Cryptanalysis of Self-Generated-Certificate Public Key Encryption without Pairing in PKC07

Xu An Wang Xiaoyuan Yang Yiliang Han

Key Laboratory of Information and Network Security
Department of Electronic Technology, Engineering College of Chinese Armied Police Force
Xi'an 710086
Wangxahq@yahoo.com.cn

Abstract: In PKC07, Junzuo Lai and Weidong Kou proposed a self-generated-certificate public key encryption without pairing scheme. In this paper, we show that this scheme cannot resist man-in-the-middle attack. We further point out the reason for successfully attacking is binding the user's secret key with the **multiply** of partial public key from KGC and user's self-generated public key instead of binding with partial public key from KGC and user's self-generated public key **independently**. At last, we give a rescue SGC-PKE scheme by giving little change to Lai and Kou's scheme which can resist this attack.

Keywords: Certificateless public key cryptography, Self-generated-certificate public key encryption, Man-in-the-middle attack.

1. Introduction

In traditional Public Key Cryptography (PKC), each user selects his own private key and computes the corresponding public key. If a user wants to send an encrypted message to other user, he needs to know the user's public key. However, it is easy to suffer from the man-in-the-middle attack. There is a need to provide an assurance to the user about the relationship between a public key and the identity (or authority) of the holder of the corresponding private key. In a traditional Public Key Infrastructure, This assurance is delivered in the form of certificate, essentially a signature by a Certification Authority (CA) on a public key. However, a PKI faces with many challenges in the practice, such as revocation, storage and distribution of certificates. Identity-Based Public Key Cryptography (ID-PKC), first proposed by Shamir [12], solves the problem of authenticity of keys in a different way to traditional PKI. In ID-PKC, a user's public key is derived directly from its identity, for example, an IP address belonging to a network host, or an e-mail address associated with a user. Private keys are generated for entities by a trusted third party called a Private Key Generator (PKG). The only disadvantage of ID-PKC is an unconditional trust to the PKG, which results that PKG can impersonate any user, or decrypt any ciphertext.

In order to solve for the above problem, Certificateless Public Key Cryptography (CL-PKC) was introduced by Al-Riyami and Paterson [2, 3]. It is a new paradigm which lies between Identity-Based Cryptography and traditional Public Key Cryptography. The concept is to eliminate the inherent key-escrow problem of Identity-Based Cryptography (IBC). At the same time, it preserves the attractive

advantage of IBC which is the absence of digital certificates (issued by Certificate Authority) and their important management overhead. Different from IBC, the user's public key is no longer an arbitrary string. Rather, it is similar to the public key used in the traditional PKC generated by the user. A crucial difference between them is that the public key in CL-PKC does not need to be explicitly certified as it has been generated using some partial private key obtained from the trusted authority called Key Generation Center (KGC). Note here that the KGC does not know the user's private keys since they contain secret information generated by the users themselves, thereby removing the escrow problem in IBC [4, 5, 6, 7, 8, 9, 10].

It seems that CL-PKC can solve the problem of explicit certification. Nevertheless it suffers Denial-of-Decryption (DoD) Attack called by Liu and Au [2, 3].Suppose Alice wants to send an encrypted message to Bob. She takes Bob's public key and his identity (or personal information) as input to the encryption function. However, Carol, the adversary, has replaced Bob's public key by someone's public key. Although Carol cannot decrypt the ciphertext, Bob also cannot decrypt the message while Alice is unaware of this. This is similar to Denial of Service (DoS) Attack in the way that the attacker cannot gain any secret information but precluding others from getting the normal service. Liu and Au [2, 3] propose a new paradigm called Self-Generated-Certificate Public Key Cryptography (SGC-PKC) to defend the above attack while preserving all advantages of Certificateless Public Key Cryptography. Similar to CL-PKC, every user is given a partial secret key by the KGC and generates his own secret key and corresponding public key. In addition, he also needs to generate a certificate using his own secret key. The purpose of this self-generated certificate [11] is similar to the one in traditional PKC. That is, to bind the identity (or personal information) and the public key together. The main difference is that, it can be verified by using the user's identity and public key only and does not require any trusted party. It is implicitly included in the user's public key. If Carol uses her public key to replace Alice's public key (or certificate), Bob can be aware of this and he may ask Alice to send him again her public key for the encryption.

