2008 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing

Simple Certificateless Signature with Smart Cards

Kazumasa Omote and Atsuko Miyaji
School of Information Science,
Japan Advanced Institute of
Science and Technology (JAIST)
Email: {omote,miyaji}@jaist.ac.jp

Kazuhiko Kato
Department of Computer Science,
Graduate School of ISE,
University of Tsukuba
Email: kato@cs.tsukuba.ac.jp

Abstract

A certificateless public key cryptosystem (CL-PKC) was proposed for the first time in 2003, and since then it has been attracted. But in fact most CL-PKC is implemented using the pairing technique. So it is difficult so far to apply CL-PKC to an existing encryption system that uses the conventional encryption method such as RSA. Needless to say, it is also difficult in even the system with smart cards. In this paper we propose the signature scheme with the RSA based smart card in the simple CL-PKC. Our scheme does not have user's public key certificate as well as the ID-based cryptography. Also, it has some advantages that there is no escrow problem which occurs in the ID-based cryptography, and that even the manager cannot generate user's falsified signature without changing user's public key.

1. Introduction

The public key cryptosystem is used by various scenes such as the Internet. In a general public key cryptosystem, it is necessary to confirm whether or not the public key is legitimate. The public key certificate is used to verify the validity of the public key. Such a mechanism is called PKI. However, it is pointed out that the management of the public key certificate may become complex in the PKI.

On the other hand, as the first method that does not use the public key certificate, the ID-based public key cryptosystem (ID-PKC) was proposed by Shamir in 1985 [1]. In the ID-PKC the public key need not be verified because user's public key is the public information that specifies the user individuals such as the mail address and the telephone number. The private key used with ID-PKC is generated and safely distributed to each user by the private key generator (PKG). But in ID-PKC the escrow problem is pointed out. This is a problem that PKG can personate all users because PKG knows all private keys of users. Of course, a conventional public key cryptosystem does not have such an escrow problem.

A certificateless public key cryptosystem (CL-PKC) was proposed for the first time by Al-Riyami and Paterson in 2003 [2] to solve the above-mentioned escrow problem in ID-PKC. The CL-PKC does not have the public key certificate as well as the ID-PKC. The key generation center (KGC) generates user's partial private key. In addition, user's private key is made by adding user's confidential information to a partial private key of each user. Therefore the escrow problem is solved in CL-PKC because user's private key is not known to KGC. Hence it is said that CL-PKC is the middle scheme of conventional PKC and ID-PKC.

Recently, a lot of papers about CL-PKC have been proposed and in fact most CL-PKC is implemented using the pairing technique. However the public key cryptosystem applied with a lot of systems is still conventional method such as RSA. So it is difficult so far to apply CL-PKC to an existing encryption system. Needless to say, it is also difficult in even the system with smart cards.

In this paper we propose the signature scheme of the simple CL-PKC with the RSA based smart card. Concretely, as the signature method in CL-PKC, we combine the RSA signature using smart cards and Shamir's ID-based signature. Our scheme does not have user's public key certificate. Also, it has some advantages that there is no escrow problem which occurs in the ID-based cryptography, and that even the manager cannot generate user's falsified signature without changing user's public key.

The organization of this paper is as follows. In the next section we organize related works. After this we state our building blocks in Section 3, our scheme in detail in Section 4 and the experiment of our scheme in Section 5. We discuss our scheme in Section 6 and finally summarize this paper in Section 7.

2. Related work

As the public key cryptosystem that doesn't use the public key certificate, there are a lot of papers such as the

978-0-7695-3492-3/08 $25.00 © 2008 IEEE
DOI 10.1109/EUC.2008.162

ID-based public key cryptosystem (ID-PKC), certificateless public key cryptography (CL-PKC), certificate based encryption and self-certified key.

In 1985, ID-PKC was proposed for the first time by Shamir [1]. This is RSA based ID-PKC. Because ID-PKC uses the ID instead of the public key, it needs not use the public key certificate that guarantees the link of the public key and user's ID. There are a lot of papers about ID-PKC, especially recently, most ID-PKC is implemented using the pairing technique. In the other RSA based ID-PKC, there are Guillou-Quisquater signature [3] and Mediated RSA scheme [4].

