Comment on the Public Key Substitution Attacks*

Xiaofeng Chen, Fangguo Zhang, and Baodian Wei
(Corresponding author: Xiaofeng Chen)

School of Information Science and Technology, Sun Yat-sen University
Guangzhou 510275, P.R.China (Email: {isschxf,isdzhfg,isdwbd}@zsu.edu.cn)

(Received June 25, 2005; revised and accepted July 25, 2005)

Abstract

In this paper we present a comment on some previous works about the Public Key Substitution Attacks (PKSA in brief). Though there exist some security flaws for the schemes being attacked, we point out that these attacks on them are either trivial or avoidable after a little modification.

Keywords: Public key cryptography, public key substitution attacks, public key authentication, proxy signature, key agreement

1 Introduction

The concept of Public Key Cryptography, introduced by Diffie and Hellman in 1976 [5], may be a milestone in the history of cryptology. In public key systems, each user has two different keys: a public key and a private key. Therefore, the capacities for encryption and decryption are separated. This separation allows important improvement in the key management and makes it possible for digital signature [4].

In public key systems, all public keys are usually stored in a public database called a public key directory. The public key directory is very important yet vulnerable. An intruder can impersonate a target user by substituting his public key with a forged public key, called Public Key Substitution Attacks (PKSA in brief). Therefore, the public key should be verified to prevent this attack before communicating. Depending on the ways of verifying the public key, the public key systems can be categorized as follows: Certificate-based public key systems [8], Identity-based public key systems [15] and Self-certified public key systems [7].

Though PKSA are dangerous events for public key systems, we found some previous works about PKSA, including attacks on a public key authentication scheme, some proxy signature schemes and Station-to-Station Message Authentication Code (STS-MAC) protocol, are either trivial or avoidable after a little modification.

The rest of the paper is organized as follows: Some preliminaries are given in Section 2. The comment on PKSA on a public key authentication scheme is given in Section 3. The comment on PKSA on some proxy signature schemes is given in Section 4. The comment on PKSA on STS-MAC protocol is given in Section 5. Finally, conclusions will be made in Section 6.

2 Preliminaries

In this section, we will briefly introduce the public key substitution attack in certificate-based public key systems.

In certificate-based public key systems, a highly trusted organization, i.e., Certificate Authority (CA) issues a witness $\mathcal{W}$, often called certificate, which binds the user's identity $\mathcal{I}$ to the public key $\mathcal{P}$. The certificate format usually adopts CCITT X.509 standard [3]. Everyone can be convinced that $\mathcal{P}$ is really the public key of $\mathcal{I}$ by the CA's digital signature. The disadvantages of the systems is the requirement of the additional cost of computation and storage for $\mathcal{W}$ and $\mathcal{P}$.

There exists a security flaw called Public Key Substitution Attacks (in brief, PKSA) for Certificate-based systems. If a dishonest CA wants to impersonate the user whose public key is pk. He can do as follows:

- He randomly chooses a different secret key sk' and computes the corresponding public key pk'.
- He then generates a certificate C' which binds the users' identity information ID with pk'. Obviously, CA can impersonate the user to communicate with others.

However, the user can accuse the dishonest CA because there exist his two different "valid" certificates issued by

*Supported by National Natural Science Foundation of China (No. 60403007)

the CA for a same period T. The arbiter deduces that CA is dishonest because no one could forge CA's signature.

Therefore, Certificate-based systems reach Girault's trusted level 3, *i.e.*, the authority does not know the private key of the users, and it can be proven that the authority generates false witness if he does so. Consequently, we usually omit this security flaw when using Certificate-based systems.

