On The Security Of Self-Certified Public Keys

Cheng-Chi Lee[§] Min-Shiang Hwang[‡] I-En Liao[†]

Department of Information & Communication Engineering[§]
Asia University
No. 500, Lioufeng Raod, Wufeng Shiang, Taichung, Taiwan, R.O.C.
Email: cclee@asia.edu.tw

Department of Management Information Systems[‡]
National Chung Hsing University
250 Kuo Kuang Road, 402 Taichung, Taiwan, R.O.C.
Email: mshwang@nchu.edu.tw

Department of Computer Science[†]
National Chung Hsing University
250 Kuo Kuang Road, 402 Taichung, Taiwan, R.O.C.

April 5, 2010

On The Security Of Self-Certified Public Keys

Abstract

Many cryptosystems have been developed to solve the problem of information security and some of the approaches were based on the self-certified public key proposed by Girault. In Girault's scheme, the public key is computed cooperatively by both the system authority (SA) and the user. One of the advantages is that the public key is able to implicitly authenticate itself without any additional certificates. Another advantage is that the SA is not able to forge a public key without knowing the user's secret key. Despite the advantages of Girault's system, we will prove in this paper that the system still suffers from two weaknesses by an evil user who impersonates the SA and generates a legal signature without knowing the secret key of the SA.

Keyword: Cryptosystem, cryptography, self-certified public key, security, signature.

1 Introduction

Some well-known public key systems have been developed since 1976 [4, 5, 7, 9]. In those systems, each user has two keys, namely, the private key and the public key. The private key is kept secretly by a user, and it is used to provide the legal signature of a message or to decrypt a message sent by another user. The public key is accessible to public through directory lookup, and it is used to verify the validity of a signature or to encrypt a message. Since the public key is published to the public key directory, an adversary can modify the public key of a target user from the public key directory. A public-key authentication is an important research's issue. The purpose of public-key authentication is to verify the public key of a legal user and to prevent public key from being forged.

Three of the most popular schemes for public-key authentication are ID-based scheme [14], certificate-based scheme [8], and self-certified scheme [6]. We briefly review each of them in the following.

In ID-based scheme, a user first chooses his/her own secret key, and then the system authority (SA) generates a public key using the user's identity and the secret key. Since the public key is derived from the user's identity, the direct relation between the identity and the public key makes it impossible for an evil user to forge a public key. In addition, there is no need to store the public key in a public directory. However, this scheme has a drawback that the SA can impersonate a user, since SA knows every user's secret key. In general, public keys are derived from user's identities and secret keys. For example, the public key is equal to $s = ID^d \bmod n$, where ID is user's identity and d is user's secret key. This procedure is generated by SA.

In certificate-based scheme, the public key of a user is generated by the SA and is used as the user's certificate. The process of generating a public key is also known to the public. The difference between ID-based scheme and certificate-based scheme is that the certificate-based scheme has a certificate to verify the public key of a user. The procedure of generating public keys is public. For example, the public is (y, C), where y is user's public key and C is the public key's certificate. Therefore, one can recalculate a user's public key and compare it with the one stored in the SA's system to verify the validity of a public key. These schemes suffers from the same drawback as in the ID-based scheme, namely, the SA is able to impersonate a user by generating a false certificate. In addition, the certificates have to be stored in SA's system which may occupy too much storage space.

The self-certified scheme was developed by Girault to overcome the problems of the above two, in which a user first chooses his/her own secret key, and then the public key is computed using both the user's and SA's secret keys. That is to say, the public key is generated by both of user and SA. If SA doesn't know the user's secret key, SA cannot generate public key. The detail of this procedure can be seen in Section 2. The main feature of this system is that the SA is a trusted parity. The SA is unable to forge a public key. In other words, it makes the SA more trust worthy. Due to such an advantage, this scheme received a lot more attentions than the other schemes did [3, 10, 11, 15, 16]. These schemes also need an SA to help users to sign users' public keys. The public key is computed by using both of the user's and SA's secret keys. Therefore, SA cannot impersonate a user to derive a user's public key. Using the Girault's system, theses schemes can achieve their proposed requirements.

Despite the advantages of Girault's system, Saeednia showed that their system is insecure [12]. Saeednia pointed out that the authority SA can know the users' secret keys if the authority generates modulo n in a special dishonest way. In this paper, we will propose a different cryptanalysis of the system. we assume that the authority is a trusted parity. This paper will show two weaknesses that an evil user who impersonates the SA and generates a legal public key of a user without knowing the secret key of the SA.

The rest of the paper is organized as follows. In the next section, the Girault's self-certified public key is briefly reviewed. The problem of Girault's system is described in Section 3. Finally, we give a few concluding remarks in Section 4.

2 Girault's Self-Certified Public Key System

Girault proposed a self-certified public key system [6] that is based on the RSA cryptosystem [2, 9] and consists of three phases: the initialization, the registration, and the verification phases.