On the other hand, CL-PKC was published for the first time in 2003 [2]. This cryptosystem does not have the escrow problem though it does not use the public key certificate. Concretely, the escrow problem is solved by introducing user's partial secret information. Hence it is said that this is the middle method of conventional PKC and ID-PKC. A CL-PKC is mainly composed of Certificateless Encryption (CLE) and Certificateless Signature (CLS). There are a lot of papers about CL-PKC [2][5][6][7][8][9][10], especially recently, most CL-PKC is implemented using the pairing technique in the same way of ID-PKC. Some CL-PKC schemes are implemented based on the discrete logarithm problem without the paring technique [9][10]. Also, there are the papers about the attack that replace user's public key [11] and about discussing the cheating in the master key generation phase of KGC [12].

The self-certified key has been proposed as a similar research of CL-PKC since 1991 [13][14]. In this method anyone can compute user's public key from the public information of the system. In [14] the trusted third party computes to link user's public key with the user's ID. Furthermore, the self-certificated signature [15] is proposed as an applied research of a self-certificated key. This certificate based method can calculate user's public key from the public information of the system when the signature is verified.

Besides, the certificate based encryption [16] was proposed in 2003 as a related research of CL-PKC. In this method the concept is similar to CL-PKC though it seems to be opposite to CL-PKC judging from the name. The public key certificate used here is treated as one of the private keys, and is different from the conventional public key certificate.

3. Building Blocks

3.1. Notations

The notations used in this paper are as follows.

U_i	:	User i
ID_i	:	U_i's identifier
(e, d, n)	:	RSA keys of PKG/KGC
(e_i, d_i, n_i)	:	RSA keys of U_i
pw_i	:	U_i's access password
psk_i	:	U_i's partial secret key
sk_i	:	U_i's secret key
pk_i	:	U_i's public key
$\phi(\cdot)$	:	Euler totient function
t_i	:	Ticket of U_i
m	:	Message
$Sig_i(\cdot)$	:	Signature of U_i
$h(\cdot)$	:	Hash function (e.g. SHA1)

3.2. Shamir-IBS Scheme

Here, we briefly explain Shamir-IBS (Identity-Based Signature) [1]. Shamir-IBS is an application of the RSA scheme, and the signature scheme to regard user's ID as a public key.

1. PKG (used as a trusted third party in ID-PKC) computes each private key $k_i = h(ID_i)^d \pmod{n}$ based on ID_i and transfers k_i to U_i in the secure channel. Note that ID_i is public information that specifies the user individuals such as the mail address or the telephone number, and $d \in Z^*_{\phi(n)}$ is a private key of PKG.
2. U_i generates a random number $r_i \in Z^*_n$, and calculates the following signature (Y_i, Z_i) to the message m. Note that $h(\cdot)$ is a hash function that PKG opens to the public.
$$\begin{aligned} Y_i &= r_i^e \pmod{n} \\ Z_i &= k_i \cdot r_i^{h(Y_i||m)} \pmod{n} \end{aligned}$$
3. The verifier can verify the signature (Y_i, Z_i) by using ID_i, (e, n), m and $h(\cdot)$ as follows.
$$Z_i^e \stackrel{?}{=} h(ID_i) \cdot Y_i^{h(Y_i||m)} \pmod{n}$$

3.3. Signature System with eLWISE Card

The eLWISE card is a smart card (IC card) made in NTT communications Ltd. In the signature with contact-type eLWISE card, we can use the RSA signature, the DSA signature, and the ECDSA signature. In this paper we use the RSA signature from these, whose padding method is RSASSA-PKCS1-v1_5. The RSA key is computed in the smart card. One example of the signature system with eLWISE card is shown as follows (refer to Fig. 1).

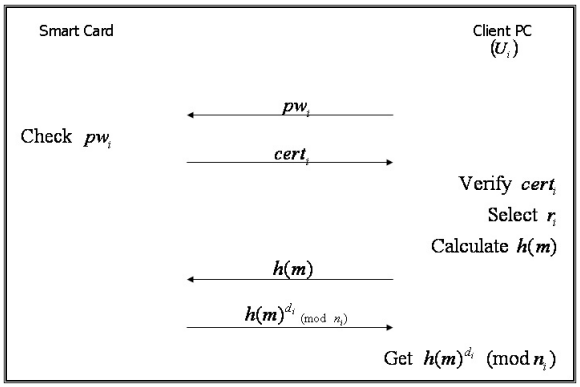

Figure 1. Example of signature system with eLWISE card (RSA method)

1. U_i inputs the access password to the smart card on client PC. When the smart card confirms that this given password is legitimate, it transmits user's certificate $cert_i$ to the client PC.

