputs "reject"; otherwise, it outputs $m = e/(u_1)^z$.In our simulation,On input $c = (u_1, u_2, e, v)$ to user j, B first computes $\alpha = H(u_1, u_2, e)$, and search the $KeyGen$ list if finding an item including j and *uncorrupted* then tests if $u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha} = v$. If this condition does not hold, the decryption algorithm outputs "reject"; otherwise, it outputs $m = e/u_1^{z_1} u_2^{z_2}$as in Cramer-Shoup encryption, if (g_1, g_2, u_1, u_2) is a DDH tuple.Our simulated decryption is a perfect decryption.In lemma l, we will show anybody cannot construct valid (u_1, u_2, e, v) by himself with (g_1, g_2, u_1, u_2) being not a DDH tuple.Thus ,anybody can not distinguish the real decryption and the simulated decryption.So this is also a perfect simulation.

Lemma 1. *If (g_1, g_2, u_1, u_2) is not a DDH tuple, the decryption oracle(includes dec_1,dec_2,Decryptionlevel$_1$ Oracle and Decryptionlevel$_2$ Oracle) will reject all invalid ciphertexts, except with negligible probability.*

Proof. The proof of this lemma is same as [CH98],the only difference is that in the dec_1 and Decryptionlevel$_1$ Oracle simulation, the adversary must solve the

first three equations

$$x_1 + wx_2 = log^c_{g_1} modq \quad (1)$$

$$y_1 + wy_2 = log^d_{g_1} modq \quad (2)$$

$$kr_1x_1 + kr_1\alpha y_1 + kwr_2x_2 + kwr_2\alpha y_2 = log^V_{g_1} modq \quad (3)$$

$$x_1 + wx_2 = log^c_{g_1} modq \quad (4)$$

$$y_1 + wy_2 = log^d_{g_1} modq \quad (5)$$

$$r_1x_1 + r_1\alpha y_1 + wr_2x_2 + wr_2\alpha y_2 = log^V_{g_1} modq \quad (6)$$

while in the dec_1 and Decryptionlevel$_1$ Oracle simulation, the adversary must solve the latter three equations The above two equations are all have q solvations, thus the adversary can guess the right v is negligible.For full proof readers may refereed [CH98].

Thus, our simulation is perfect for the external adversary, if A can break our re-encryption scheme, B can solve the DDH problem in G.Thus we prove our theorem.

Theorem 2. *Our re-encryption scheme is secure against adaptive chosen plaintext attack for the proxy assuming that the DL problem is hard in the group G.*

Proof. For the proxy, our scheme just like a Elgamal re-encryption scheme proposed by [BBS98].We know that scheme is CPA secure, so our scheme is CPA secure for the proxy.

5.3 Efficiency

We compare our scheme's efficiency with other CCA2 re-encryption schemes in table 4. We know from [B06]that one pairing almost equals six exponentiation, so our scheme is the most efficiency until now which can achieve CCA2 secure.

Scheme	**Security**	**Re-enc key be protected**	**Checking**	**Re-encryption**
CH07	CCA2	Yes	1verifying and 4pairing	1exp
LV08a	CCA2	No	1verifying and 2pairing	3pairing
KR08*	WCCA2	Yes	No	1exp
Ours	CCA2	Yes	1exp	5exp

* This is a RSA-TBOS signcryption with proxy re-encryption scheme,but this scheme have not the process checking inital ciphertext,any one can construct invalid ciphertext for DDos attacking,so it is not efficient for resisting DDos attacking.

Table 4. Efficiency comparison between our scheme and other CCA2 secure schemes

6 Concluding Remarks

In this paper, we introduce a new attack- DDos attack against proxy in the proxy re-cryptography. Although this attack can also be implemented against other cryptographic primitives, the danger caused by it in proxy re-cryptography seems more serious. We revisit the current literature, paying attention on their resisting DDos attack ability. We suggest a solution to decline the impact of DDos attacking. Also we give a new efficient re-encryption scheme which can achieve CCA2 secure based on Cramer-Shoup encryption scheme and prove its security. We point out this is the most efficient proxy re-encryption schemes for the proxy which can achieve CCA2 secure in the literature until now.But we also note that our scheme is a bidirectional scheme with private re-encryption key, which maybe restrict its application. Finding a unidirectional re-encryption scheme with public re-encryption key and other good properties defined in [AFGH05] is still an open problem.

