Non-Transferable Proxy Re-Encryption Scheme

Yi-Jun He, Tat Wing Chim, Lucas Chi Kwong Hui, Siu-Ming Yiu
Department of Computer Science,
The University of Hong Kong, Hong Kong
{*yjhe, twchim, hui, smyiu*}*@cs.hku.hk*

***Abstract*—A proxy re-encryption (PRE) scheme allows a proxy to re-encrypt a ciphertext for Alice (delegator) to a ciphertext for Bob (delegatee) without seeing the underlying plaintext. However, existing PRE schemes generally suffer from at least one of the followings. Some schemes fail to provide the *non-transferable* property in which the proxy and the delegatee can collude to further delegate the decryption right to anyone. This is the main open problem left for PRE schemes. Other schemes assume the existence of a fully trusted private key generator (PKG) to generate the re-encryption key to be used by the proxy for re-encrypting a given ciphertext for a target delegatee. But this poses two problems in PRE schemes if the PKG is malicious: the PKG in their schemes may decrypt both original ciphertexts and re-encrypted ciphertexts (referred as the *key escrow* problem); and the PKG can generate re-encryption key for arbitrary delegatees without permission from the delegator (we refer to it as the *PKG despotism* problem).**

In this paper, we propose the first non-transferable proxy re-encryption scheme which successfully achieves the *non-transferable* property. We show that the new scheme solved the *PKG despotism* problem and *key escrow* problem as well.

***Keywords*-proxy re-encryption; PKG despotism; non-transferable property**

I. Introduction

A. Proxy Re-encryption

The proxy re-encryption [2] allows a third-party (the proxy) to re-encrypt a ciphertext which has been encrypted for one party without seeing the underlying plaintext so that it can be decrypted by another. For example: Alice keeps some sensitive files in encrypted form in the file server; Bob fetches encrypted files from file server, and then transmits them to proxy; Alice sends a re-encryption key to the proxy which re-encrypts the encrypted files and sends Bob the re-encrypted ciphertext which can be decrypted by Bob with his own private key. This scheme aroused much interest in the encryption community [1], [2], [6], [7], [8], [9], [10], [11], [12], [13], [14] since it could be exploited in a number of applications for achieving better information security and privacy, such as: email forwarding, encrypted files distribution, law-enforcement monitoring, *etc*..

B. Review of the Transferable Problem

A proxy re-encryption scheme is said to be non-transferable [1] if the proxy and a set of colluding delegatees cannot re-delegate decryption rights to other parties. On one hand, this is a very desirable property. For example, user A saves some encrypted private confidential files on the file server. If A delegates B the decryption right for accessing those files, A may need some guarantee that his files "go no further". It requires that the delegatee B plus the proxy cannot re-delegate decryption right to others. On the other hand, researchers [1], [7] are even not sure that transferability can be preventable since the delegatee B can always decrypt and forward the plaintext to another party. However, this approach requires that the delegatee remains an active, online participant. What we want to prevent is the delegatee (plus the proxy) providing other parties with a secret value that can be used offline to decrypt A's ciphertexts. Again, the delegatee can always send its secret key to another party. But in doing so, the delegatee puts itself in a risky situation. Therefore, achieving a non-transferable PRE scheme, in the sense that the only way for delegatee to transfer decryption capabilities to another party is to expose his own secret key, seems to be the main open problem left for PRE.

C. Limitations of Existing Solutions

Libert and Vergnaud [7] indicated that it is quite difficult to prevent the proxy and delegatees from colluding to do re-delegation and that discouraging collusion rather than preventing illegitimate re-delegation is an easier approach. Thus, they try to trace the malicious proxy after its collusion with one or more delegatees. No doubt that it works to deter collusion from happening. However, it is more desirable to have a better way to prevent collusion, not just discourage collusion. Some identity-based PRE schemes [8], [10], [11], [12], [13], [14] assume the existence of a fully trusted private key generator (PKG) which helps to generate the re-encryption key to be used by the proxy for re-encrypting a given ciphertext for a target delegatee. Since the re-encryption key is generated using the master key of the PKG, the proxy and the delegatee(s) cannot further delegate the decryption right to others without the help of the PKG. However, this creates two problems in PRE schemes. First, there is another key escrow problem, in which the PKG in their schemes may be able to decrypt both original and re-encrypted ciphertexts; And the PKG despotism problem, in which the PKG itself has the power of generating a re-encryption key for transferring decryption right to ar-