2. After the smart card confirms that the certificate $cert_i$ is legitimate, the client PC transmits the hash value of a message to the smart card and acquires the RSA signature of the message. Note that the client PC is assumed to have the root certificate and the revocation list.

The smart card is tamper resistant. So it is possible to safely store some confidential information into the smart card. Also, even the trusted third party cannot extract the secret key d_i of RSA from the smart card. However, there are the following some drawbacks in the system which uses the certificate. We set user's RSA key and user's public key certificate into each smart card. So the trusted third party must issue public key certificates only of the number of users, and manage them. She also needs to newly generate the public key certificate every time the user changes own public key. In addition, the size of one public key certificate is about 1KB in our system, and we confirmed that it took a few seconds to transmit the certificate between a smart card and a client PC.

4. Our scheme

4.1. Problems to Solve

A conventional public key cryptosystem has a problem that the management of the public key certificate becomes complex in the system. Of course, it includes the system with eLWISE card with the certificate. On the other hand, the ID-based public key cryptosystem like Shamir's method has an escrow problem that the manager can make a forged signature though it has an advantage not to use a certificate. In our scheme we solve above two problems at the same time, and develop the certificateless public cryptosystem by combining the RSA signature using smart cards and ID-based signature. We use Shamir-IBS as one of the ID-based RSA signature.

4.2. Premise

The premise of our scheme is as follows.

1. The smart card is tamper resistant, and the confidential information is not stolen from the smart card itself or while it is being transferred between the smart card and the client PC.

2. The calculation algorithm (e.g., signature generation) in the smart card is not falsified.

3. Nobody can read user's RSA private key in the smart card.

4. Only the person who knows the access password of the smart card can read the confidential information except for the RSA private key stored in the smart card.

5. The access password of the smart card is not cracked because the smart card is locked when a user inputs a wrong password more than the predetermined frequency.

6. We do not put the confidential information about the key in the client PC because the client PC is not safe.

4.3. Adversary Model

In [2], two kinds of attacks (Type I / Type II) are assumed. Type I adversary can replace the public key of an arbitrary user with another public key though he does not know the master key of KGC. On the other hand, Type II adversary can use the master key though he does not replace the public key of an arbitrary user with another. In this paper we assume these two kinds of attacks as well as [2]. Hence we do not assume the adversary who both replaces user's public keys with another and knows the master key.

4.4. Our Protocol

We concretely show details of the protocol according to the procedure of a certificateless signature. We also show the whole figure of protocol in Fig. 2. Note that the part of the shadow in this figure means the area secretly preserved. In our protocol we newly set the "Change-User-Keys" phase.