Initialization Phase:

The SA first generates an RSA key pair (e, d) satisfying $e \times d \bmod (p-1)(q-1) = 1$, where p and q are two large primes. Here, e and d denote a public key and secret key of the SA, respectively. Then the SA calculates two integers n and g, where $n = p \times q$ and g is a maximal order in the multiplicative group (Z/nZ)*. After that, the parameters p, q, and d are kept secret by the SA and n, e, and g are open to the public.

Registration Phase:

When a user U_i with identity ID_i, wants to join the system, he/she chooses a secret key s_i and calculates an integer v_i in the following:

$$v_i = g^{-s_i} \bmod n. \tag{1}$$

Next, U_i sends ID_i and v_i to the SA. Upon receiving these messages, the SA calculates a public key p_i for the user by

$$p_i = (v_i - ID_i)^d \bmod n, \tag{2}$$

and then the SA sends p_i back to U_i. Upon receiving p_i, U_i checks the validity of p_i by

$$(p_i^e + ID_i) \bmod n = v_i. \tag{3}$$

If the above equation holds, U_i is certain that p_i is indeed generated by the SA. Note that the public key p_i of a user U_i is generated by SA using both the secret key s_i of U_i and d of SA. However, s_i is unknown to the SA.

Verification Phase:

When a verifier wants to verify the validity of a user's p_i, he/she can follow identity ID_i [1, 13]:

Step 1. U_i sends ID_i and p_i to the verifier, who then calculates $v_i = (p_i^e + ID_i) \bmod n$.

Step 2. U_i selects a random integer r_i, calculates $t_i = g^{r_i} \bmod n$, and then sends t_i to the verifier.

Step 3. The verifier selects a random integer r_v and sends it to U_i.

Step 4. U_i calculates $y_i = r_i + s_i \times r_v$, and sends it to the verifier.

Step 5. Upon receiving y_i, the verifier checks the following equation $(g^{y_i} \times v_i^{r_v}) \bmod n = t_i$. If it holds, the verifier prove that ID_i is valid and p_i was generated by the SA.

Note the name *self-certified* comes from the fact that no certificate is necessary when verifying ID_i and p_i, as the certificate is embedded in the public key itself. An evil user, who has the knowledge of p_i, is highly unlikely to obtain U_i's secret key s_i by solving the discrete logarithm problem [6]. Although the SA can impersonate U_i by generating a false public key p_i' with the use of an arbitrary secret key s_i'. The existence of two or more public keys linked to U_i can prove that the SA has cheated, as the SA can generate only one valid public key for a particular user.

3 Cryptanalysis of Girault's System

Despite the advantages of Girault's self-certified public key system, it is still vulnerable that an evil user who impersonates the SA to sign a legal public key for a user. We propose two weaknesses as follows.

Weakness One:

Step 1. An evil user U_i arbitrarily chooses a random secret key s_i' and calculates $v_i' = g^{-s_i'} \bmod n$, which satisfies $v_i + ID_i = v_i' - ID_i'$, where ID_i' is U_i's another identity.

Step 2. U_i registers his/her another ID_i' to SA, and sends (ID_i', v_i') to SA.

Step 3. SA calculates a legal public key p_i' for the user ID_i' by $p_i' = (v_i' - ID_i')^d \bmod n$ and sends it to U_i.

Step 4. Upon receiving p_i', U_i checks it by Equation (3), who was convinced that p_i' is signed by SA.

Step 5. U_i impersonates SA to sign a legal p_i'' for ID_i'' as follows, where $p_i'' = p_i \times p_i'$ and $ID_i'' = ID_i^2$.

$$
\begin{aligned}
p_i'' &= p_i \times p_i' \bmod n \\
&= (v_i - ID_i)^d (v_i' - ID_i')^d \bmod n \\
&= [(v_i - ID_i)(v_i + ID_i)]^d \bmod n \\
&= (v_i^2 - ID_i^2)^d \bmod n.
\end{aligned} \tag{4}
$$

The secret key of ID_i'' is $2s_i$ which is derived from $v_i'' = v_i^2 \bmod n = g^{-2s_i} \bmod n$. Therefore, U_i can impersonate SA to sign a validity of p_i'' and ID_i''. But, SA does not know that ID_i'' is a forged user.

When a verifier wants to verify the validity of a user's p_i'', he/she can identity ID_i'' in the following verification phase:

Step 1. U_i sends ID_i'' and p_i'' to the verifier, who then calculates $v_i'' = ({p_i''}^e + ID_i'') \bmod n$.

Step 2. U_i selects a random integer r_i, calculates $t_i = g^{r_i} \bmod n$, and then sends t_i to the verifier.

Step 3. The verifier selects a random integer r_v and sends it to U_i.

Step 4. U_i calculates $y_i = r_i + s_i'' \times r_v$, where $s_i'' = 2s_i$, and sends y_i to the verifier.

Step 5. Upon receiving y_i, the verifier checks the following equation: $g^{y_i} \times v_i''^{r_v} \bmod n = t_i$. If it holds, the verifier can prove that ID_i'' is valid and p_i'' was generated by the SA.

