Non-Transferable Proxy Re-Encryption

Yi-Jun He, L.C.K. Hui, and S.M. Yiu

Department of Computer Science, The University of Hong Kong
Chow Yei Ching Building, Pokfulam Road, Hong Kong
{yjhe,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. With the help of the proxy, Alice can delegate the decryption right to any delegatee. However, existing PRE schemes generally suffer from one of the followings. Some schemes fail to provide the *non-transferability property* in which the proxy and the delegatee can collude to further delegate the decryption right to anyone. 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 encrypting a given ciphertext for a target delegatee. But this poses two problems in PRE schemes: the PKG in their schemes may decrypt all ciphertexts (referred as the *key escrow* problem) and the PKG can generate re-encryption key for arbitrary delegatees (we refer it as the *PKG despotism* problem). In this paper, we provide a more satisfactory solution to the problems. We follow the idea of using PKG to generate a re-encryption key to achieve the non-transferability property. To tackle the PKG despotisum problem in our scheme, if the PKG generates a re-encryption key for an unauthorized party, the delegator is able to retrieve the master secret of the PKG. We also show that with a tamper-proof hardware device, we can guarantee that the PKG cannot transfer decryption right to unauthorized delegatee. In addition, we solve the key escrow problem as well.

Key words: proxy re-encryption, identity based encryption, non-transferability property

1 Introduction

1.1 Background

In 1998, Blaze, Bleumer and Strauss [1] proposed a novel cryptographic scheme known as proxy re-encryption scheme (PRE), which 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. This is illstrated in Figure 1 where the Sender encrypts a text for Alice; Alice sends a re-encryption key and the ciphertext to the proxy which performs the re-encryption and sends Bob the re-encrypted ciphertext which can be decrypted by Bob without knowing the secret key of Alice. The above scheme aroused much

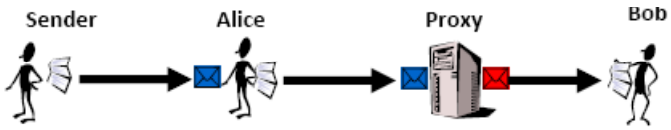

Fig. 1. Proxy Re-Encryption

interest in the encryption community [1,2,4-7,10-14] since it could be exploited in a number of applications for achieving better information security and privacy, such as:

- Email forwarding: The proxy can "forward" re-encrypted messages to a delegated recipient.
- Encrypted files distribution: The encrypted files are stored in a file server. Only the content owners can grant the access right of the files to the target users; even the file server operator has no right to access the files.
- Law-enforcement monitoring: The encrypted communication data are transferred via an Internet service provider (ISP). The ISP can require the content owners to provide the access right to the law enforcement officers to let them monitor the data being transferred to various users; however, the ISP operator cannot access the data.

1.2 Limitations of Existing Schemes and Our Contributions

However, to the best of our knowledge, existing PRE schemes have the following problems (details of existing schemes will be given in the next section). Some schemes suffer from the problem of failing to provide the *non-transferability* property which was first addressed in 2005 [2]. A proxy re-encryption scheme is said to be non-transferable if the proxy and a set of colluding delegatees cannot re-delegate decryption rights to other parties. One may argue that transferability may not be preventable (see also Ateniese *et al.* [2]) since the delegatee can always decrypt and forward the plaintext to another party, or the delegatee can always send its secret key to another party. However, in doing so, the delegatee has to explicitly send the plaintext over or disclose the secret key which increases the chance of being caught and put 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.

To resovle this problem, some other identity-based PRE schemes 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 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 for which the PKG in their schemes may be able to decrypt all re-encrypted cipertexts. And the PKG has the power of generating re-encryption key for arbitrary delegatees (we refer this as the PKG despotism problem). In this paper, we propose a non-transferable re-encryption scheme which also makes use of a PKG, however, the design of our scheme exploits the idea of master secret key retrieval mentioned in [3] so that if the PKG tries to generate a re-encryption key for an unauthorized delegatee, the original delegator is able to retrieve the master secret of the PKG. The characteristics of our proposed scheme are summarized as follow.

- The proposed scheme has the non-transferability property. The re-encryption key is generated by a private key generator (PKG); thus delegatee and proxy cannot collude to re-delegate decryption rights since they do not have knowledge of PKG's master secret.
- With the cooperation of proxy, the delegator can retrieve the master secret of PKG if it generates a re-encryption key to grant the access right of a ciphertext to an unauthorized party. Therefore, the proposed scheme could deter PKG from delegating access rights to other parties without approval from the legitimate delegator, thus tackling the PKG despotisum problem.
- With the introduction of a tamper-proof hardware device to the proposed scheme, PKG can no longer transfer decryption right to unauthorized parties and completely resolve the PKG despotism problem.
- In our scheme, the PKG cannot decrypt the re-encrypted ciphertexts, thus solving the key escrow problem.

2 Related Work

Blaze, Bleumer and Strauss [1] proposed the first proxy re-encryption scheme, which is based on ELGamal encryption. But this scheme is bi-directional, that is, when the proxy is allowed to re-encrypt Alice's messages under Bob's key, it can also re-encrypt Bob's messages under Alice's key. Bob may not like this. Another weakness is that if the proxy colludes with Alice, they can easily learn Bob's secret key SK_B. Likewise, the proxy and Bob may collude to learn Alice's secret key. Furthermore, in order to compute the re-encryption key from A to B, denoted as $rk_{A->B}$, one party must share his or her secret key with the other or they must rely on a trusted third party. The other drawback is that the scheme is transitive in the following sense. Suppose that the proxy is allowed to generate two re-encryption keys $rk_{A->B}$ and $rk_{B->C}$; then the proxy can derive an additional re-encryption key $rk_{A->C}$ for delegation from A to C.