References

[CH98] R.Cramer and V.Shoup.A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack.In *Advances in Cryptology - Crypto'98,* LNCS 1462,pp. 13–25.Springer–Verlag,1998.

[BBS98] M. Blaze, G. Bleumer, and M. Strauss, Divertible Protocols and Atomic Proxy Cryptography.In *Advances in Cryptology - Eurocrypt'98,* LNCS 1403, pp. 127–144.Springer–Verlag,1998.

[J99] Markus Jakobsson. On quorum controlled asymmetric proxy re-encryption.In *, PKC '99*,pages 112–121,Springer–Verlag,1999.

[DI03] Yevgeniy Dodis and Anca-Andreea Ivan. Proxy cryptography revisited.In *NDSS '03*,2003.

[ZMSR04] Lidong Zhou, Michael A. Marsh, Fred B. Schneider, and Anna Redz. Distributed blinding for ElGamal re-encryption.TR 1924, Cornell CS Dept.,2004.

[AFGH05] G.Ateniese, K. Fu, M. Green, and S. Hohenberger, Improved proxy re-encryption schemes with applications to secure distributed storage.In *ACM Trans. Inf. Syst. Secur. 9 (2006), no. 1, pages 1–30.*

[AH06] G. Ateniese, S. Hohenberger, Proxy re-signatures: new definitions, algorithms, and applications.In *ACM CCS'05*, pp. 310–319.ACM Press,2005.

[H06] S. Hohenberger. Advances in Signatures, Encryption, and E-Cash from Bilinear Groups. Ph.D. Thesis, MIT, May 2006.

[P1363.3/06] Eu-Jin Goh,and Toshihiko Matsuo. Proposal for P1363.3 Proxy Re-encryption.http://grouper.ieee.org/groups/1363/IBC/submissions/NTTDataProposal-for-P1363. 3-2006-08-14.pdf and NTTDataProposal-for-P1363. 3-2006-09-01.pdf.

[B06] M.Barbosa,L.Chen,Z.Cheng et al.SK−KEM:An Identity−based Kem. http://grouper.ieee.org/groups/1363/IBC/submissions/Barbosa-SK-KEM-2006-06.pdf.

[CH07] R. Canetti and S. Hohenberger, Chosen Ciphertext Secure Proxy Re-encryption.In *In Proceedings of the 14th ACM conference on Computer and Communications Security(CCS 2007)*,pp. 185–194.2007. Also available at Cryptology ePrint Archive: http://eprint.iacr.org/2007/171.pdf.

[HRSV07] S. Hohenberger, G. N. Rothblum, a. shelat, V. Vaikuntanathan.B Securely Obfuscating Re-encryption.In *TCC'07,*LNCS 4392, pp. 233–252.Springer–Verlag,2007.

[SCWL07] J. Shao,Z. Cao,L. Wang,X. Liang. Proxy Re-Signature Schemes without Random Oracles.In *Indocrypt'07,*LNCS 4859, pp. 197–209.Springer–Verlag,2007.

[GA07] M. Green and G. Ateniese, Identity-Based Proxy Re-encryption. In *Applied Cryptography and Network Security'07,*LNCS 4521, pp. 288–306.Springer–Verlag,2007.

[M07] T. Matsuo, Proxy Re-encryption Systems for Identity-Based Encryption.In *, First International Conference on Pairing-Based Cryptography - Pairing 2007,*LNCS 4575, pp. 247–267.Springer–Verlag,2007.

[CT07] C.Chu and W.Tzeng. Identity-based proxy re-encryption without random oracles. In *ISC 2007,*LNCS 4779, pp. 189–202.Springer–Verlag,2007.

[SXC08] J.Shao,D.Xing and Z.Cao, Identity-Based Proxy Rencryption Schemes with Multiuse, Unidirection, and CCA Security.Cryptography ePrent Achieve.http://eprint.iacr.org/2008/103.pdf,2008.

[KR08] Varad Kirtane, C.Pandu Rangan, RSA-TBOS Signcryption with Proxy Re-encryption.http://eprint.iacr.org/2008/324.pdf,2008.

[CAP08] ChandrasekarS.,Ambika K.,Pandu Rangan C.Signcryption with Proxy Re-encryption.http://eprint.iacr.org/2008/276.pdf,2008.

[MAL07] C.Ma ,J.Ao,and J.Li.Group-oriented encryption secure against collude attack.http://eprint.iacr.org/2007/371.pdf,2007.

[MA08] C.Ma,J.Ao.Revisit of Group-based Unidirectional Proxy Re-encryption Scheme.http://eprint.iacr.org/2008/325.pdf,2008.