Liu and Au proposed the first SGC-PKE scheme in [2, 3], which defends the DoD attack that exists in CL-PKE. In PKC07, Junzuo Lai and Weidong Kou proposed a self-generated-certificate public key encryption without pairing scheme, which is the second SGC-PKE scheme. In this paper, we show that this scheme cannot resist a man-in-the-middle attack. We further point out the reason for successfully attacking is binding the user's secret key with the multiply of partial public key from KGC and user's public key instead of binding with partial public key from KGC and user's public key independently.

We organize the paper as following. In section 2, we give the definition and security notions for SGC-PKE. In section 3, we review the SGC-PKE scheme proposed by Lai and Kou in PKC07 [1]. In section 4, we give the man-in-the-middle attack and propose a rescue scheme which can resist this attack. We give our conclusion in section 5.

2. Definition and Security Notions for SGC-PKE

Definition 1 (Certificateless Public Key Encryption). A generic Certificateless Public Key Encryption scheme, denoted by Π, consists of the following algorithms:

- **Setup:** is a probabilistic polynomial time (PPT) algorithms run by a Key Generation Center (KGC), given a security parameter k as input, outputs a randomly chosen master secret mk and a list of public parameter param.We write **(mk, param) = Setup(k).**
- **UserKeyGeneration:** is PPT algorithm, run by the user, given a list of public parameters param as inputs, outputs a secret key **sk** and a public key **pk**. We write **(sk, pk) = UserKeyGeneration (param).**
- **PartialKeyExtract:** Taking **param**, **mk**, a user's identity **ID** and **pk** received from the user, the KGC runs this PPT algorithm to generate a partial private key $\mathbf{D_{ID}}$ and a partial public key $\mathbf{P_{ID}}$. We write **(P_{ID}, D_{ID}) = PartialKeyExtract (param, mk, ID, pk)**.
- **SetPrivateKey:** Taking **param**, $\mathbf{D_{ID}}$ and **sk** as input, the user runs this PPT algorithm to generate a private key $\mathbf{SK_{ID}}$. We write $\mathbf{SK_{ID}}$ **= SetPrivateKey (param, D_{ID}, sk)**.
- **SetPublicKey:** Taking **param**, $\mathbf{P_{ID}}$ and **pk** as input, the user runs this PPT algorithm to generate a public key $\mathbf{PK_{ID}}$. We write $\mathbf{PK_{ID}}$ **= SetPublicKey(param, P_{ID}, pk)**.
- **Encrypt:** Taking a plaintext **M**, list of parameters **param**, a receiver's identity **ID** and $\mathbf{PK_{ID}}$ as inputs, a sender runs this PPT algorithm to create a ciphertext **C**. We write **C = Encrypt (param, ID, PK_{ID}, M)**.
- **Decrypt:** Taking **param**, $\mathbf{SK_{ID}}$, the ciphertext **C** as inputs, the user as a recipient runs this deterministic algorithm to get a decryption δ, which is either a plaintext message or a "Reject" message. We write δ **= Decrypt (param, SK_{ID}, C)**.

Security Model. According to the original scheme in [2], there are two types of adversaries. Type I adversary does not have the KGC's mater secret key but it can replace public keys of arbitrary identities with other public keys of its own choices. It can also obtain partial and full secret keys of arbitrary identities.Type II adversary know the master secret key (hence it can compute partial secret key by itself). It is still allowed to obtain full secret key for arbitrary identities but is not allowed to replace public keys at any time.

Definition 2 (IND-CCA Security). A Certificateless Public Key Encryption scheme Π is IND-CCA secure if no PPT adversary A of Type I or Type II has anon-negligible advantage in the following game played against the challenger:

1. The challenger takes a security parameter k and runs the **Setup** algorithm. It gives A the resulting system parameters **param**. If A is of Type I, the challenger keeps the master secret key **mk** to itself, otherwise, it gives **mk** to A.
2. A is given access to the following oracles:

- **Public-Key-Request-Oracle:** on input a user's identity **ID**, it computes **(sk, pk) = UserKeyGeneration (param)** and **(P_{ID}, D_{ID}) =PartialKeyExtract (param, mk, ID, pk).** It then computes $\mathbf{PK_{ID}}$ **=SetPublicKey (param, P_{ID}, pk)** and

returns it to A.