3 PKSA on a Public Key Authentication Scheme

In 2003, Lee, Hwang and Li [10] proposed a new public key authentication scheme for cryptosystems with a trusty server. Their scheme is based on discrete logarithm too, and in their scheme, the certificate of the public key is a combination of user's password and private key. Later, Zhang and Kim [18] showed that Lee-Hwang-Li's key authentication scheme is not secure. From the obtained public information, any one can get the private key of the user. Furthermore, they proposed an improved scheme. Recently, Sun, Cao and Sun [14] showed that Zhang-Kim's proved scheme still suffered from public key substitution attack, *i.e.*, a dishonest user can forge his public key via the verification equation. Therefore, they claimed that Zhang-Kim's scheme cannot achieve the non-repudiation service like the previous variants.

In this section, we point out that their attack is trivial, which is same to PKSA in Certificate-based systems. First of all, we overview Zhang and Kim's scheme and Sun-Cao-Sun's PKSA on it.

3.1 Zhang and Kim's Scheme

The system parameters of the scheme are as follows: Let p and q be prime numbers such that $q|p-1$, g is a generator with order q in Z_p^*. The one-way function f is defined by $f(x) = g^x \bmod p$. The user of the system has Prv as his/her private key and PWD as his/her password. Let $Pub = g^{Prv} \bmod p$ be the user's public key.

- Registration:
  In the registration phase, each user chooses a random number $r \in Z_q^*$, and calculates $f(PWD + r)$. The user then himself generates his public key certificate $C = PWD + r + Prv \times Pub \bmod q$.

  The user then sends $f(PWD + r)$, $R = g^r \bmod p$ and his ID to the server secretly. The server then verifies if $f(PWD + r) = f(PWD) \times R$ and verifies the $f(PWD + r)$ sent by the legal user, and then stores ID and $f(PWD + r)$ in public password table in the server. The public password table cannot be modified or forged by an attacker because the server can use the technique of access control to protect it. The certificate C and public key Pub of the user are opened to the public over the network.

- Authentication:
  In the key authentication phase, when someone wants to communicate with a user, the sender first obtains C, Pub and $f(PWD + r)$ of the receiver from the public directory in the network and public password table in the server, and then checks the certificate C of the public key of the receiver by computing the following equation:

  $$f(C) = f(PWD + r) \times Pub^{Pub} \bmod p.$$

  If and only if the above equation holds, the sender accepts the public-key Pub of the receiver.

3.2 Sun-Cao-Sun's PKSA

In Zhang and Kim's scheme, assume that a dishonest legal user who has a key pair (prv, pub) uses his private key prv to generate his signature for a document. Now anyone can verify this signature using his public key pub. To deny his signature, the dishonest user does as follows:

1) Arbitrarily selects a forged private key prv' such that $prv' \neq prv \bmod q$, and then computes the corresponding forged public key $pub' = g^{prv'} \bmod p$.

2) Computes the forged certificate C' as follows:

$$C' = pwd + r + prv' \times pub' \bmod q.$$

3) Substitutes the fabrication C' and pub' in the public key directory.

Since the dishonest user knows his own password pwd and random number r, he can easily obtain the forged certificate C'. Now anyone will also be convinced of the integrity of the forged public key pub'.

3.3 Comment

Sun, Cao and Sun claimed that the signature generated using prv cannot be verified using the forged public key pub'. As a result, the dishonest user can deny his signature. However, when a dispute occurs, the receiver can send the signature together with the public key pub and corresponding certificate C to the arbiter to accuse the dishonest user. If the certificate C ensures the validity of the public key pub and the signature can be verified by this public key pub, the arbiter can deduce the user dishonest because only the user can generate a valid certificate for himself. Actually, note that the equation $g^C/pub^{pub} = g^{C'}/pub'^{pub'}$ holds, the arbiter can easily deduce the dishonest user generated the forged certificate and corresponding public key.

This is the same as Certificate-based systems. For example, a user generated his key pair (sk, pk) and signed messages with the private key sk, where the validity of pk is ensured by a certificate C issued by a trusted CA. After a period of time T, he updated his key pair (sk', pk') and got a new certificate C' from the CA. Now he uses the

new secret key sk' to generate signatures which can be verified by pk'. However, he still could not deny his any signature signed by the private key sk.