978-1-4673-0229-6/12/$31.00 ©2012 IEEE

bitrary delegatees. Thus those PKG-based PRE schemes just transformed the "delegatee-proxy-collusion transferable problem" to a "PKG alone transferable problem". So it is fair to say that they did not solve the transferable problem. Recently, Hayashi *et al*. [4] tried to achieve the non-transferability by reducing the non-transferable property to a relaxed notion "the unforgeability of re-encryption keys against collusion attack (UFReKey-CA)", which meas that proxies and delegatees cannot collude to generate a re-encryption key for some user, but as pointed out in their paper, they have not succeeded in constructing a scheme that meets the non-transferability, because the malicious user can extract the plaintext even without the re-encryption key.

D. Our Contributions

- The proposed scheme has the non-transferable property. The re-encryption key is generated by a key generating centre (PKG); Delegator participants actively to help generating partial decryption key for delegatee using part of his private key. Thus delegatee and proxy cannot collude to re-delegate decryption rights since they do not have knowledge of PKG's master secret or the delegator's private key.
- Without the participation of the delegator, PKG is unable to generate any useful re-encryption key for delegating decryption right, thus completely resolves the PKG despotism problem.
- PKG cannot decrypt the original ciphertext and re-encrypted ciphertexts as well, thus solving the key escrow problem.

II. Our Non-Transferable PRE Scheme

We construct the Non-Transferable PRE scheme based on the basic IBE system proposed in [3]. The main ideas of the scheme are as follow: Before delegation, delegator will send delegatee's identity to PKG. PKG is responsible for generating the re-encryption key, and sending this key and some other information to delegator. Delegator checks the correctness of the re-encryption key, and generates a partial decryption key making use of the information received from PKG. Then, delegator sends the re-encryption key to the proxy, and the partial decryption key to delegatee. The proxy re-encrypts the original ciphertext from delegator, and sends the re-encrypted ciphertext to delegatee. The delegatee can decrypt the ciphertext using his private key and the partial decryption key received from delegator.

In the following sections, we let Alice (*A*) be the delegator, and Bob (*B*) be the delegatee.

Setup:

Let G and G_T be groups of order p such that p is a *k*-bit prime, and let $e : G \times G \rightarrow G_T$ be the bilinear map. H_I:$\{0,1\}^* \rightarrow \mathrm{Z}_p$, H:$\{0,1\}^* \rightarrow \mathrm{Z}_p$, H':$G_T \rightarrow \mathrm{Z}_p$ are secure hash functions. The PKG selects four random generators $h_1, h_2, h_3, g \in G$ and randomly chooses $\alpha \in Z_p$. It sets $g_1 = g^{\alpha}$. Define the message space $\mathcal{M} \in G_T$. The public parameters *mpk* and master secret key *msk* are given by $mpk = (g, g_1, h_1, h_2, h_3, H_I, H, H', \mathcal{M})$, $msk = (\alpha)$.

Key Generation:

This is a protocol through which a user *U* with an identity *ID* can securely get his partial private key from PKG.

On input the public key/master secret key pair (*mpk*, *msk*) and an identity $ID_A \in \{0,1\}^k$ of entity *A*, the PKG computes $id_A = H_I(ID_A)$. If $id_A = \alpha$, it aborts. Otherwise, the protocol proceeds as follow:

- Set-Secret-Value. Entity *A* selects $r_A \in Z_p$ at random. r_A is *A*'s secret value.
- Partial-Private-Key-Extract.
  1) *A* sends $R = {h_1}^{r_A}$ to PKG, and gives PKG the following zero-knowledge proof of knowledge:
  $$PK\{r_A : R = {h_1}^{r_A}\}$$
  2) PKG randomly selects r'_A, $r_{A,2}$, $r_{A,3} \in Z_p$ and computes
  h'_A=$(Rg^{-r'_A})^{1/(\alpha - id_A)}$,
  $h_{A,2}$=$(h_2 g^{-r_{A,2}})^{1/(\alpha - id_A)}$,
  $h_{A,3}$=$(h_3 g^{-r_{A,3}})^{1/(\alpha - id_A)}$
  and sends *A*'s partial private key (r'_A, h'_A, $r_{A,2}$, $h_{A,2}$, $r_{A,3}$, $h_{A,3}$) to *A*.
- Set-Private-Key. *A* computes
  $$r_{A,1} = r'_A / r_A, h_{A,1} = (h'_A)^{1/r_A} = (h_1 g^{-r_{A,1}})^{1/(\alpha - id_A)}$$
  Then, *A*'s private key can be denoted as
  $$usk_A = (r_A, r_{A,1}, h_{A,1}, r_{A,2}, h_{A,2}, r_{A,3}, h_{A,3})$$
  Similarly, the delegatee *B*'s private key is denoted as
  $$usk_B = (r_B, r_{B,1}, h_{B,1}, r_{B,2}, h_{B,2}, r_{B,3}, h_{B,3})$$
- Set-Public-Key. *A* publishes her public key $upk_A = (p_{A,1}, p_{A,2})$, where $p_{A,1} = {g_1}^{r_A}$, and $p_{A,2} = g^{r_A id_A}$. Anyone can verify the validity of upk_A by checking if the equality $e(g^{id_A}, p_{A,1})$=$e(g_1, p_{A,2})$ holds.