- **Partial-Key-Extract-Oracle:** on input a user's identity **ID** and **pk**, it computes **(P_{ID}, D_{ID}) = PartialKeyExtract (param, mk, ID, pk)** and returns it to A. (Note that it is only useful to Type I adversary.)
- **Private-Key-Request-Oracle:** on input a user's identity **ID**, it computes **(sk, pk)=UserKeyGeneration (param) and (P_{ID}, D_{ID}) =PartialKeyExtract (param, mk, ID, pk).** It then computes **SK_{ID} =SetPrivateKey (param, D_{ID}, sk)** and returns it to A. it outputs $\perp$ if the user's public key has been replaced (in the case of Type I adversary.)
- **Public-Key-Replace-Oracle:** (For Type I adversary only) on input identity and a valid public key, it replaces the associated user's public key with the new one.
- **Decryption-Oracle:** on input a ciphertext and an identity, returns the decrypted plaintext using the private key corresponding to the current value of the public key associated with the identity of the user.

3. After making oracle queries a polynomial times, A outputs and submits two message (M_0, M_1), together with an identity ID^* of uncorrupted secret key to the challenger. The challenger picks a random bit $\beta \in \{0, 1\}$ and computes C^*, the encryption of M_β under the current public key PK_{ID^*} for ID^*. If the output of the encryption is $\perp$, then A immediately losses the game. Otherwise C^* is delivered to A.
4. A makes a new sequence of queries.
5. A outputs a bit β'. It wins if $\beta' = \beta$ and fulfills the following conditions:

-At any time, ID^* has not been submitted to **Private-Key-Request-Oracle.**

-In Step (4), C^* has not been submitted to **Decryption-Oracle** for the combination (ID^*, PK_{ID^*}) under which M_β was encrypted.

-If it is Type I, ID* has not been submitted to both **Public-Key-Replace-Oracle** before Step (3) and **Partial-Key-Extract-Oracle** at some step.

Define the guessing advantage of A as

$$Adv_{cle}^{ind-cca}(A) = |\Pr[\beta' = \beta] - \frac{1}{2}|.$$

The definition of SGC Encryption is same as the definition of CL-encryption given in Definition 1, except for **SetPublicKey** in which the user generates a certificate using his own secret key.

For security, in addition to IND-CCA, we require the scheme to be DoD-Free, which is formally defined as follow as a game played between the challenger and a PPT adversary (DoD Adversary), which has the same power of a Type I adversary defined in CL-encryption.

Definition 3 (DoD-Free Security). A SGC Encryption scheme is DoD-Free secure if no PPT adversary A has a non-negligible advantage in the following game played against the challenger:

1. The challenger takes a security parameter k and runs the **Setup** algorithm. It gives A the resulting systems parameters **param.** The challenger keeps the master secret key **mk** to itself.
2. A is given access to **Public-Key-Request-Oracle, Partial-Key-Extract-Oracle, Private-Key-Request-Oracle and Public-Key-Replace-Oracle.**

3. After making oracle queries a polynomial times, A outputs a message M^*, together with an identity ID^* to the challenger. The challenger computes C^*, the encryption of M* under the current public key PK_{ID^*} for ID^*. If the output of the encryption is $\perp$, then A immediately losses the game. Otherwise it outputs C^*.

4. A wins if the following conditions are fulfilled:

- The output of the encryption in Step (3) is not $\perp$.

-Decrypt (param, SK_{ID^*}, C^*) = M^*.

- At any time, ID^* has not been submitted **to Partial-Key-Extract-Oracle.**

Define the advantage of A as

$$Adv_{SGCE}^{DoD-Free}(A) = \Pr[A\ wins]$$

3. Lai and Kou's SGC-PKE scheme

Setup: Generate two large primes p and q such that $q \mid p-1$. Pick a generator g of Z_p^*. Pick $x \in Z_q^*$ uniformly at random and compute $y = g^x$. Choose hash functions H_1: $\{0,1\}^* \times Z_p^* \to Z_q^*$, $H_2 : \{0,1\}^{l_0} \times \{0,1\}^{l_1} \to Z_q^*$ and $H_3 : Z_p^* \to \{0,1\}^l$, where $l = l_0 + l_1 \in N$. Return **param =** (p,q,g,y,H_1,H_2,H_3) and **mk =** (p,q,g,x,H_1,H_2,H_3).