[LV08a] B. Libert and D. Vergnaud, Unidirectional Chosen-Ciphertext Secure Proxy Re-encryption.In *11th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2008,*LNCS 4939, pp. 360–379.Springer–Verlag,2008.

[LV08b] B. Libert and D. Vergnaud, Tracing Malicious Proxies in Proxy Re-Encryption.In *, First International Conference on Pairing-Based Cryptography - Pairing 2008*,Springer–Verlag,2008.

[LV08c] B. Libert and D. Vergnaud, Multi-Use Unidirectional Proxy Re-Signatures.In *CCS08*,ACM Press,2008.

A Revisit Re-encryption Schemes in the Literature

A.1 BBS98 Schem

This Elgamal based scheme operates with a safe prime modulus $p = 2q + 1$,

1. **Key Generation** (KG)**:** A user A's key pair is of the form $pk_a = g^a, sk_a = a$.
2. **Re-Encryption Key Generation** (RG)**:** A user A delegates to B by publishing the re-encryption key $b/a \bmod q$.
3. **Re-Encryption** (R)**:** the proxy is entrusted with the delegation key $b/a \bmod q$ for the purpose of diverting ciphertexts from Alice to Bob via computing $(mg^k \bmod p, (g^{ak})^{b/a} \bmod p)$.

4. **Decryption** (D_1, D_2)**:** To decrypt a ciphertext $c_a = (\alpha, \beta)$ with secret key $sk = a$, compute $m = \beta/\alpha^{1/a}$.

In this scheme the delegation key must be protected well, otherwise the scheme is definitely not secure. The workload of the proxy can be divided into two parts, one part is validating the initial ciphertext,this scheme does not require this;the other part is the re-encryption operation, this scheme just needs an exponentiation.

But from the other **Condition of Nine Properties**[AFGH05],this scheme is not collusion-"safe" ,non-transitive, non-transferable and unidirectional,non-interactive, that is ,the scheme is not secure and good.

A.2 CH07 Scheme

Let 1^k be the security parameter and (q, g, h, G, G_T, e) be the bilinear map parameters output by $BSetup(1^k)$. Let $Sig = (G, S, V)$ be a strongly unforgeable one-time signature scheme, where $l = l_{sig}(k)$denotes the length of the verification keys output by $G(1^k)$. Moreover, we assume that the verification key space produced by G has super-logarithmic minimum entropy; that is, any given key has a negligible chance of being sampled. Let $H : \{0,1\}^l \rightarrow G$ and $F : \{0,1\}^l \rightarrow G$ be two independent hash functions, which we will treat as random oracles.

Define the algorithm $Check$ on input a ciphertext tuple (A, B, C, D, E, S) and a key pk as follows:

1. Run $\upsilon(A, (C, D, E), S)$ to verify signature S on message (C, D, E) with respect to key A.
2. Check that $e(B, F(A)) = e(pk, D)$ and that $e(B, h) = e(pk, E)$.
3. If any of these checks fail, output 0; else output 1.

The scheme is described as follows:

1. **Key Generation (KeyGen):**On input 1^k, select random $x \in Z_q$. Set $pk = g^x$ and $sk = x$.
2. **Re-Encryption Key Generation(ReKeyGen):** On input $sk_X = x$ and $sk_Y = y$, output the bidirectional re-encryption key $rk_{X \leftrightarrow Y} = x/y \bmod q$.
3. **Encryption (Enc):** On input pk and a message $m \in G_T$:
   (a) Select a one-time signature keypair as $G(1^k) \rightarrow (svk, ssk)$. Set $A = svk$.
   (b) Select a random $r \in Z_q$ and compute $B = pk^r$, $C = e(g, H(svk))^r.m$, $D = F(svk)^r, E = h^r$.
   (c) Run the signing algorithm $S(ssk, (C, D, E))$, where the message to sign is the tuple (C, D, E), and denote the signature S.
   (d) Output the ciphertext (A, B, C, D, E, S).
4. **Re-Encryption (ReEnc):** On input a re-encryption key $rk_{X/Y} = x/y$ and a ciphertext $K = (A, B, C, D, E, S)$ under key pk_Y, re-encrypt the ciphertext to be under key pk_X as:
   (a) Compute $B' = B^{rk_{X \leftrightarrow Y}} = g^{(yr)(x/y)} = g^{xr}$.