Private Key Correctness Check:

On input (mpk, usk_{ID}) and an identity $ID \in \{0,1\}^k$, *A* computes $id_A = H_I(ID_A)$ and checks whether
$$e(h_{A,i}, g_1 / g^{id_A}) = e(h_i g^{-r_{A,i}}, g)$$
for *i*=1,2,3. If correct, output 1. Otherwise, output 0.

Encryption:

To encrypt a message $m \in G_T$ using public key, sender checks that whether the equality $e(g^{id_A}, p_{A,1})$=$e(g_1, p_{A,2})$ holds. If not, output $\perp$ and abort encryption. Otherwise, sender generates a unique randomly-selected secret parameter $s \in Z_p$, and computes $id_A = H_I(ID_A)$. Finally, sender outputs the ciphertext *C* where:
$C = (C_1, C_2, C_3, C_4, C_5, C_6)$ =$({p_{A,1}}^s {p_{A,2}}^{-s}$, $e(g,g)^s$, $m \cdot e(g,h_1)^{-s}$, $e(g,g)^{H'(m)}$, $g^{s\beta + H'(m)}$, $e(g,h_2)^s e(g,h_3)^{s\beta})$.

We set $\beta = H(C_1, C_2, C_3, C_4)$.

Decryption(delegator):

To decrypt a ciphertext $C = (C_1, C_2, C_3, C_4, C_5, C_6)$ using secret key usk_A, delegator *A* computes $\beta = H(C_1, C_2, C_3, C_4)$ and tests whether $e(C_5, g) = C_2^{\beta} C_4$ and $C_6 = e(C_1, h_{A,2} h_{A,3}{}^{\beta})^{1/r_A} \cdot C_2{}^{r_{A,2}+r_{A,3}\beta}$. If it is not equal, outputs $\perp$. Else *A* computes $m = C_3 \cdot e\,(C_1, h_{A,1})^{1/r_A} \cdot C_2{}^{r_{A,1}}$. If $e(g,g)^{H'(m)} = C_4$ holds, return *m*; otherwise return $\perp$.

The following Re-Encryption process is done through an interactive protocol among delegator *A*, delegatee *B*, PKG and Proxy.

Re-Encryption Key Generation:

1) In our PRE scheme, *B* is only allowed to decrypt messages intended for *A* during some specific time period *i*. To achieve this property, *A* generates a random value $a_i \in Z_p$ for each time period *i*, where $i \geq 1$. a_i will be invalid after the period *i*. *A* signs *B*'s identity ID_B, and sends the signature σ, ID_B, a_i to PKG via a secure channel.
   Delegator Sign:
   - Choose $z \in Z_p$, and compute $U = g^z$.
   - Compute $V = H_I\,(ID_B, U)$.
   - Compute $W = g^{\alpha r_A + V}$.
   - The signature on ID_B is $\sigma = (U, W)$.
2) PKG verifies the signature.
   PKG Verify:
   - Compute $V = H_I\,(ID_B, U)$.
   - Accept the signature iff $e(h_1, W) = e(h_1{}^{r_A}, g^{\alpha}) e(h_1, g)^V$.
3) If verification passes, PKG generates a unique randomly-selected secret parameter $y \in Z_p$, and computes re-encryption key $rk_{A \to B} = (\frac{\alpha - id_B}{\alpha - id_A} + a_i y) \bmod p, A_1 = \left(h_1{}^{r_A} g^{-r'_A}\right)^{y}, B_1 = \left(h_1{}^{r_B} g^{-r'_B}\right)^{a_i y/(\alpha - id_B)}, B_2 = h_1{}^{a_i y}$ and sends $rk_{A \to B}$, A_1, B_1, B_2 to *A*.