Later, Ivan and Dodis [4] proposed three unidirectional proxy re-encryption schemes based on ElGamal, RSA, and IBE (ID-based encryption) respectively. Their main contribution is that they solved (i) the bi-directional problem and (ii) the transitive problem in [1]. But in their schemes, Alice's private key is split into two parts DK_1 and DK_2, with DK_1 distributed to proxy and DK_2 distributed to Bob. Thus when the proxy colludes with Bob, they can derive Alice's private key.

In 2005, Ateniese *et al.* [2] presented three proxy re-encryption schemes which are considered to be more secure than other approaches. Their major advantages are the following. The schemes are unidirectional and the delegator's private key is protected from being disclosed by the collusion of proxy and a delegatee. They implemented one of their proposed schemes in a secure distributed file system to show that the scheme can work efficiently in practice. They summarized nine important properties of proxy re-encryption schemes, which include the non-transferability property. Lacking the non-transferable property in all existing schemes was considered an open problem of the contemporary PRE schemes.

This open problem was first addressed in 2008 by Libert and Vergnaud [5]. They 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 proposed, instead of preventing the collusion of proxy and delegatee, tracing the malicious proxy after its collusion with one or more delegatees. It is the first attempt to address the open problem. However, it still cannot prevent re-delegation from happening.

Matsuo's PRE schemes [6] use the PKG to help generating re-encryption key for the delegator and the delegatee. Based on this approach, they proposed two PRE schemes: one for the decryption right delegation from a user of PKI-based public key encryption system to IBE system users, and the other for the delegation among IBE system users. This is the first set of schemes that use PKG to generate re-encryption key. However, the PKG in the schemes can decrypt all re-encrypted ciphertexts; so, there is a potential security problem as long as PKG is untrusted or malicious.

In 2008, Wang *et al.*[7] extended the idea of Matsuo's scheme by allowing PKG to generate re-encryption keys based on its master secret key. They proposed several proxy re-encryption schemes:(i) PRE from IBE to Certificate Based Public Key Encryption; (ii) PRE based on a variant of the first system of Selective identity secure IBE [8]; (iii) PRE based on the second system of Selective identity secure IBE [8];and (iv) PRE based on Sakai-Kasahara IBE scheme [9]. Based on this work, Wang *et al.* proposed five other schemes [10-14] to address different problems of proxy re-encryption schemes. However, there are still some issues not yet addressed in each one of them. In [10], the proxy can re-encrypt on its own the ciphertext for the delegator into ciphertext for any delegatee; this is not a desired property of PRE. In [11], it seems that they solved the open problem related to the non-transferability issue, since proxy and delegate cannot collude to re-delegate decryption right; however, in the scheme, the PKG can delegate arbitrarily to anyone as it can generate a re-encryption key for any delegatee. In [12,14], the PKG can also delegate arbitrarily as what it could do in [11]. Among the five schemes, [11] seems to be the best in solving the non-transferability issue, we will compare our scheme with [11] in Section 5.2.

3 Preliminaries

3.1 Bilinear Map

Let G and G_T be multiplicative cyclic groups of prime order p, and g be generator of G. We say that G_T has an admissible bilinear map e: $G \times G \rightarrow G_T$, if the following conditions hold.

- $e(g^a, g^b) = e(g, g)^{ab}$ for all a, b.
- $e(g, g) \neq 1$.
- There is an efficient algorithm to compute $e(g^a, g^b)$ for all a, b and g.

3.2 Assumption

The security of our concrete construction is based on a complexity assumption, called "Truncated Decision Augmented Bilinear Diffie-Hellman Exponent Assumption (Truncated *q-ABDHE*)"proposed in [15].

Let $e : G \times G \rightarrow G_T$ be a bilinear map, where G and G_T are cyclic groups of large prime order p. Given a vector of q+3 elements:

$$(g', g'^{(\alpha^{q+2})}, g, g^{\alpha}, \ldots, g^{(\alpha^q)}) \in G^{q+3}$$

and an element $Z \in G_T$ as input, output 0 if $Z = e(g^{(\alpha^{q+1})}, g')$ and output 1 otherwise.

An algorithm $\mathcal{B}$ has advantage ε in solving the truncated *q-ABDHE* if:

$$|\Pr[\mathcal{B}(g', g'^{(\alpha^{q+2})}, g, g^{\alpha}, \ldots, g^{(\alpha^q)}, e(g^{(\alpha^{q+1})}, g')) = 0]$$

$$-\Pr[\mathcal{B}(g', g'^{(\alpha^{q+2})}, g, g^{\alpha}, \ldots, g^{(\alpha^q)}, Z) = 0]| \geq \varepsilon$$

where the probability is over the random choice of generators g, g' in G, the random choice of α in Z_p, the random choice of $Z \in G_T$, and the random bits consumed by $\mathcal{B}$.

Definition 1 [15].We say that the truncated (decision) (t, ε, q)-ABDHE assumption holds in G if no t-time algorithm has advantage at least ε in solving the truncated (decision) q-ABDHE problem in G.

4 Framework of Our Non-Transferable PRE Scheme and Security Model

4.1 Non-Transferable PRE Model

Our Non-Transferable PRE scheme is composed of nine algorithms:

- Setup. On input a security parameter 1^k, the public parameters *mpk* and master secret key *msk* are generated. A value $\mathcal{T}$ is generated and can be verified by anyone.