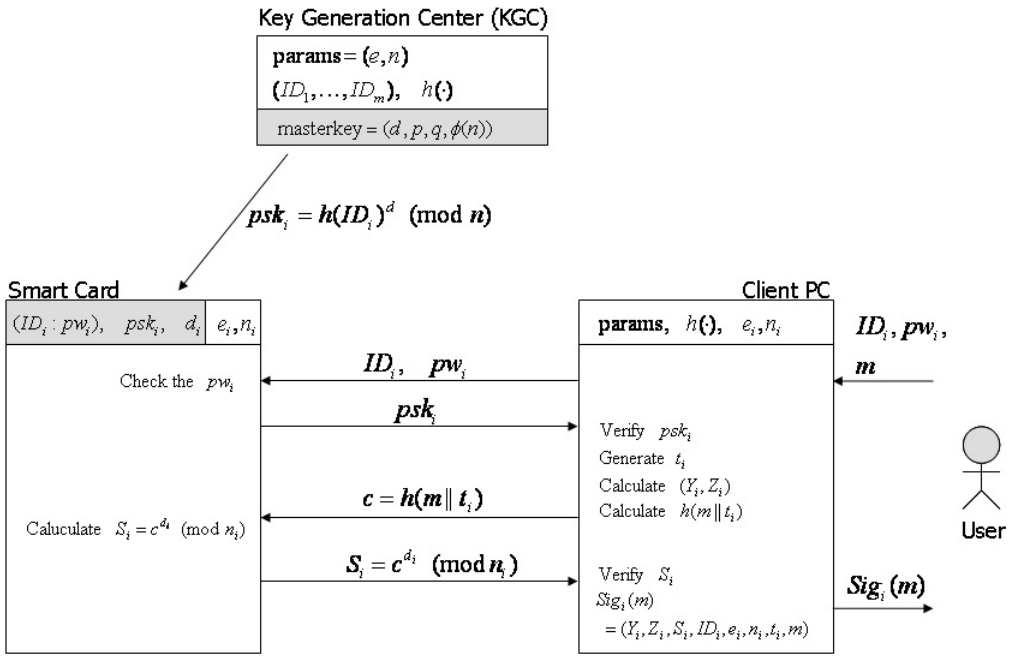

Figure 2. Flow of our protocol

[Setup] KGC selects large primes p and q, computes $n = pq$ and computes (e, d) which satisfies $ed = 1 \pmod{\phi(n)}$. Then KGC publishes **params** $= (e, n)$ as the public parameter and keeps master key **masterkey** $= (d, p, q, \phi(n))$ secretly. He also publishes all users' ID and hash functions.
[Partial-Secret-Key-Extract] KGC calculates $psk_i = h(ID_i)^d \pmod{n}$ by using her own parameters (d, n), and then U_i stores psk_i in his smart card by a safe means.
[Set-Secret-Value] U_i generates RSA keys (e_i, d_i, n_i) using his smart card and preserves these keys in his smart card, sets both ID_i and pw_i, and then copies his public keys (e_i, n_i) to the client PC.
[Set-Secret-Key] U_i sets $sk_i = (psk_i, d_i)$ as secret keys.
[Set-Public-Key] U_i sets $pk_i = (ID_i, e_i, n_i)$ as public keys.
[Sign] At first, U_i gets psk_i from his smart card under the following procedure. U_i inputs both ID_i and pw_i on user's client PC. After confirming that ID_i and pw_i are legitimate in his smart card, psk_i is transferred from his smart card to the client PC. Then U_i verifies psk_i by using **params** and the hash function on the client PC as follows.

$$psk_i^e \stackrel{?}{=} h(ID_i) \pmod{n}$$

Secondly, U_i generates the signature using his smart card under the following procedure. U_i generates a ticket t_i. The ticket is similar to information described in the public key certificate (e.g. the signature issuer and the date of issue). U_i calculates (Y_i, Z_i) as Shamir-IBS on the client PC. It is a signature to the value which joins (e_i, n_i) with t_i. Furthermore, U_i generates the RSA signature S_i to the hash value which joins m and t_i by using his smart card. Hence the signature of U_i is as follows.

$$\begin{aligned} Sig_i(m) &= (Y_i, Z_i, S_i, ID_i, e_i, n_i, t_i, m) \\ Y_i &= r_i^e \pmod{n} \\ Z_i &= psk_i \cdot r_i^{h(Y_i||e_i||n_i||t_i)} \pmod{n} \\ S_i &= h(m||t_i)^{d_i} \pmod{n_i} \end{aligned}$$

[Verify] The verifier can verify $Sig_i(m)$ using **params** and hash function as follows.

$$\begin{aligned} Z_i^e &\stackrel{?}{=} h(ID_i) \cdot Y^{h(Y_i||e_i||n_i||t_i)} \pmod{n} \\ S_i^{e_i} &\stackrel{?}{=} h(m||t_i) \pmod{n_i} \end{aligned}$$

[Change-User-Keys] When U_i wants to change his own public key, he generates new RSA keys (e_i', d_i', n_i') in his smart card and overwrites the old RSA keys.

5. Experiment

5.1. Purpose

Our scheme uses a smart card with a low processing ability. So the processing time that the smart card executes might become large. Especially, we worry about the influence on both data transfer time from smart card to the client

PC and the processing time of the RSA signature. We therefore implemented our scheme and measured both the transfer time and the processing time in order to confirm that the smart card can complete the processing within a reasonable time.