(b) If Check$(K, pk_Y) = 1$, output the new ciphertext (A, B', C, D, E, S); otherwise, output $\perp$.

5. **Decryption (Dec):** On input a secret key sk and any ciphertext $K = (A, B, C, D, E, S)$, if Check$(K, g^{sk}) = 1$, then output the message $C/e(B, H(A))^{1/sk}$; otherwise, output $\perp$.

In this scheme, the delegation key needs to be private and protected well. For the initial ciphertext checking we need one computation of verifying signature and four pairing computation.In the re-encryption process, we need an exponentiation.But this scheme is CCA2 secure.

A.3 LV08a Scheme

Given a security parameter 1^k, choose bilinear map groups (G, G_T) of prime order $p > 2$, generators $g, u, v \in G$ and a strongly unforgeable one-time signature scheme $Sig = (G, S, V)$. The global parameters are $par := \{G, G_T, g, u, v, Sig\}$.

1. **Keygen(1^k):** user i sets his public key as $X_i = g^{x_i}$ for a random $x_i \in Z_p^*$.
2. **ReKeygen(x_i, X_j):**given user i's private key x_i and user j's public key X_j, generate the unidirectional re-encryption key $R_{ij} = X_j^{1/x_i} = g^{x_j/x_i}$.
3. **Enc$_1$(m, $\mathbf{X}_i$, par):** to encrypt a message $m \in GT$ under the public key X_i at the first level, the sender proceeds as follows.
   (a) Select a one-time signature key pair $(ssk, svk) \in G(1^k)$ and set $C_1 = svk$.
   (b) Pick $r, t \in Z_p^*$ and compute $C_2^{'} = X_i^t$,$C_2^{''} = g^{1/t}$ $C2 = X_i^{rt}$ $C_3 = e(g, g)^r.m$ $C_4 = (u_s vkv)^r$
   (c) Generate a one-time signature $= S(ssk, (C_3, C_4))$ on (C_3, C_4).
   The ciphertext is $C_i = (C_1, C_2^{'}, C_2^{''}, C_2^{'''}, C_3, C_4,)$.
4. **Enc$_2$(m, $\mathbf{X}_i$, par)**: to encrypt a message $m \in G_T$under the public key X_i at level 2, the sender conducts the following steps.
   (a) Select a one-time signature key pair $(ssk, svk) \in G()$ and set $C_1 = svk$.
   (b) Choose $r \in RZ_p^*$ and compute $C_2 = X_i^r$ $C_3 = e(g, g)^r m$ $C_4 = (u_{svk} v)^r$.
   (c) Generate a one-time signature $\sigma = S(ssk, (C_3, C_4))$ on the pair (C_3, C_4).
   The ciphertext is $C_i = (C_1, C_2, C_3, C_4, \sigma)$.
5. **ReEnc**(R_{ij}, C_i)**:** on input of the re-encryption key $R_{ij} = g^{x_j/x_i}$ and a ciphertext $C_i = (C_1, C_2, C_3, C_4,)$, check the validity of the latter by testing the following conditions $e(C_2, u^{C_1}.v) = e(X_i, C_4)$ $V(C_1, \sigma, (C_3, C_4)) = 1$. If well-formed, C_i is re-encrypted by choosing $t \in Z_p^*$ and computing $C_2^{'} = X_i^t$ $C_2^{''} = R_{ij}^{1/t} = g^{(x_j/x_i)^{t-1}}$ $C_2^{'''} = C_2^t = X_i^{rt}$ The re-encrypted ciphertext is $(C_j = C_1, C_2^{'}, C_2^{''}, C_2^{'''}, C_3, C_4, \sigma)$.If ill-formed, C_i is declared 'invalid'.
6. **Dec$_1$($\mathbf{C}_j$, sk_j)**:the validity of a level 1 ciphertext C_j is checked by testing if $e(C_2^{'}, C_2^{''}) = e(X_j, g)$ $e(C_2^{'''}, u^{C_1}.v) = e(C_2^{'}, C_4)$ $V(C_1, \sigma, (C_3, C_4)) = 1$.If above relations hold, the plaintext $m = C_3/e(C_2^{''}, C_2^{'''})^{1/x_j}$ is returned. Otherwise, the algorithm outputs 'invalid'.
7. **Dec$_2$($\mathbf{C}_i$, sk_i)**:if the level 2 ciphertext $C_i = (C_1, C_2, C_3, C_4, \sigma)$ satisfies valid ciphertext relations, receiver i can obtain $m = C_3/e(C_2, g)^{1/x_i}$. The algorithm outputs 'invalid' otherwise.