Partial-Decryption-Key Generation:

1) Delegatee *B* sends h'_B to *A* via a secure and authenticated channel.
2) *A* checks whether $e(h_1, B_1) = e(B_2, h'_B)$ to ensure B_1 is a valid value which will help delegatee for decryption. If correct, output 1, otherwise, output 0.
3) *A* checks whether
$$h'_A{}^{(id_A - id_B)} \cdot A_1{}^{a_i} \cdot \left(h_1{}^{r_A} g^{-r'_A}\right) = \left(h_1{}^{r_A} g^{-r'_A}\right)^{rk_{A \to B}}$$
to ensure that $rk_{A \to B}$ is a re-encryption key generated properly for delegation from her to *B*.
4) *A* sends the re-encryption key $rk_{A \to B}$ to Proxy via an authenticated channel.
5) *A* computes $h'_B{}^{1/r_A}$ and $B_1{}^{1/r_A}$, and sends them to *B* as partial decryption key.

Table I
SECURITY ANALYSIS

Property	BBS [2]	ID [6]	Ateniese [1]	Wang [11]	Ours
Uni-directional	No	Yes	Yes	Yes	Yes
Non-interactive	No	Yes	Yes	No	No
Proxy invisibility	Yes	No	Yes$^{\#}$	Yes	Yes$^{\#}$
Original-access	Yes	Yes	Yes	No	Yes
Key optimal	Yes	No	Yes	Yes	Yes
Collusion-safe	No	No	Yes	Yes	Yes
Temporary	Yes!	Yes!	Yes!	No	Yes!
Non-transitive	No	Yes	Yes	Yes	Yes
Non-transferable	No	No	No	No*	Yes
Non-Key-escrow	$--$	No	$--$	No	Yes
Non-PKG-despotism	$--$	No	$--$	No	Yes

($*$) PKG alone can transfer
($^{\#}$) Ateniese [1] can only achieve proxy invisible to delegatee, our scheme can only achieve proxy invisible to delegator.
(!) possible to achieve with additional assumption and overhead.

Re-Encryption:

Proxy computes $\beta = H(C_1, C_2, C_3, C_4)$ and tests whether $e(C_5, g) = C_2^{\beta} C_4$. If it is not equal, output $\perp$. Else computes $C_1{}' = C_1{}^{rk_{A \to B}} = g^{r_A s(\alpha - id_A)(\frac{\alpha - id_B}{\alpha - id_A} + a_i y)}$, and sends $(C_1{}', C_1, C_2, C_3, C_4, C_5)$ to ***B***.

Decryption (delegatee):

B computes $\beta = H(C_1, C_2, C_3, C_4)$ and tests whether $e(C_5, g) = C_2^{\beta} C_4$. If it is not equal, output $\perp$. Else *B* computes

$$C_3 \frac{e(C_1{}', h'_B{}^{(1/r_A)(1/r_B)}) C_2{}^{r_{B,1}}}{e(C_1, B_1{}^{(1/r_A)(1/r_B)})}$$
$$= C_3 \frac{e(g^{r_A s(\alpha - id_A)(\frac{\alpha - id_B}{\alpha - id_A} + a_i y)}, (h_1 g^{-r_{B,1}})^{\frac{1}{(\alpha - id_B) r_A}}) (e(g,g)^s)^{r_{B,1}}}{e(g^{r_A s(\alpha - id_A)}, (h_1 g^{-r_{B,1}})^{\frac{a_i y}{(\alpha - id_B) r_A}})}$$
$$= m$$

If $e(g,g)^{H'(m)} = C_4$ holds, return *m*; otherwise return $\perp$.

III. COMPARISON WITH EXISTING PROXY RE-ENCRYPTION SCHEMES

The main advantage of our scheme is: It achieved Non-transferable property, Non-Key-escrow property and Non-PKG-despotism property, in which Non-Key-escrow property and Non-PKG-despotism property are defined by us especially for estimating security of a PKG involved PRE schemes. To compare some existing proxy re-encryption schemes with our proposed scheme as fully as possible, we also analyze below some important properties defined in [1]. The comparison results are presented in Table 1. For more analysis details, please refer to the full version of our paper [5].