UserKeyGeneration: Pick $z \in Z_q^*$ at random and compute $u = g^z$, Return $(sk, pk) = (z,u)$.

PartialKeyExtract: Taking **param, mk**, **ID** and **pk** as input, it outputs $(P_{ID}, D_{ID}) = (\omega = g^s,\ t = s + xH_1(ID, \omega * pk) = s + xH_1(ID, \omega u))$.

SetPrivateKey: outputs $SK_{ID} = sk + D_{ID} = z + t$.

SetPublicKey: Except for taking **param, P_{ID}** and **pk** as input, it includes ID and SK_{ID} as inputs. Chooses a new hash function $H_0 : \{0,1\}^* \times Z_p^* \times Z_p^* \times Z_p^* \to Z_q^*$, then computes $PK_{ID}^1 = pk * P_{ID} = \mu\omega$ and $PK_{ID}^2 = pk * P_{ID} * y^{H_1(ID, pk * P_{ID})} = \mu\omega y^{H_1(ID, pk * P_{ID})} = g^{z+t} = g^{SK_{ID}}$. Next, it does the following performances to sign the user's identity ID and PK_{ID}^1, PK_{ID}^2 using the user's private key SK_{ID} and Schnnor's signature scheme [13]. (1) Choose a random $r \in Z_q^*$, (2) compute $R = g^r \bmod p$ (3) set the signature to be (R, σ), where $\sigma = r + SK_{ID} * H_0(ID, PK_{ID}^1, PK_{ID}^2, R)$. Finally, returns $PK_{ID} = (PK_{ID}^1, PK_{ID}^2, (R, \sigma))$.

Encrypt: Parses PK_{ID} as $(PK_{ID}^1, PK_{ID}^2, (R, \sigma))$ .If $PK_{ID}^2 \neq PK_{ID}^1 * y^{H_1(ID, PK_{ID}^1)}$ or $g^\sigma \neq R * (PK_{ID}^2)^{H_0(ID, PK_{ID}^1, PK_{ID}^2, R)}$ it returns $\perp$, else pick $\sigma \in \{0,1\}^{l_1}$ at random, and compute $r = H_2(M, \sigma)$. Compute $C = (C_1, C_2)$ such that $C_1 = g^r$, $C_2 = H_3((\mu\omega y^{H_1(ID, \omega\mu)})^r) \oplus (M \| \sigma) = H_3((PK_{ID}^2)^r) \oplus (M \| \sigma)$.

Decrypt: Parse C as (C_1, C_2) and SK_{ID} as (z,t). Compute $M \| \sigma = H_3((c_1)^{z+t}) \oplus c_2$. If $g^{H_1(M,\sigma)} = c_1$, return M. Else return "Reject".

Firgure1 Lai and Kou's SGC-PKE scheme

4. Attack on Lai and Kou's SGC-PKE scheme and a rescue scheme

Note that in Lai and Kou's SGC-PKE scheme, the SK_{ID} binds with the multiply of partial public key from KGC and user's self-generated public key instead of with partial public key from KGC and user's self-generated public key independently. We can explore this shortcoming to give a **man-in-the-middle attack**. We attack the target user when he generates his privatekey and public key. First the attacker corrupts the target ID and gets his key $(sk, pk) = (z, u)$. Then the attacker pretends to be a user with identity "ID" for KGC and pretends to be the KGC for the target user. He can always control the target user's privatekey be equal to his privatekey which is definitely insecure.