In this scheme, the delegation key need not be private, the initial ciphertext checking need one computation of verifying the signature and two pairing computation. The re-encryption needs three exponentiation.Also this scheme has the property of containing two-level ciphertexts, some additional computation must be operated.This scheme is CCA2 secure.

A.4 GA06 scheme

This scheme is based on the Gentry-Silverberg HIBE and making use of technique of CHK transformation from CPA to CCA, the transformation using the BLS short signature.

1. **Setup.** Let n be polynomial in the security parameter k. Let $e : G_1 \times G_1 \rightarrow G_T$ be a bilinear map, where G_1, G_T have order q and $G_1 = \langle g \rangle$. To generate the scheme parameters, select $s \in Z_q^*$ and output $params = (H_1, H_2, H_3, H_4, H_5, H_6, g, g^s)$, $msk = s$, with independent hash functions H_{1-6} defined as below: $H_1 : \{0,1\}^* \rightarrow G_1$,$H_2 : \{0,1\}^* \rightarrow G_1$,$H_3 : \{0,1\}^* \rightarrow G_1$,$H_4 : G_T \times \{0,1\}^n \rightarrow Z_q^*$, $H_5 : G_T \rightarrow \{0,1\}^n$
2. **KeyGen**$(params, msk, id)$. To extract a decryption key for identity $id \in \{0,1\}^*$, return $sk_{id} = H_1(id)^s$.
3. **Encrypt**$(params, id, m \in \{0,1\}^n)$. To encrypt m under identity $id \in \{0,1\}^*$:
   (a) Select $\in G_T$, and set $r = H_4(, m)$.
   (b) Compute $c' = (g^r, \times e(g^s, H_1(id)^r)$,$m \oplus H_5())$.
   (c) Compute $S = H_3(id \parallel c')^r$.
   (d) Output the ciphertext $c = (S, c')$.
4. **RKGen(**$params, sk_{id_1}, id_1, id_2)$. To compute a re-encryption key from $id_1 \rightarrow id_2$:
   (a) Select $N \leftarrow \{0,1\}^n$, and compute $K = e(sk_{id_1}, H_1(id_2))$.
   (b) Output $rk_{id_1 \rightarrow id_2} = (N, H_2(K \parallel id_1 \parallel id_2 \parallel N) \times sk_{id_1}$.
5. **Reencrypt**$(params, rk_{id_1 \rightarrow id_2}, c_{id_1})$. To re-encrypt a first-level ciphertext, first parse c_{id_1} as (S, A, B, C), and parse $rk_{id_1 \rightarrow id_2}$ as (N, R). Next:
   (a) Let $h = H_3(id_1 \mid (A, B, C))$.
   (b) Check if $e(g, S) = e(h, A)$. If not, return $\perp$.
   (c) Otherwise, select $t \in Z_q^*$ and compute $B' = B'/\frac{e(A, Rh')}{e(g^t, S)}$.
   (d) Output the re-encrypted ciphertext $c_{id_2} = (A, B', C, id_1, N)$.
6. **Decrypt**$(params, sk_{id}, c_{id})$. To decrypt a first-level (non re-encrypted) ciphertext, first parse $c_i d$ as (S, A, B, C). Next:
   (a) Let $h = H_3(id, h_A, B, C)$.
   (b) Select $t \in Z_q^*$, and compute $' = B/\frac{e(A, sk_{id} \times h')}{e(g^t, S)}$.
   (c) Compute $m' = C \oplus H_5(')$, an d $r' = H_4(', m')$.
   (d) Verify that $S = h^r$and $A = g^r$. If either check fails, return $\perp$, otherwise output m'.

   To decrypt a second-level (re-encrypted) ciphertext, first parse c_{id} as (A, B, C, id_{src}, N). Next:
   (a) Compute $K = e(H_1(id_{src}), sk_{id})$.
   (b) Compute $= B \times e(A, H_2(K \mid id_{src} \parallel id \parallel N))$.

(c) Compute $m' = C \oplus H_5(')$, and $r' = H_4(, m')$.
(d) Verify that $A = g^{r'}$. If this check fails, return $\perp$, otherwise output m'.

In this scheme, the proxy key need not to be private, checking the initial ciphertext requiring two pairing computation and re-encryption requiring two pairing computation.Also this scheme is a two-level ciphertext, some additional computation must be operated .This scheme is CCA2 secure.