5.2. Circumstances

We used as the client PC a ThinkPad X60 (CPU: Core 2 Duo 2GHz, Memory: 1GB), used as the smart card an eLWISE (NTT Communications), and used as the smart card reader an AES drive IIIE (Athena Smartcard Solutions). This smart card is equipped with a CPU, RAM and ROM and corresponds to PKCS#11. The software we used was the OS Linux Fedora Core 6, a smart card library group, the encryption library OpenSSL 0.9.8b, the multiple-precision arithmetic library GMP 4.1.4-9.

5.3. Measurement Items

In the generation of the signature using the smart card, we measured the transfer time and the processing time for the six items in Table 1. Note that we measured items 1 and 2 by using the scheme with a certificate in order to compare with our scheme. Items 1, 3 and 6 were executed in the smart card, and the others were executed in the client PC.

5.4. Results

Each of the times for the items listed in Table 1 is the average of five measurements. The measurement was conducted by inserting the gettimeofday function in the source code. Especially large values of the measurement results were items 1, 3 and 6 as we expected, and all these items were processed in the smart card. The measurement time of $cert_i$ transfer was longest among them. Note that we adopted a mail address as user's ID and used information about the signature issuer, affiliation, signature date and usage as a ticket. The results of the measurement items 1 and 3 were the transfer time from the smart card to the client PC. On the other hand, the result of measurement item 6 contains not only time of the generation of RSA signature but also the processing transfer time of the signature data. We can guess that the calculation processing time of the RSA signature with the smart card would become approximately 100ms because the size of RSA signature is the same as the size of psk_i (i.e. $420 - 321 \simeq 100$). Therefore, we understood that the data transfer between smart card and the client PC made more significant impact on the time of entire processing than the signature calculation in this smart card.

Also, we confirmed that the entire processing time was able to be shortened by not using $cert_i$ because the transfer time of $cert_i$ is larger than psk_i. Hence we were able to show the effectiveness of our scheme that does not use $cert_i$.

Table 1. Experimental Results

Measurement times	Time
1. Transfer of $cert_i$	2140 ms
2. Verification of $cert_i$	19.4 ms
3. Transfer of psk_i	321 ms
4. Verification of psk_i	0.222 ms
5. Calculation of (Y_i, Z_i)	2.20 ms
6. Generation of RSA Signature	420 ms

6. Discussion

6.1. Certificateless

The certificateless in CL-PKC stands for an implicit public key certificate. Of course, it is not defined as an explicit public key certificate in the system. The public key certificate is a signature to the predefined format data such as ID, the public key and the expiration date. It is generated by the trusted third party. We can regard psk_i as an implicit public key certificate in our scheme because psk_i is the signature which is generated by the trusted third party. Also, we can regard (Y_i, Z_i) as a self-generated public key certificate because (Y_i, Z_i) is the signature to his public key by U_i.

6.2. Escrow Problems

The escrow problem is the problem that the trusted third party can personate all users since she knows all users' secret keys. As for secret key $sk_i = (psk_i, d_i)$ in our scheme, the trusted third party and U_i generates psk_i and d_i, respectively. As a result, our scheme can solve the escrow problem because the trusted third party does not know one of sk_i, that is, d_i.

6.3. Attacks

Two kinds of attacks (Type I / Type II) are assumed in this paper as described in section 4.3. First of all, Type I attack is discussed. This attack is also introduced as a key replacement attack in [11]. Type I adversary attempts to generate a forged signature by replacing user's public key to the advantageous value without knowing user's partial secret key. When the adversary replaces U_i's public key n_i in our scheme, it is necessary to calculate new (Y_i, Z_i) to n_i. To compute new (Y_i, Z_i) the adversary has to use psk_i. However, Type I adversary does not know user's psk_i.

Hence we can say that our scheme is resistant to Type I attack (the key replacement attack) because Type I adversary cannot make U_i's forged signature.

On the other hand, Type II adversary attempts to generate a forged signature using a master key without replacing user's public key to another. Type II adversary has to know not only psk_i but also U_i's secret key d_i in order to generate U_i's forged signature. However, it is difficult for Type II adversary to know d_i due to the difficulty of the factorization on prime numbers of n_i. Hence we can say that our scheme is resistant to Type II attack because Type II adversary cannot generate U_i's forged signature.