- Key Generation. On input a user's identity *ID*, *msk*, algorithm generates public key and one part of private key for user. Combining this part of the private key, user generates the whole private key for himself.
- Correctness-Check. Algorithm checks the correctness of the private key.
- Encryption. The encryption algorithm takes public key pk_i of delegator i and message m as input, outputs a ciphertext C_i encrypted under pk_i.
- Decryption(delegator). The decryption algorithm takes secret key sk_i of delegator i and ciphertext C_i as input, outputs message m. This algorithm actually is not necessary for PRE scheme. We put it here just for indicating that delegator has the ability to decrypt ciphertext C_i.
- Re-Encryption Key Generation. Algorithm verifies the delegator i's signature, and extracts delegatee j's *ID* from signature. The re-encryption key generation algorithm outputs a re-encryption key $rk_{i\to j}$ and other relational values.
- Re-Encryption. The re-encryption algorithm takes re-encryption key $rk_{i\to j}$ and ciphertext C_i as input, outputs a re-encrypted ciphertext C_j under pk_j.
- Decryption(delegatee). The decryption algorithm takes secret key sk_j of delegatee j and ciphertext C_j as input, outputs message m.
- Retrieve master secret key. If PKG cheating, delegator i can retrieve PKG's master secret key.

4.2 Security Model for Non-Transferable PRE Scheme

We define a new property called retrievability of Non-Transferable PRE Scheme in the paper. This property is for PRE, thus it is different from the one mentioned in [3].

Retrievability of Non-Transferable PRE Scheme: If the adversary is able to come up with two re-encryption keys for delegation from one party to two different parties without delegator's approval, the delegator can retrieve the master secret key of adversary.

The following retrievability game considers a malicious PKG which tries to generate two re-encryption keys for one party, and its master secret key cannot be retrieved.

Retrievability Game: On input a security parameter 1^k, $k \in N$, a simulator $\mathcal{S}$ invokes an adversary $\mathcal{A}$ on 1^k. $\mathcal{A}$ returns a master public key *mpk*, two re-encryption keys, one is *rk* for delegation from A to B, the other one is *rk'* for delegation from A to C, and six values $\left(h_1{}^{r_A} g^{-r'_A}\right)^y$, $\left(h_1{}^{r_A} g^{-r'_A}\right)^k$, $h_i{}^{a_i y}$, $h_i{}^{a_i k}$, $(h_1{}^{r_B} g^{-r'_B})^{a_i y/(\alpha - id_B)}$, $(h_1{}^{r_C} g^{-r'_C})^{a_i k/(\alpha - id_C)}$, A's partial private key $h'_A = (Rg^{-r'_A})^{1/(\alpha - id_A)}$, B's partial private key $h'_B = (Rg^{-r'_B})^{1/(\alpha - id_B)}$, C's partial private key $h'_C = (Rg^{-r'_C})^{1/(\alpha - id_C)}$. $\mathcal{A}$ wins if

1. $1 \leftarrow$ verification$(rk, \left(h_1{}^{r_A} g^{-r'_A}\right)^y, h'_A, a_i)$;

2. $1 \leftarrow$ verification(rk', $\left({h_1}^{r_A} g^{-r'_A}\right)^k$, h'_A, a_i);
3. $1 \leftarrow$ Verification($({h_1}^{r_B} g^{-r'_B})^{a_i y/(\alpha - id_B)}$,${h_i}^{a_i y}$,$h'_B$);
4. $1 \leftarrow$ Verification($({h_1}^{r_C} g^{-r'_C})^{a_i k/(\alpha - id_C)}$,${h_i}^{a_i k}$,$h'_C$);
5. $\perp \leftarrow$ Retrieve(mpk, id_B, id_C, rk, rk',${h_i}^{a_i y}$, ${h_i}^{a_i k}$);

The advantage of $\mathcal{A}$ in this game is defined as Pr[$\mathcal{A}$ wins].

5 Our Non-Transferable PRE Scheme

We construct the Non-Transferable PRE scheme based on the basic crypto system proposed in [16]. Our PRE scheme can successfully solve the transferable problem in existing PRE schemes. 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 to delegator. Delegator checks the correctness of the key, and then sends it to the proxy. 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. If PKG generates a re-encryption key to transfer decryption rights without delegator's approval, PKG's master secret would be retrieved by delegator.

5.1 The Construction

In the following sections, we let Alice be the delegator, and Bob 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:$\{0,1\}^* \rightarrow \mathrm{Z}_p$ 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 $\mathcal{M} = G_T$. The public parameters mpk and master secret key msk are given by

$$mpk = (g, g_1, h_1, h_2, h_3, H_I, H, \mathcal{M}), msk = (\alpha)$$

PKG randomly chooses $x \in Z_p$. It sets $X = e(g,g)^x$ which is published, but x is kept secure. Let N:=poly(k) for some polynomial poly($\cdot$), be a security parameter. Let H_E:$\{0,1\}^* \rightarrow \{0,1\}^N$ be a secure hash function. Then PKG runs the non-interactive verifiable encryption algorithm from [17] to verifiably encrypt the secret value α under X. This can be done as follows:

1. PKG randomly selects $u_j \in Z_p$ and computes ($T_j = g^{u_j}$), for $j = 1$ to N.
2. For $j = 1$ to N, PKG computes the following.
   - Compute $Z_{j,0} = u_j$, $Z_{j,1} = u_j - \alpha$.
   - Randomly select $v_{j,0}$, $v_{j,1} \in Z_p$, compute $E_{0,j,i} = X^{v_{j,i}} \oplus Z_{j,i}$, $E_{1,j,i} = g^{v_{j,i}}$ for $i \in \{0,1\}^5$.

3. PKG computes

$$L = H_E\left(T_1||E_{0,1,0}||E_{1,1,0}||E_{0,1,1}||E_{1,1,1}||...||T_N||E_{0,N,0}||E_{1,N,0}||E_{0,N,1}||E_{1,N,1}\right)$$

Let b_j be the j^{th} bit of L.
4. Output $\mathcal{T} = \{(T_j, E_{0,j,0}, E_{1,j,0}, E_{0,j,1}, E_{1,j,1}, Z_{j,b_j}, v_{j,b_j})\}^{N}_{j=1}$. N controls the cheating probability of the verifiable encryption.
5. Verification. Anyone can check if $\mathcal{T}$ is a valid encryption of α under X by computing L from $\mathcal{T}$ and checking if the following equations hold:
$E_{0,j,b_j} \stackrel{?}{=} X^{v_{j,b_j}} \oplus Z_{j,b_j}$, $E_{1,j,b_j} \stackrel{?}{=} g^{v_{j,b_j}}$, $T_j \stackrel{?}{=} {g_1}^{b_j} g^{Z_{j,b_j}}$
where $j = 1$ to N and b_j is the j^{th} bit of L.