4 PKSA on Some Proxy Signature Schemes

In 2001, Lee, Kim and Kim proposed a non-designated proxy signature scheme [11] and used it to design secure mobile agents in electronic commerce [12]. Many researchers presented the security analysis of this scheme [16, 9, 17]. Since Wang et al's analysis is a general case and the others can be regarded as a special case of it, we only focus on it in this paper.

4.1 Lee-Kim-Kim's Scheme

The system parameters of the scheme are as follows: Let p and q be prime numbers such that $q|p-1$, g is a generator with order q in Z_p^*. Let h be a cryptographic secure hash function. Suppose Alice be the original signer and Bob be the proxy signer. Each entity I has a certified key pair (x_I, y_I), here x_I is the private key and $y_I = g^{x_I}$ is the corresponding public key.

- Proxy Key Generation:
  The original signer Alice uses the Schonrr scheme [13] to sign a warrant m_w, which specifies which messages Bob can sign on behalf of Alice, the validity period of delegation, etc. That is, Alice chooses a random number $k_A \in Z_q$, computes $r_A = g^{k_A} \bmod p$ and $s_A = k_A + x_A + x_A h(m_w, r_A) \bmod q$. Then, the tuple (m_w, r_A, s_A) is sent to the proxy signer Bob, and Bob checks its validity by $g^{s_A} = y_A^{h(m_w, r_A)} \cdot r_A \bmod p$.

  If this verification is correct, Bob sets his proxy key pair (x_P, y_P) as $x_P = s_A + x_B \bmod q$, $y_P = g^{x_P} \bmod p = y_A^{h(m_w, r_A)} \cdot r_A \cdot y_B \bmod p$.

- Proxy Signature Generation:
  With the proxy key pair (x_P, y_P), Bob can use any DLP-based signature scheme to generate proxy signature on any delegated message m. The resulting proxy signature is a tuple $\sigma = (sign(m, x_P), m_w, r_A, y_A, y_B)$.

- Proxy Signature Verification:
  To check the validity of σ, a verifier first checks whether the message m conforms to the warrant m_w. He then computes the proxy public key $y_P = y_A^{h(m_w, r_A)} \cdot r_A \cdot y_B \bmod p$. Finally, the verifier checks whether $sign(m, x_P)$ is a valid signature on message m with respect to the proxy public key y_P in the corresponding DLP-based signature scheme.

4.2 Wang-Bao-Zhou-Deng's PKSA

The original signer Alice can mount the following attack:

- Create a warrant $\bar{m}_w$, select a random number $c \in Z_q^*$, and then define $\bar{r}_A = y_B^{-1} g^c \bmod p$.
- The forged proxy key pair $(\bar{x}_P, \bar{y}_P)$ is given by:

  $$\bar{x}_P = c + x_A h(\bar{m}_w, \bar{r}_A) \bmod q, \quad \bar{y}_P = g^{\bar{x}_P} \bmod p.$$

- Anyone can directly verify that $(\bar{x}_P, \bar{y}_P)$ is a valid proxy key pair with respect to $(\bar{m}_w, \bar{r}_A)$, since $\bar{y}_P = g^{\bar{x}_P} \bmod p = y_A^{h(\bar{m}_w, \bar{r}_A)} \cdot \bar{r}_A \cdot y_B \bmod p$.

4.3 Comment

In certificate-based systems, the validity of the users' public key pk is ensured by the signature σ of CA. In Lee, Kim and Kim's proxy signature scheme, note that the proxy public key $y_P = y_A^{h(m_w, r_A)} \cdot r_A \cdot y_B \bmod p = g^{s_A} \cdot y_B \bmod p$, i.e., the validity of y_P is ensured by the signature (r_A, s_A) of the original signer.