1 The attacker gets the target ID's user's key $(sk, pk) = (z, u)$ where $u = g^z$ by corruption the target ID or via **UserKeyGeneration Oracle.**

2 The attacker generates his own key $(sk', pk') = (z', u') = (z', g^{z'})$

3 The attacker pretends to be the target ID to the KGC, and then he gets the partial key $(P_{ID}, D_{ID}) = (\omega = g^s,\ t = s + xH_1(ID, \omega * pk') = s + xH_1(ID, \omega u'))$ via the **PartialKey Extract Oracle (param, mk, ID, pk').**

4 The attacker computes his private key $SK_{ATTACKER} = sk + D_{ID} = z' + t$

5 The attacker pretends to be the KGC to the target ID, and he sets $(P^*_{ID}, D^*_{ID}) = (\omega^* = \frac{\omega u'}{g^z}, SK_{ATTACKER} - z)$. Send it as the result of **PartialKeyExtract (param, mk, ID, pk)** to the target ID**.**

6 The target ID checks whether equation $g^{D^*_{ID}} = \omega^* \bullet y^{H_1(ID, \omega^* * PK)}$ holds. If it holds, he computes his own private key $SK_{TARGET} = sk + D^*_{ID} = z + SK_{ATTACKER} - z = SK_{ATTACKER}$, else reject.

7 Thus the attacker can control the target ID's Privatekey be equal to his Privatekey and decrypt all the ciphertext sends to the target ID.

Firgure2 Man-in-the-middle attack

First we verify the equation $g^{D^*_{ID}} = \omega^* \bullet y^{H_1(ID, \omega^* * PK)}$ always holds

$$
\begin{aligned}
g^{D^*_{ID}} &= g^{SK_{ATTACKER} - z} \\
&= g^{z' + t - z} \\
&= g^{z' + s + xH_1(ID, \omega u') - z} \\
&= g^{z' + s + xH_1(ID, \omega^* \bullet g^z) - z} \\
&= g^{z' + s + xH_1(ID, \omega^* \bullet pk) - z} \\
&= \frac{\omega u'}{g^z} \bullet y^{H_1(ID, \omega^* \bullet pk)} \\
&= \omega^* \bullet y^{H_1(ID, \omega^* \bullet pk)}
\end{aligned}
$$

In our attack, the attacker can access two oracles: Usergeneration Oracle and PartialKey Extract Oracle. Assuming UserKeyGeneration Oracle's output is (sk, pk), we must note that query to the PartialKey Extract Oracle is (param,

mk, ID, pk') instead of (param, mk, ID, pk). Otherwise our attack is a trivial attack.

Actually, we just need give little change to the Lai and Kou's scheme to resist this attack. In the new scheme, the SK_{ID} binds with partial public key from KGC and user's self-generated public key independently, so the man-in-the-middle attack can not work any more. Following is the rescue scheme.

Setup: Generate two large primes p and q such that $q \mid p-1$. Pick a generator g of Z_p^*. Pick $x \in Z_q^*$ uniformly at random and compute $y = g^x$. Choose hash functions $H_1: \{0,1\}^* \times Z_p^* \times Z_p^* \to Z_q^*$, $H_2: \{0,1\}^{l_0} \times \{0,1\}^{l_1} \to Z_q^*$ and $H_3: Z_p^* \to \{0,1\}^l$, where $l = l_0 + l_1 \in N$. Return **param =** $(p, q, g, y, H_1, H_2, H_3)$ and **mk =** $(p, q, g, x, H_1, H_2, H_3)$.

UserKeyGeneration: Pick $z \in Z_q^*$ at random and compute $u = g^z$, Return $(sk, pk) = (z, u)$.

PartialKeyExtract: Taking **param, mk**, **ID** and **pk** as input, it outputs $(P_{ID}, D_{ID}) = (\omega = g^s,\ t = s + xH_1(ID, \omega, pk) = s + xH_1(ID, \omega, u))$.

SetPrivateKey: outputs $SK_{ID} = sk + D_{ID} = z + t$.

SetPublicKey: Except for taking **param, P_{ID}** and **pk** as input, it includes ID and SK_{ID} as inputs. Chooses a new hash function $H_0: \{0,1\}^* \times Z_p^* \times Z_p^* \times Z_p^* \to Z_q^*$, then computes $PK_{ID}^1 = (pk, P_{ID}) = (\mu, \omega)$ and $PK_{ID}^2 = pk * P_{ID} * y^{H_1(ID, pk, P_{ID})} = \mu\omega y^{H_1(ID, pk, P_{ID})} = g^{z+t} = g^{SK_{ID}}$. Next, it does the following performances to sign the user's identity ID and PK_{ID}^1, PK_{ID}^2 using the user's private key SK_{ID} and Schnnor's signature scheme. (1) Choose a random $r \in Z_q^*$, (2) compute $R = g^r \bmod p$ (3) set the signature to be (R, σ), where $\sigma = r + SK_{ID} * H_0(ID, PK_{ID}^1, PK_{ID}^2, R)$. Finally, returns $PK_{ID} = (PK_{ID}^1, PK_{ID}^2, (R, \sigma))$.