Key Generation:

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

On input the public key/master secret key pair (mpk, msk) and an identity $ID \in \{0,1\}^k$ of a user U, the PKG computes id=$H_I(ID)$ as U's public key. If $id = \alpha$, it aborts. Otherwise, the protocol proceeds as follow:

1. $r_{ID} \in Z_p$ is a secret value of user U. U sends $R = {h_1}^{r_{ID}}$ to the PKG.
2. U gives PKG the following zero-knowledge proof of knowledge:

$$PK\{r_{ID} : R = {h_1}^{r_{ID}}\}$$

3. The PKG randomly selects r'_{ID}, $r_{\text{ID},2}$, $r_{\text{ID},3} \in Z_p$ and computes
$h'_{ID} = (Rg^{-r'_{ID}})^{1/(\alpha - id)}$, $h_{ID,2} = (h_2 g^{-r_{ID,2}})^{1/(\alpha - id)}$, $h_{ID,3} = (h_3 g^{-r_{ID,3}})^{1/(\alpha - id)}$
and sends (r'_{ID}, h'_{ID}, $r_{ID,2}$, $h_{ID,2}$, $r_{ID,3}$, $h_{ID,3}$) to U.
4. U computes

$$r_{ID,1} = r'_{ID}/r_{ID}, h_{ID,1} = (h'_{ID})^{1/r_{ID}} = (h_1 g^{-r_{ID,1}})^{1/(\alpha - id)}$$

The secret key usk_{ID} is ($r_{ID}, r_{ID,1}$, $h_{ID,1}$, $r_{ID,2}$, $h_{ID,2}$, $r_{ID,3}$, $h_{ID,3}$).
Therefore, the delegator Alice's private key can be denoted as

$$usk_A = (r_A, r_{A,1} = r'_A/r_A, h_{A,1} = (h_1 g^{-r_{A,1}})^{1/(\alpha - id_A)}, r_{A,2}, h_{A,2}, r_{A,3}, h_{A,3})$$

and the delegatee Bob's private key is denoted as

$$usk_B = (r_B, r_{B,1} = r'_B/r_B, h_{B,1} = (h_1 g^{-r_{B,1}})^{1/(\alpha - id_B)}, r_{B,2}, h_{B,2}, r_{B,3}, h_{B,3})$$

Correctness-Check:

On input (mpk, usk_{ID}) and an identity $ID \in \{0,1\}^k$, user computes $id = H_I(ID)$ and check whether $e(h_{ID,i}, g_1/g^{id}) = e(h_1 g^{-r_{\text{ID},i}}, g)$ for i=1,2,3. If correct, output 1. Otherwise, output 0.

Encryption:

To encrypt a message $m \in G_T$ using public key, delegator Alice generates a unique randomly-selected secret parameter $s \in Z_p$, and computes $id_A = H_I(A)$. Finally, delegator outputs the ciphertext C where:

$$C = (C_1, C_2, C_3, C_4) = (g_1{}^s g^{-sid_A}, e(g,g)^s, m \cdot e(g,h_1)^{-s}, e(g,h_2)^s e(g,h_3)^{s\beta})$$

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

Decryption(delegator):

To decrypt a ciphertext $C = (C_1, C_2, C_3, C_4)$ using secret key usk_A, Alice computes $\beta = H(C_1, C_2, C_3)$ and test whether

$$C_4 = e(C_1, h_{ID,2} h_{ID,3}{}^{\beta}) \cdot C_2{}^{r_{ID,2} + r_{ID,3}\beta}$$

If it is not equal, output $\perp$. Else output

$$m = C_3 \cdot e\left(C_1, h_{A,1}\right) \cdot C_2{}^{r_{A,1}}$$

The following Re-Encryption is done through an interactive protocol among Alice, Bob, PKG and Proxy, which is shown in Figure 2.

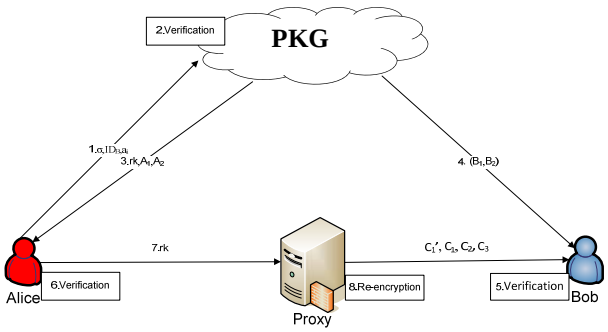

Fig. 2. Proposed Non-Transferable Proxy Re-encryption framework

Re-Encryption Key Generation:

1. The delegator Alice 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. Alice signs Bob'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 Alice to identify the identity of the delegator.
   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 $rk = (\frac{\alpha - id_B}{\alpha - id_A} + a_i y) \bmod p$ for Alice. PKG broadcasts a random value $h_i \in G$ for each time period $i \geq 1$, and sends rk, $A_1 = \left({h_1}^{r_A} g^{-r'_A}\right)^y$, $A_2 = g^x {h_i}^{(\frac{\alpha}{\alpha - id_A})}$ to Alice.
4. PKG generates $B_1 = \left({h_1}^{r_B} g^{-r'_B}\right)^{a_i y/(\alpha - id_B)}$, $B_2 = {h_i}^{a_i y}$ and sends them to the delegatee Bob.
5. Bob checks whether
$$e(h_i, B_1) = e(B_2, h'_B)$$
to ensure B_1 is a valid value which will help him for decryption later. If correct, output 1, otherwise, output 0.
6. Alice checks whether
$${h'_A}^{(id_A - id_B)} \cdot {A_1}^{a_i} \cdot \left({h_1}^{r_A} g^{-r'_A}\right) = \left({h_1}^{r_A} g^{-r'_A}\right)^{rk}$$
to ensure that rk is a re-encryption key generated properly for delegation from her to Bob.
7. Alice sends the re-encryption key rk to Proxy via an authenticated channel.