Our scheme adds more rounds of interaction for the following reasons:

- Private Key Correctness Check is added for checking the partial private key generated by PKG, since PKG is not fully trusted in our assumption.
- In Re-encryption Key Generation, step 1 and 2 are added for PKG to verify delegator and get delegatee's

identity, since PKG is responsible for generating re-encryption key. Without verification, attacker may impersonate delegator to trick PKG into generating re-encryption key.

- Partial-Decryption-Key Generation is added to prevent PKG, proxy and delegatee re-delegating decryption right from colluding. With this step, even if PKG, proxy and delegatee's collude, they are unable to generate re-encryption key for re-delegating decryption right without the original delegator's help.

Note that these extra interaction will not affect the efficiency of our scheme, since they are performed only once, at initialization time.

IV. Conclusions

In this paper, we attempt to solve the open problem pointed out in *NDSS* 2005, in proposing a non-transferable proxy re-encryption scheme in which the proxy and a delegatee cannot collude to transfer decryption rights. We also introduced two important properties, namely *Non-Key-escrow* and *Non-PKG-despotism*, into the proposed PRE scheme. The principle behind our solution is that instead of 'prohibiting' a party to propagate information, we punish the party who illegitimately propagates information by exposing the important secrets of the party. This method is feasible due to the fact that nobody would run the risk of exposing its own secrets to do illegal decryption right transfer. Thus, our 'punish' method is more practicable and effective than the 'tracing' method in [7] and the 'unforgeability' method in [4], because it can strongly prevent illegal decryption right transfer from happening, but not just tracing the malicious proxy after the illegal decryption right transfer.

To the best of our knowledge, our paper is the first paper which practically solves the transferable problem.

Acknowledgment

We would like to show our deepest gratitude to Sherman S.M. Chow, for all his kindness and help. Without his valuable comment, we could not have solved the difficult part of this paper. The work described in this paper was partially supported by the General Research Fund from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. RGC GRF HKU 713009E), the NSFC/RGC Joint Research Scheme (Project No. N_HKU 722/09), HKU Seed Fundings for Applied Research 201102160014, and HKU Seed Fundings for Basic Research 201011159162 and 200911159149.

References

[1] G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. In *NDSS*, pages 29–43, February 2005.

[2] M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. In *EUROCRYPT*, pages 127–144, June 1998.

[3] V. Goyal. Reducing trust in the pkg in identity based cryptosystems. In *CRYPTO*, pages 430–447, August 2007.

[4] R. Hayashi, T. Matsushita, T. Yoshida, Y. Fujii, and K. Okada. Unforgeability of re-encryption keys against collusion attack in proxy re-encryption. In *Proceedings of the 6th International conference on Advances in information and computer security*, IWSEC'11, pages 210–229, Berlin, Heidelberg, 2011. Springer-Verlag.

[5] Y.-J. He, T. W. Chim, L. C. K. Hui, and S. M. Yiu. Non-transferable proxy re-encryption scheme for data dissemination control. *http://eprint.iacr.org/2010/192.pdf*.

[6] A. Ivan and Y. Dodis. Proxy cryptography revisited. In *NDSS*, February 2003.

[7] B. Libert and D. Vergnaud. Tracing malicious proxies in proxy re-encryption. In *Pairing*, pages 332–353, September 2008.

[8] T. Matsuo. Proxy re-encryption systems for identity-based encryption. In *Pairing*, pages 247–267, July 2007.

[9] K. Niu, X. A. Wang, and M. Q. Zhang. How to solve key escrow problem in proxy re-encryption from cbe to ibe. In *DBTA*, pages 95–98, April 2009.

[10] X. A. Wang and X. Y. Yang. Identity based broadcast encryption based on one to many identity based proxy re-encryption. In *IEEE International Conference on Computer Science and Information Technology*, pages 47–50, August 2009.

[11] X. A. Wang and X. Y. Yang. Proxy re-encryption scheme based on bb2 identity based encryption. In *IEEE International Conference on Computer Science and Information Technology*, pages 134–137, August 2009.

[12] X. A. Wang and X. Y. Yang. Proxy re-encryption scheme based on sk identity based encryption. In *IAS*, pages 657–660, August 2009.

[13] X. A. Wang, X. Y. Yang, and F. G. Li. On the role of pkg for proxy re-encryption in identity based setting. *Cryptology ePrint Archive, Report 2008/410*, 2008.

[14] X. A. Wang, X. Y. Yang, and M. Q. Zhang. Proxy re-encryption scheme from ibe to cbe. In *DBTA*, pages 99–102, April 2009.