It is quite obvious that an evil user U_i can impersonate SA to sign a validity of p_i'' and ID_i''. It also passes the public-key authentication by the above protocol. That is to say, it is possible that this weakness can be performed if a user U_i can find a correct format ID_i' and ID_i'' such that $v_i + ID_i = v_i' - ID_i'$ and $ID_i'' = ID_i^2$.

Weakness Two:

Step 1. An evil user U_i arbitrarily chooses a random secret key s_i' and a public key p_i'.

Step 2. U_i calculates $v_i' = g^{-s_i'} \bmod n$.

Step 3. U_i derives ID_i' by $(p_i'^e + ID_i') \bmod n = v_i'$.

Step 4. U_i keeps the triplet (ID_i', s_i', p_i'), where the ID_i' is U_i's another identity, and (s_i', p_i') is his/her another key pair. Therefore, U_i can impersonate SA to sign a validity of p_i' and ID_i', SA is unknown to that ID_i' is a forged user.

When a verifier wants to verify the validity of a user's p_i', he/she can identity ID_i' in the following verification phase:

Step 1. U_i sends ID_i' and p_i' to the verifier, who then calculates $v_i' = (p_i'^e + ID_i') \bmod n$.

Step 2. U_i selects a random integer r_i, calculates $t_i = g^{r_i} \bmod n$, and then sends t_i to the verifier.

Step 3. The verifier selects a random integer r_v and sends it to U_i.

Step 4. U_i calculates $y_i = r_i + s_i' \times r_v$, and sends y_i to the verifier.

Step 5. Upon receiving y_i, the verifier checks the following equation: $g^{y_i} \times v_i'^{r_v} \bmod n = t_i$. If it holds, the verifier can prove that ID_i' is valid and p_i' was generated by the SA.

It can prove that an evil user U_i can impersonate SA to sign a validity of p_i' and ID_i'. Therefore, the self-certified public key system is vulnerable. That is to say, it is possible that this weakness can be performed if a user U_i can find a correct format ID_i' such that $(p_i'^e + ID_i') \bmod n = v_i'$.

4 Conclusions

We have shown that Girault's self-certified public key system is possible that it has two weaknesses: An evil user can easily impersonate the SA to sign a public key without knowing the secret key of the SA.

References

[1] T. Beth, "A Fiat-Shamir-like authentication protocol for the ElGamal scheme," in *Advances in Cryptology, EUROCRYPT'88*, pp. 77–86, Lecture Notes in Computer Science, 330, 1988.

[2] Chin-Chen Chang and Min-Shiang Hwang, "Parallel computation of the generating keys for RSA cryptosystems," *IEE Electronics Letters*, vol. 32, no. 15, pp. 1365–1366, 1996.

[3] Yun-Shihng Chang, Tzong-Chen Wu, and Shih-Chan Huang, "ElGamal-like digital signature and multisignature schemes using self-certified public keys," *The Journal of Systems and Software*, vol. 50, pp. 99–105, Feb. 2000.

[4] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. IT-22, pp. 644–654, Nov. 1976.

[5] T. ElGamal, "A public-key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. IT-31, pp. 469–472, July 1985.

[6] M. Girault, "Self-certified public keys," in *Advances in Cryptology, EUROCRYPT'91*, pp. 491–497, Lecture Notes in Computer Science, 1991.

[7] Min-Shiang Hwang, Chin-Chen Chang, and Kuo-Feng Hwang, "An ElGamal-like cryptosystem for enciphering large messages," *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 2, pp. 445–446, 2002.

[8] M. Kohnfelder. "A method for certification,". Tech. Rep. MIT Press, Cambridge, MA, MIT Laboratory for Computer Science, 1978.

[9] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," *Communications of the ACM*, vol. 21, pp. 120–126, Feb. 1978.

[10] S. Saeednia, "Identity-based and self-certified key-exchange protocols," in *The Second Australasian Confonference on Information Security and Privacy*, pp. 303–313, Sydney, Australia, June 1997.

[11] S. Saeednia and H. Ghodosi, "A self-certified group-oriented cryptosystem without a combiner," in *Lecture Notes in Computer Science 1587, ACISP 99*, pp. 192–201, Sydney, Australia, 1999.

[12] S. Saeednia, "A note on Girault's self-certified model," *Information Processing Letters* , vol. 86, no. 6, pp. 323–327, 2003.

[13] C. P. Schnorr, "Key distribution system using ID-related information directory suitable for mail systems," in *Proceedings of SECURICOM'90*, pp. 115–122, 1990.

[14] A. Shamir, "Identity based cryptosystems & signature schemes," in *Advances in Cryptology, CRYPTO'84*, pp. 47–53, Lecture Notes in Computer Science, 1984.

[15] Y. M. Tseng and J. K. Jan, "A group signature scheme using self-certified public keys," in *Proceedings of the Ninth National Conference Information Security*, pp. 165–172, Taichung, Taiwan, May 1999.

[16] Hyung-Kyu Yang, Jong-Ho Choi, and Young-Hwa Ann, "Self-certified identity information using the minimum knowledge," in *IEEE TENCON: Digital Signal Processing Applications*, pp. 641–647, 1996.