***Re-Encryption**:*

Proxy computes ${C_1}' = {C_1}^{rk} = g^{s(\alpha - id_A)(\frac{\alpha - id_B}{\alpha - id_A} + a_i y)}$, and sends $({C_1}', C_1, C_2, C_3)$ to Bob.

Decryption (delegatee):

Bob computes

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

Retrieve master secret key:

In our scheme, PKG is allowed to generate only one re-encryption key for the same delegator during a time period i. If during the same time period i, PKG generates another re-encryption key $rk' = (\frac{\alpha - id_C}{\alpha - id_A} + a_i k) \bmod p$ to proxy for delegation from Alice to another party (for example, Carole), Alice can retrieve PKG's master secret. This strategy deters PKG from generating re-encryption keys arbitrarily.

The retrieving process is like this: Alice gets ${h_i}^{a_i k}$ from Carole, B_2 from Bob, and *rk'* from proxy. Then, Alice computes

$$P_1 = \frac{A_2}{{h_i}^{rk'}} = g^x {h_i}^{(\frac{id_C}{\alpha - id_A} - a_i k)}$$

$$P_2 = \frac{A_2}{{h_i}^{rk}} = g^x {h_i}^{(\frac{id_B}{\alpha - id_A} - a_i y)}$$

From P_1, Alice derives $P_3 = P_1 {h_i}^{a_i k} = g^x {h_i}^{(\frac{id_C}{\alpha - id_A})}$
From P_2, Alice derives $P_4 = P_2 B_2 = g^x {h_i}^{(\frac{id_B}{\alpha - id_A})}$
From P_3 and P_4, Alice derives

$$g^x = \left(\frac{{P_3}^{id_B}}{{P_4}^{id_C}} \right)^{\frac{1}{id_C - id_B}}$$

and check whether $X \stackrel{?}{=} e(g, g^x)$.

Once g^x is obtained, for any $j \in \{1, ..., N\}$, one can get (T_j,Z_{j,b_j}, $E_{0,j,1-b_j}$, $E_{1,j,1-b_j}$) in $\mathcal{T}$ (the verifiable encryption in *Setup*). For simplicity, we omit the subscript j. That is, one can get ($T := g^u, Z_b, E_{0,1-b}, E_{1,1-b} := g^{v_{1-b}}$), such that $b \in \{0, 1\}$ where

$$Z_b = u - b\alpha$$

Compute $e(E_{1,1-b}, g^x) \oplus E_{0,1-b}$ to get

$$Z_{1-b} = u - (1 - b)\alpha$$

From Z_b and Z_{1-b}, Alice computes α. Check whether $g_1 \stackrel{?}{=} g^\alpha$. If not, use another $j \in \{1, ..., N\}$ to compute α.

5.2 Security Analysis

To compare some existing proxy re-encryption schemes with our proposed scheme, we outline below some important properties defined in [2]. The comparison results are presented in Table 1.

- *Unidirectional*: Delegation from $A \rightarrow B$ does not allow re-encryption from $B \rightarrow A$.
- *Non-interactive*: Re-encryption keys can be generated by Alice using Bob's public key; no trusted third party or interaction is required. In our proposed scheme, PKG is employed to generate re-encryption keys, so delegator needs to interact with PKG to generate the keys.
- *Proxy transparent*: This is an important feature possessed by the original BBS scheme [1]. The proxy in the BBS scheme is *transparent* in the sense that neither the sender of an encrypted message nor any of the delegatees has to be aware of the existence of the proxy. In BBS scheme, this property

is achieved at the price of allowing transitive delegation and recovery of the master secrets of the delegator. In Ateniese's scheme, only a weaker form of proxy transparency, called *proxy invisibility* can be achieved, because the sender needs to know the existence of proxy, in order to decide whether to generate first-level encryption or second-level encryption. In our proposed scheme, proxy is transparent. Both sender and delegatees do not have to know the proxy, since there is only one form of encryption.

- *Original-access*: Alice can decrypt re-encrypted ciphertexts that were originally sent to her.
- *Key optimal*: The size of Bob's secret storage remains constant, regardless of how many delegations he accepts. Like Ateniese's scheme, delegatee is allowed to decrypt re-encrypted ciphertext during some specific time period i. Thus information received from PKG for decryption only need to exist temporarily in delegatee's side. After a time period i, the information would be invalid. Delegatee can delete the information immediately. Thus in the long run, our scheme is still key optimal.
- *Collusion-"safe"*: Bob and the proxy's collusion cannot recover Alice's secret key. In our proposed scheme, secrecy of Alice's secret key depends on a random value r_A. It is chosen by Alice, and is not used in re-encryption key. Although Bob and proxy collude, they cannot recover it.
- *Temporary*: Bob is only able to decrypt messages intended for Alice that were authored during some specific time period i. In our scheme, to achieve temporary proxy re-encryption, for each time period $i \geq 1$, Alice generates a random value $a_i \in Z_p$, and PKG broadcasts a random value $h_i \in G$. Because a_i and h_i will be invalid after time period i, the re-encryption key's life cycle is also period i. We remark that in all existing schemes including our scheme, the temporary property is achieved based on the assumption that the proxy will update the re-encryption key after each period expires.
- *Non-transitive*: Based on the re-encryption keys, $rk_{A\to B}$ and $rk_{B\to C}$, the proxy cannot produce $rk_{A\to C}$. In our proposed scheme, the re-encryption key is generated using the master secret key of the PKG, proxy cannot generate $rk_{A\to C}$ without knowing the master secret key. And the delegatee's identity is included in the re-encryption key, the proxy is unable to replace the delegatee with another party. So even with the keys $rk_{A\to B}$ and $rk_{B\to C}$, the proxy cannot produce $rk_{A\to C}$.
- Non-transferable: The proxy and a set of colluding delegatees cannot re-delegate decryption rights. For example, from $rk_{A\to B}$, sk_B and pk_C, they cannot produce $rk_{A\to C}$. The re-encryption key is generated by PKG, thus even if delegatee colludes with proxy, they are unable to re-delegate their decryption rights since they do not have knowledge of the master secret.