When a dispute between the original signer and the proxy signer occurs, the proxy signer presents the object proxy signature $\sigma = (sign(m, x_P), m_w, r_A, y_A, y_B)$ to the arbiter. The arbiter first computes the corresponding proxy public key y_P, and then checks whether $sign(m, x_P)$ is a valid signature on message m with respect to y_P. If the check is negative, he claims that the original signer is dishonest. Otherwise, he asks the original signer to provide s_A which satisfies $g^{s_A} = y_A^{h(m_w, r_A)} \cdot r_A \bmod p$. If the original signer can not provide such a value, he deduces that the original signer is dishonest. Otherwise, he deduces that the proxy signer generated the signature. This requires the original signer to keep the value s_A, just as CA should keep the users' certificate in Certificate-based systems. We argue that it should be an implicit condition though this is never mentioned in the Lee-Kim-Kim's scheme. On the other hand, it is not desirable in situations where the storage is small.

Furthermore, wang et al. showed that the revised version of Lee-Kim-Kim's scheme [9] also suffered from the PKSA by the dishonest original signer. Our above comment is also suitable for this attack.

5 PKSA on Key Agreement Protocol

It is well-known that PKSA can also be used in key agreement protocol. First, we introduce this attack on Station-to-Station Message Authentication Code (STS-MAC) protocol.

5.1 PKSA on STS-MAC Protocol

Let A is the initiator and B responder. In the attack against the responder, the adversary E registers A's public key P_A as its own; i.e., $P_E = P_A$. When A sends B message (1), E intercepts it and replaces the identity I_A with I_E. E then passes message (2) from B to A

unchanged. Finally E intercepts message (3), and replaces $Cert_A$ with $Cert_E$. Since $P_A = P_E$, we have $S_A(\alpha^{r_A}; \alpha^{r_B}) = S_E(\alpha^{r_A}; \alpha^{r_B})$. Hence B accepts the key K and believes that K is shared with E, while in fact it is shared with A. Note that E does not learn the value of K. The attack is depicted below. The notation $A \hookrightarrow B$ means that A transmitted a message intended for B, which was intercepted by the adversary E and not delivered to B.

1) $A \hookrightarrow B : I_A, \alpha^{r_A}$

1') $E \hookrightarrow B : I_E, \alpha^{r_A}$

2) $E \hookleftarrow B : Cert_B, \alpha^{r_B}, S_B(\alpha^{r_B}, \alpha^{r_A}), MAC_K(S_B(\alpha^{r_B}, \alpha^{r_A}))$

2') $A \hookleftarrow E : Cert_B, \alpha^{r_B}, S_B(\alpha^{r_B}, \alpha^{r_A}), MAC_K(S_B(\alpha^{r_B}, \alpha^{r_A}))$

3) $A \hookrightarrow B : Cert_A, S_A(\alpha^{r_A}, \alpha^{r_B}), MAC_K(S_A(\alpha^{r_A}, \alpha^{r_B}))$

3') $E \hookrightarrow B : Cert_E, S_A(\alpha^{r_A}, \alpha^{r_B}), MAC_K(S_A(\alpha^{r_A}, \alpha^{r_B}))$

Similarly, E can perform an attack against the initiator A. For details, see [1].

5.2 Comment

The reason why PKSA occur in STS-MAC protocol is that CA issued two "valid" certificates which correspond to a same public key. The possibility for two different users to choose the same private key is negligible. Consequently, the public key for each user should be different.

It is obvious that this attack can be prevented by requiring that each entity proves to CA possession of a private key corresponding to his public key during the certificate issuing process [1]. Actually, in CCITT X.509 standard, a user randomly chooses his secret key sk, and computes the corresponding public key pk. He then sends pk to the authority and proves to the authority that he knows sk without revealing it. If the proof holds, the authority issued a certificate to the user. Therefore, PKSA in STS-MAC protocol can always be avoidable.

6 Conclusions

In this paper we present a comment on some previous works about the Public Key Substitution Attacks, including attacks on a public key authentication scheme, some proxy signature schemes and STS-MAC protocol. Though there exist some security flaws for the schemes being attacked, we point out that these attacks on them are either trivial or avoidable with a little modification.