Therefore, we can say that our scheme is resistant to two kinds of attacks (Type I / Type II).

6.4. Change of User's Key

When user wants to change his key in a conventional public key cryptosystem, it is necessary to make a new public key certificate. In our scheme the user can freely alter user's keys (e_i, d_i, n_i). Concretely U_i who wants to change his keys generates new keys (e_i', d_i', n_i') and has only to replace (e_i, d_i, n_i) with (e_i', d_i', n_i') in his smart card. Then U_i has only to execute the "Set-Secret-Value" phase in our scheme. As a result, the certificate reissue procedure becomes unnecessary.

7. Summary

In this paper we proposed a simple certificateless signature scheme that combined the ID base signature and the RSA signature. This method is appropriate for an existing RSA based smart card. We described that our scheme does not have an escrow problem, and is resistant to two kinds of attacks (Type I / Type II). As a future work we would like to prevent user's secret key psk_i from getting out of the smart card.

References

[1] A. Shamir, "Identity-based cryptosystems and signature schemes," Advances in Cryptology – CRYPTO'84, LNCS 0196, pp.47–53, Springer-Verlag, 1984.

[2] S.S. Al-Riyami and K. Paterson, "Certificateless public key cryptography," Advances in Cryptology – ASIACRYPT'03, LNCS 2894, pp.452–473, Springer-Verlag, 2003.

[3] L.C. Guillou and J.J. Quisquatar, "A paradoxical identity-based signature scheme resulting from zero-knowledge," Advances in Cryptology – CRYPTO'88, LNCS 403, pp.216–231, Springer-Verlag, 1988.

[4] X. Ding and G. Tsudik, "Simple identity-based encryption with mediated RSA," Proc. RSA Conference'03, 2003.

[5] B. Libert and J.J. Quisquater, "On constructing certificateless cryptosystems from identity based encryption," Proc. PKC'06, LNCS 3958, pp.474–490, Springer-Verlag, 2006.

[6] J.K. Liu, M.H. Au and W. Susilo, "Self-Generated-Certificate Public Key Cryptography and Certificateless Signature/Encryption Scheme in the Standard Model," Proc. ASIACCS'07, pp.273–283, ACM, 2007.

[7] W.S. Yap, S.H. Heng and B.M. Goi, "An efficient certificateless signature scheme," Proc. EUC Workshops'06, LNCS 4097, pp.322–331, Springer-Verlag, 2006.

[8] M.C. Gorantla and A. Saxena, "An efficient certificateless signature scheme," Proc. CIS'05, LNCS 3802, pp.104–109, Springer-Verlag, 2005.

[9] J. Lai and K. Kou, "Self-generated-certificate public key encryption without pairing," Proc. PKC'07, LNCS 4450, pp.476–4890, Springer-Verlag, 2007.

[10] J. Baek, R. Safavi-Naini and W. Susilo, "Certificateless Public Key Encryption without Pairing," Proc. ISC'05, LNCS 3650, pp.134–148, Springer-Verlag, 2005.

[11] B.C. Hu, D.S. Wong, Z. Zhang and X. Deng, "Key replacement attack against a generic construction of certificateless signature," Proc. ACISP'06, LNCS 4058, pp.235–246, Springer-Verlag, 2006.

[12] M.H. Au, Y. Mu, J. Chen, D.S. Wong J.K. Liu and G. Yang, "Malicious KGC attack in certificateless cryptography," Proc. ASIACCS'07, pp.302 - 311, ACM, 2007.

[13] M. Girault, "Self-certified public keys," Advances in Cryptology – EUROCRYPT'91, LNCS 0547, pp.491-497, Springer-Verlag, 1991.

[14] H. Petersen and P. Horster, "Self-certified keys concepts and applications," Proc. CMS'97, pp.102–116, 1997.

[15] B. Lee and K. Kim, "Self-certified Signatures," Progress in Cryptology – INDOCRYPT'02, LNCS 2551, pp.199–214, Springer-Verlag, 2002.

[16] C. Gentry, "Certificate-based encryption and the certificate revocation problem," Advances in Cryptology – EUROCRYPT'03, LNCS 2656, pp.272–293, Springer-Verlag, 2003.