As to IBE-based PRE, we add two more properties:

- *Non-Key-escrow*: In IBE system, PKG is responsible for generating private keys for users. But in IBE-based PRE, PKG should not be allowed to decrypt all ciphertext for anyone. In our proposed scheme, users' private keys are computed from the values provided by PKG, but they do not use directly

the values from PKG as the private keys. Only users themselves can get the knowledge of their own private keys.
- *Non-PKG-despotism*: PKG cannot generate re-encryption key arbitrarily without permission from delegator, otherwise there is some way to punish PKG by retrieving its master secret key. Moreover, we discussed one approach related to using a hardware device to prevent completely PKG despotism (see Section 6).

Table 1. Comparison of existing PRE schemes and our proposed scheme

Property	BBS [1]	ID [4]	Ateniese[2]	Wang[11]	Our Scheme
Unidirectional	No	Yes	Yes	Yes	Yes
Non-interactive	No	Yes	Yes	No	No
Proxy transparent	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
($^{\#}$) can only achieve proxy invisibility which is a weaker form of proxy transparent

Theorem 1. The advantage of an adversary in the *IND-ID-CCA* game is negligible for the proposed scheme under the truncated (decision) *q-ABDHE* assumption in the random oracle model.
Proof: Our proof closely follows the line of the proof of Gentry's scheme [15]. Assume there is an adversary $\mathcal{A}$ that breaks the *IND-ID-CCA* security of our scheme. We construct an algorithm $\mathcal{B}$ that solves the truncated (decision) *q-ABDHE* problem. $\mathcal{B}$ takes as input a random truncated decision *q-ABDHE* challenge $(g', g'_{q+2},\ g, g_1, ..., g_q, Z)$, where Z is either $e(g_{q+1}, g')$ or a random element of G_T. Algorithm $\mathcal{B}$ proceeds as follows.
Setup:

$\mathcal{B}$ generates random polynomials $f_i(x) \in Z_p[x]$ of degree q for $i \in \{1, 2, 3\}$. It sets $h_i = g^{f_i(\alpha)}$. It sends the public key (g, g_1, h_1, h_2, h_3) to $\mathcal{A}$. Since g, α, and $f_i(x)$ for $i \in \{1, 2, 3\}$ are chosen uniformly at random, h_1, h_2 and h_3 are uniformly random and the public key has a distribution identical to that in the actual construction. $\mathcal{B}$ randomly generates $x \in Z_p$ and sets $X = e(g, g)^x$. $\mathcal{B}$ also moderates the hash function $H_I()$, $H()$ as random oracle. By controlling the random oracle, $\mathcal{B}$ can simulate the transcript $\mathcal{T}$ easily.
Phase 1:

- Secret Key Queries.
  $\mathcal{A}$ makes key generation queries. $\mathcal{B}$ responds to a query on $ID \in Z_p$ as follows. If $ID = \alpha$, $\mathcal{B}$ uses α to solve truncated decision *q-ABDHE* immediately. Else, to generate a pair $(r_{ID,1}, h_{ID,1})$ such that $h_{ID,1} = (h_1 g^{-r_{ID,1}})^{1/(\alpha - id)}$, $\mathcal{B}$ sets $r_{ID,1} = f_1(ID)$ and computes $h_{ID,1} = g^{F_{ID}(\alpha)} = g^{(f(\alpha)-f(ID))/(\alpha-id)} = (hg^{-f(ID)})^{1/(\alpha-id)}$. Then we use the same technique in [16] to extract r_{ID} from the user and set r'_{ID}=$r_{ID} r_{ID,1}$. $\mathcal{B}$ sends $(r'_{ID}, h'_{ID,1} = {h_{ID,1}}^{r_{ID}})$, ($r_{ID,2}$,$h_{ID,2}$), $(r_{ID,3}, h_{ID,3})$ to U. The user U will compute $r_{ID,1} = r'_{ID}/r_{ID}$, $h_{ID,1} = (h'_{ID,1})^{1/r_{ID}}$. $(r_{ID}, r_{ID,1}$,$h_{ID,1}$, $r_{ID,2}$, $h_{ID,2}$, $r_{ID,3}$, $h_{ID,3})$ are private keys of user.
- Re-encryption Key Queries.
  When $\mathcal{A}$ queries $\mathcal{B}$ for a re-encryption key about $ID_j \rightarrow ID_k$, $\mathcal{B}$ returns $rk = (\frac{\alpha - id_k}{\alpha - id_j} + a_i y) \bmod p$ to $\mathcal{A}$.
- Re-encryption Queries.
  Suppose that $\mathcal{A}$ queries $\mathcal{B}$ the re-encryption ciphertext C_k about $(C_j = (C_1, C_2, C_3, C_4), ID_j \rightarrow ID_k)$. $\mathcal{B}$ runs Re-encryption Key Queries to get the re-encryption key $rk = (\frac{\alpha - id_k}{\alpha - id_j} + a_i y) \bmod p$. Finally, $\mathcal{B}$ sets $C_k = ({C_1}', C_1, C_2, C_3) = ({C_1}^{rk}, C_1, C_2, C_3)$ and returns C_k to $\mathcal{A}$.
- Decryption Queries.
  To respond to a decryption query on (ID_k, C_k), $\mathcal{B}$ generates a private key for ID_k as above. It then decrypts C_k by performing the usual Decrypt algorithm with this private key.