References

[1] S. Blake-Wilson and A. Menezes, "Unknown key-share attacks on the station-to-station (STS) protocol," in *PKC 99*, LNCS 1560, pp. 154–170, Springer-Verlag, 1999.

[2] S. Blake-Wilson and A. Menezes, "Entity authentication and authenticated key agreement protocols employing asymmetric techniques," in *Proceeding of Security Protocols Workshop'97*, LNCS 1361, pp. 137–158, Springer-Verlag, 1997.

[3] *The Directory-Authentication Framework*, CCITT Recommendation X.509.

[4] W. Diffie, "The first ten years of public-key cryptography," in *Proceeding of IEEE*, vol. 76, no. 5, pp. 560–577,1988.

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

[6] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, 1985.

[7] M. Girault, "Self-certified public keys," *Advances in Cryptology-Eurocrypt 1991*, LNCS 547, pp.490–497, Springer-Verlag, 1991.

[8] M. Kohnfelder, *A Method for Certification*, Technical Report (MIT Laboratory for Computer Science), MIT Press, Cambridge, MA, 1978.

[9] J. Lee, J. Cheon, and S. Kim, "An analysis of proxy signatures: is a secure channel necessary?," *CT-RSA 2003*, LNCS 2612, pp. 68–79, Springer-Verlag, 2003.

[10] C. C. Lee, M. S. Hwang, L. H. Li, "A new key authentication scheme based on discrete logarithms," Applied Mathematics and Computation, vol. 139, pp. 343–349, 2003.

[11] B. Lee, H. Kim, and K. Kim, "Strong proxy signature and its applications," in *Proceedings of the 2001 Symposium on Cryptography and Information Security (SCIS 2001)*, vol. 2/2, pp. 603-608, Oiso, Japan, Jan. 23-26, 2001.

[12] B. Lee, H. Kim, and K. Kim, "Secure mobile agent using strong non-designated proxy signature," in *ACISP 2001*, LNCS 2119, pp. 474–486, Springer-Verlag, 2001.

[13] C. Schnorr, "Efficient signature generation by smart cards," *Journal of Cryptography*, vol. 4, no. 3, pp. 161-174, 1991.

[14] D. Sun, Z. Cao, and Yu Sun, "Comment: cryptanalysis of Lee-Hwang-Li's key authentication scheme," *Applied Mathematics and Computation*, in press.

[15] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Advances in Cryptology-Crypto 1984*, LNCS 196, pp. 47–53, Springer-Verlag, 1984.

[16] H. M. Sun and B. T. Hsieh, *On the security of some proxy signature schemes*, Available at http://eprint.iacr.org/2003/068.

[17] G. Wang, F. Bao, J. Zhou, and R. H. Deng, "Security analysis of some proxy signatures," in *ICISC 2003*, LNCS 2971, pp. 305–319, Springer-Verlag, 2004.

[18] F. Zhang and K. Kim, "Cryptanalysis of Lee-Hwang-Li's key authentication scheme," *Applied Mathematics and Computation*, vol. 161, no. 1, pp. 101–107, 2005.

Xiaofeng Chen is an Associate Professor in the Department of Computer Science at Sun Yan-sen University, Guangzhou, China. He obtained his Ph.D. degree in Cryptography from School of Communication Engineering, Xidian University in 2003. His main research interests include public key cryptography and E-commerce security.

Fangguo Zhang is an Associate Professor in the Department of Electronics and Communication Engineering at Sun Yan-sen University in Guangzhou, China. He obtained his Ph.D. degree in Cryptography from School of Communication Engineering, Xidian University in 2001. His main research interests include elliptic curve cryptography, pairing-based cryptosystem and its applications.

Baodian Wei is now working in the Department of Electronics and Communication Engineering at Sun Yan-sen University in Guangzhou, China. He obtained his Ph.D degree from School of Information and Communication Engineering, Xidian University in 2004. His research interests include cryptology and its applications.