Encrypt: Parses PK_{ID} as $(PK_{ID}^1, PK_{ID}^2, (R, \sigma))$, parse PK_{ID}^1 as (μ, ω). If $PK_{ID}^2 \neq \mu\omega * y^{H_1(ID, \mu, \omega)}$ or $g^\sigma \neq R * (PK_{ID}^2)^{H_0(ID, PK_{ID}^1, PK_{ID}^2, R)}$ it returns $\perp$, else pick $\sigma \in \{0,1\}^{l_1}$ at random, and compute $r = H_2(M, \sigma)$. Compute $C = (C_1, C_2)$ such that $C_1 = g^r$, $C_2 = H_3((\mu\omega y^{H_1(ID, \omega\mu)})^r) \oplus (M \| \sigma) = H_3((PK_{ID}^2)^r) \oplus (M \| \sigma)$.

Decrypt: Parse C as (C_1, C_2) and SK_{ID} as (z, t). Compute $M \| \sigma = H_3((c_1)^{z+t}) \oplus c_2$. If $g^{H_1(M, \sigma)} = c_1$, return M. Else return "Reject".

Firgure3 the rescue scheme

5. Conclusion

In this paper, we show that Lai and Kou's SGC-PKE scheme cannot resist man-in-the-middle attack. We further point out the reason for successfully attacking is binding the user's secret key with the multiply of partial public key from KGC and user's self-generated public key instead of binding with partial public key from KGC and user's self-generated public key independently. We give an improved SGC-PKE

scheme based on Lai and Kou's scheme which can resist this attack. But we note that our scheme has not yet be proven secure in the random oracle, that is our further work.

References

1. Junzuo Lai and Weidong Kou. Self-Generated-Certificate Public Key Encryption Without Pairing. In PKC 07, LNCS 4450, pp. 476-489, Springer-Verlag, 2007.
2. J. K. Liu and M. H. Au. Self-Generated-Certificate Public Key Cryptosystem. Cryptology ePrint Archive, Report 2006/194, 2006.
3. J. K. Liu and M. H. Au. Self-Generated-Certificate Public Key Cryptography and Certificateless Signature/Encryption Scheme in the standard Model. In AsiaCCS 07, pp. 273-283, 2007.
4. S. S. Al-Riyami and K. Paterson. Certificateless public key cryptography. In Proc. ASIACRYPT 2003, LNCS 2894, pp. 452-473, Springer-Verlag, 2003.
5. S. S. Al-Riyami and K. Paterson. Certificateless public key cryptography. Cryptology ePrint Archive, Report 2003/126, 2003.
6. J. Baek, R. Safavi-Naini, and W. Susilo. Certificateless public key encryption withoutpairing. In ISC 05, LNCS 3650, pp. 134-148, Springer-Verlag, 2005.
7. B. Libert and J. Quisquater. On constructing certificateless cryptosystems from identity based encryption. In PKC 2006, LNCS 3958, pp. 474-490, Springer-Verlag,2006.
8. D. H. Yum and P. J. Lee. Generic construction of certificateless encryption. In ICCSA' 04, LNCS 3040, pp. 802-811, Springer-Verlag, 2004.
9. Y. Shi and J. Li. Provable efficient certificateless public key encryption. Cryptology ePrint Archive, Report 2005/287, 2005.
10. Z. Cheng and R. Comley. Efficient certificateless public key encryption. Cryptology ePrint Archive, Report 2005/012, 2005.
11. M. Girault. Self-certified public keys. In Proc. EUROCRYPT 91, LNCS 547, pp. 490-497, Springer-Verlag, 1992.
12. A. Shamir. Identity-based Cryptosystems and Signature Schemes. In Crypto'84,LNCS 196, pp. 47-53, Springer-Verlag, 1984.
13. C. P. Schnorr. Efficient signature generation by smart cards. Journal of Cryptology,Vol. 4, No. 3, pp. 161-174, 1991.