Challenge:

In this phase, the procedure is as the same as in [15]. $\mathcal{A}$ outputs identities ID_0, ID_1 and messages M_0, M_1. If $\alpha \in \{ID_0, ID_1\}$, $\mathcal{B}$ uses α to solve truncated decision *q-ABDHE* immediately. Else, as before, $\mathcal{B}$ generates bits *b, c*$\in \{0, 1\}$. After computing a private key $\{(r_{ID}, r_{ID,i}, h_{ID,i}) : i \in \{1, 2, 3\}\}$ for ID_b, it also computes (u, v, w) as $u = g'^{f_2(\alpha) - f_2(\mathrm{ID}_b)}$, $v = Z \cdot e(g', \prod_{i=0}^{q} g^{F_{2,ID_b,i} \alpha^i})$, $w = M_c / e(u, h_{\mathrm{ID}_b}) v^{r_{\mathrm{ID}_b}}$, using the $(r_{ID_{b,1}}, h_{ID_{b,1}})$ portion of the key to compute w. After setting $\beta = H(u, v, w)$, $\mathcal{B}$ sets $y = e(u, h_{ID,2} {h_{ID,3}}^{\beta}) v^{r_{ID,2} + r_{ID,3}\beta}$. If $Z = e(g_{q+1}, g')$, then (u, v, w, y) is a valid, appropriately-distributed challenge to $\mathcal{A}$ for essentially the same reason as before.

Phase 2:

$\mathcal{A}$ makes key generation, re-encryption and decryption queries, and $\mathcal{B}$ responds as in Phase 1.

Guess:

Finally, adversary $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$. If $b = b'$, $\mathcal{B}$ outputs 1 as the solution to the truncated decision *q-ABDHE* problem; otherwise, it outputs 0.

If $Z = e(g_{q+1}, g')$, then $\mathcal{B}$'s simulation is perfect. Otherwise, if Z is a random element of G_T, then $\mathcal{A}$ can't get any information about M. So, $\mathcal{B}$'s success probability to solve the given truncated decision *q-ABDHE* problem is as the same as $\mathcal{A}$'s, namely $\varepsilon' = \varepsilon$. Thus Theorem 1 is proved.

Theorem 2. The advantage of an adversary in the *retrievability game* is negligible for the proposed scheme.

Proof: Let the output of $\mathcal{A}$ be key *mpk*, $g^x {h_i}^{(\frac{id_C}{\alpha - id_A})}$, $g^x {h_i}^{(\frac{id_B}{\alpha - id_A})}$, $rk = (\frac{\alpha - id_B}{\alpha - id_A} + a_i y) \bmod p$, $rk' = (\frac{\alpha - id_C}{\alpha - id_A} + a_i k) \bmod p$, ${h_i}^{a_i y}$, ${h_i}^{a_i k}$. $\mathcal{A}$ wins implies that conditions 1-5 defined in section 4.2 are all fulfilled.

Condition 1 and 2 imply that *rk* and *rk'* are sound, and $rk \neq rk'$. Condition 3 and 4 imply that ${h_i}^{a_i y}$ and ${h_i}^{a_i k}$ are sound. That means computation of P_3, P_4, g^x are correct, which have been proved in section 5.1. Condition 5 implies that either

1.) $g^x {h_i}^{(\frac{id_C}{\alpha - id_A})} = g^x {h_i}^{(\frac{id_B}{\alpha - id_A})}$; or
2.) $X \neq e(g,g)^x$ where g^x is computed from P_3 and P_4; or
3.) $g_1 \neq g^\alpha$ where α is computed from Z_b and Z_{1-b}, which have been proved in section 5.1.

Case 1.) happens with negligible probability, unless $id_B = id_C$, which requires $H_I(ID_B) = H_I(ID_C)$. But due to the collision resistance property of the hash function $H_I, H_I(ID_B) \neq H_I(ID_C)$.

Case 2.) happens with negligible probability, due to the soundness of verifications which has been proven in section 5.1.

Case 3.) also happens with negligible probability, due to the security of the verifiable encryption scheme [17]. Combining all cases, the adversary only has negligible advantage to win the game. Thus Theorem 2 is proved.

6 Discussions

In our proposed scheme, we use PKG to help in generating the re-encryption key; proxy and delegatee (Bob) cannot collude to re-delegate decryption rights; therefore the proposed scheme has solved the open problem concerned with non-transferability.

However, with the introduction of PKG, we need to consider whether the PKG can carry out any significant attack by illegally generating re-encryption keys. But now we are going to state that this is impossible. Firstly, even if PKG had illegally generated a re-encryption key, this key is useless if the proxy does not carry out re-encryption. Secondly, the PKG will expose his master secret key as the result of of this illegal behavior (proved in section 5).

As an extension, if we make use of a tamper-resistant device, then we can modify the scheme so that illegal re-encryption cannot occur. This is to solve the very unlikely situation that if the PKG collude with the proxy, and the PKG is willing to risk exposing his master secret.

We suggest the use of a secure tamper-resistant hardware on the PKG side to be responsible for re-encryption key generation. The system framework is shown in Figure 3.

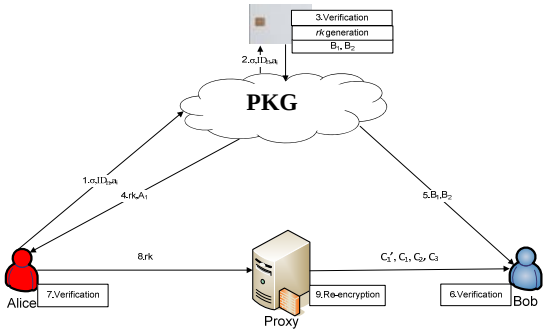

Fig. 3. Proxy Re-encryption framework with hardware

In general, the hardware is a token that contains a security system with tamper-resistant properties (*e.g.* a secure crypto processor) and is capable of providing security services (e.g. confidentiality of information in the memory). Smart cards [18] such as Java Card satisfy these requirements. We model secure hardware devices as black-boxes with internal memory which can be accessed only via its I/O interface. Master secret of PKG is generated by this hardware device, and saved in its memory. As the hardware device is tamper-proof, no one can break it to get the knowledge of the master secret. We view re-encryption key generation program as a predefined function saved in hardware device. Once input Alice's signature to hardware device, the hardware device will not execute re-encryption key generation program until it verifies the signature. Finally, the re-encryption key is the output, and will be sent to Alice. So even PKG cannot generate the re-encryption key arbitrarily, since it does not have the knowledge of master secret.

7 Conclusions

In this paper, we attempt to solve the open problem pointed out in 2005, in proposing a non-transferable proxy re-encryption scheme. With the proposed PRE scheme, the proxy and a delegatee cannot collude to transfer decryption rights, and if PKG generates re-encryption key arbitrarily, its master secret could be retrieved. We also introduced two important properties, namely *Non-Key-escrow* and *Non-PKG-despotism*, into the proposed IBE-based 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 retrieving and exposing the important secrets of the party.

In addition, we discussed how to prevent a malicious PKG from generating illegitimate re-encryption keys completely with the help of a tamper-resistant hardware device. One future direction is to explore the possibility of modifying the proposed scheme so that this hardware assumption can be eliminated.

References

1. M. Blaze, G. Bleumer, and M. Strauss.: Divertible protocols and atomic proxy cryptography. In: EUROCRYPT 1998, volume 1403 of LNCS, pp. 127-144, (1998)
2. G. Ateniese, K. Fu, M. Green, S. Hohenberger.: Improved Proxy Re-encryption Schemes with Applications to Secure Distributed Storage. In: 12th Annual Network and Distributed Systems Security Symposium, San Diego, California (2005)
3. Man Ho Au, Q. Huang, Joseph K. Liu, Willy Susilo, Duncan S. Wong and Guomin Yang.: Traceable and Retrievable Identity-based Encryption. In: 6th International Conference on Applied Cryptography and Network Security. Lecture Notes in Computer Science 5037, Springer-Verlag, pp. 94 - 110, (2008)
4. A. Ivan and Y. Dodis.: Proxy cryptography revisited. In: 10th Annual Network and Distributed Systems Security Symposium, (2003)
5. Benoit Libert, Damien Vergnaud.: Tracing Malicious Proxies in Proxy Re-Encryption. In: 2nd international conference on Pairing-Based Cryptography, Egham, UK (2008)
6. T. Matsuo.: Proxy Re-encryption Systems for Identity-Based Encryption. In: 1st International Conference on Pairing-Based Cryptography - Pairing 2007, LNCS 4575, pp. 247-267. Springer-Verlag (2007)
7. X.A. Wang, X.y. Yang, F.G. Li.: On the Role of PKG for Proxy Re-encryption in Identity Based Setting. In: Cryptology ePrint Archive, Report 2008/410. (2008)
8. D. Boneh, X.Boyen.: Efficient Selective-id Secure Identity Based Encryption without Random Oracles. In: Advances in Cryptology-EUROCRYPT 2004. LNCS 3027, pp. 223-238. Springer, (2004)
9. R. Sakai, M. Kasahara.: ID based cryptosystems with pairing on elliptic curve. Cryptology ePrint Archive, Report 2003/054. (2003)
10. X.A. Wang, X.y. Yang.: Identity based broadcast encryption based on one to many identity based proxy re-encryption. In 2nd IEEE International Conference on Computer Science and Information Technology, pp.47-50, (2009)
11. X.A. Wang, X.y. Yang.: Proxy re-encryption scheme based on BB2 identity based encryption. In: 2nd IEEE International Conference on Computer Science and Information Technology, pp.134-137, (2009)
12. X.A. Wang, X.y. Yang.: Proxy Re-encryption Scheme Based on SK Identity Based Encryption. In: 5th International Conference on Information Assurance and Security. pp.657-660, (2009)
13. K. Niu, X.A. Wang, M.Q. Zhang.: How to Solve Key Escrow Problem in Proxy Re-encryption from CBE to IBE. In: 1st International Workshop on Database Technology and Applications. pp.95-98, (2009)
14. X.A. Wang, X.y. Yang, M.Q. Zhang.: Proxy Re-encryption Scheme from IBE to CBE," 1st International Workshop on Database Technology and Applications. pp.99-102, (2009)
15. Gentry, C.: Practical identity-based encryption without random oracles. In: EUROCRYPT 2006. LNCS, vol. 4004, pp. 445-464. Springer, Heidelberg (2006)
16. Goyal, V.: Reducing trust in the PKG in identity based cryptosystems. In: Menezes, A. CRYPTO 2007. LNCS, vol. 4622, pp. 430-448. Springer, Heidelberg (2007)
17. Camenisch, J., Damgard, I.: Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In: Okamoto,T. ASIACRYPT 2000. LNCS, vol. 1976, pp. 331-345. Springer, Heidelberg(2000)
18. Smart card, `http://en.wikipedia.org/wiki/Smart_card`