Remarks on Some Proxy Signature Schemes

Sung-Ming Yen[1)] Chung-Pei Hung[1)] Yi-Yuan Lee[2)]

[1)] Dept of Electrical Engineering, Tamkang University
Tamsui, Taipei Hsien, Taiwan 25137, R.O.C.
E-mail: yensm@csie.ncu.edu.tw

[2)] Communication Network Lab.
Institute for Information Industry, Taiwan, R.O.C.

Abstract

In 1996, a new category of signature scheme called a proxy signature was proposed by Mambo, Usuda, and Okamoto. The proxy signature scheme allows a designated person, called a proxy signer, to sign on behalf of an original signer. In Mambo's paper, one kind of cryptanalysis on their scheme was considered. In this paper, it will be shown that the reported attack in Mambo's paper is not practical and a simple countermeasure can be easily developed. The proxy signature scheme plays the role in many practical applications and receives great attention after it was proposed. In 1999, Sung and Hsieh developed an enhanced version of proxy signature scheme. However, it will be proven that the Sung-Hsieh scheme is not secure. A simple modified version will be suggested.

1 Introduction

Public key based signature schemes, e.g., [1, 2, 3], are developed to enable a signer to produce the signature for a message by using his private key. To check the validity of the signature, the corresponding public key (verifying key) of a signer should be employed. An interesting problem was considered in 1996 by Mambo, Usuda, and Okamoto [4] in which a designated person will be assigned to produce a signature on behalf of an original signer when he will be absent. This new category of signature scheme is called the proxy signature.

The proxy signature scheme plays the role in many practical applications and receives great attention after it was proposed. Related works can be found in the literature [5, 6, 7, 8, 9]. Also, the concept of proxy signature was independently pointed out by Yen [10, §3.4] in 1994. So far, there have been five categories of proxy signature schemes, each with different level of delegation and security assumption. The *full delegation*, the *partial delegation*, and the *delegation with warrant* were proposed by Mambo, Usuda, and Okamoto [4]. Later on, Zhang [6] suggested two other modifications, i.e., the *partial delegation with warrant* and the *threshold delegation*.

Brief description of the above five proxy signatures are given below.

Full delegation: In a full delegation, a proxy signer is given the same secret that the original signer has. So, a proxy signer can produce exactly the same signature as the original signer can do.

Partial delegation: In a partial delegation, a proxy signing key σ will be created by the original signer. A proxy signer then uses σ to sign messages on behalf of the original signer.

Delegation by warrant: A signed warrant can be explicitly included in the delegation. It is used to claim the regulation of a delegation.

Partial delegation with warrant: In a partial delegation with warrant, a proxy signing key σ will be created and a warrant will be signed both by the original signer.

Threshold delegation: In a threshold delegation, a set of n proxy signers are given shares such that at least $t \leq n$ shares are required to produce a proxy signature. It is called a (t, n)-threshold delegation.

A partial delegation and a delegation by warrant are more secure than the full delegation. A partial delegation with warrant combines both the advantages of a partial delegation and a delegation by warrant. In general, a partial delegation with warrant provides the characteristics of acceptable performance and a reasonable way to regulate the delegation, e.g., a valid delegation period.

The property of *nonrepudiation* of generating a signature is also necessary for a sound proxy signature scheme, like in any conventional signature scheme. In a proxy signature scheme without nonrepudiation, a proxy signer can flame the original signer and vice versa. This is simply because that the original signer can sign on behalf of the proxy signer. So far, a number of proxy signature schemes with the property of nonrepudiation have been developed. They are called the proxy-protected proxy signature in [4, 5] and are called the nonrepudiable proxy signature in [6, 7].

Among the above nonrepudiable proxy signature schemes, the Mambo-Usuda-Okamoto scheme [4] and the Kam-Park-Won scheme [5] have been shown to be insecure by Sun and Hsieh [8]. Also, the Zhang's scheme [6] has been shown to be insecure by Lee, Hwang, and Wang [7].

In [8], Sun and Hsieh also suggested an enhanced proxy signature scheme based on both the Mambo-Usuda-Okamoto and the Kam-Park-Won schemes. In this paper, we will examine the security of the Sung-Hsieh scheme and will prove that the scheme is not nonrepudiable. A slightly modified version will be suggested.

In Mambo's paper [4], one kind of cryptanalysis on their scheme was considered. In this paper, it will be shown that the reported attack in Mambo's paper is not practical and a simple countermeasure can be easily developed.

2 Remark on the Sung-Hsieh proxy signature

Recently, Sun and Hsieh [8] proposed a modified nonrepudiable proxy signature scheme based on the Kim-Park-Won scheme [5]. In this section, it will be shown that the modified scheme is not nonrepudiable because the original signer can forge a proxy signature key and can sign on behalf of the proxy signer.

2.1 Brief review of the Sung-Hsieh scheme

In the scheme, like in the ordinary Kim-Park-Won scheme, the original signer and the proxy signer select s_o and s_p as their private keys, respectively. The corresponding public keys are $v_o = g^{s_o} \bmod p$ and $v_p = g^{s_p} \bmod p$ where p is a large prime and g is a primitive root modulo p. The scheme is reviewed in the following.

Step 1: The original signer computes $K = g^k \bmod p$ where $k \in_R \mathbb{Z}_{p-1}$. Then, the parameter $e = h(W, K, v_p)$ is computed where W is the warrant of the delegation and $h()$ is a one-way hash function, e.g., MD5 [11] and SHA [12]. Finally, a secret delegation parameter

$$\sigma = s_o \cdot e + k \bmod (p-1)$$

is constructed. The original signer sends $\{W, \sigma, K\}$, called the proxy certificate, to the proxy signer through a secure channel (to protect σ).

Step 2: The proxy signer checks the validity of the proxy certificate by computing

$$g^{\sigma} \stackrel{?}{\equiv} v_o^{h(W,K,v_p)} \cdot K \pmod p.$$

If it holds, the proxy signer computes the proxy signing key as

$$\sigma_p = \sigma + s_p \bmod (p-1). \tag{1}$$

Note that the corresponding public key for verifying signature generated using σ_p is

$$v_p' = g^{\sigma_p} = v_o^{h(W,K,v_p)} \cdot K \cdot v_p \bmod p. \tag{2}$$

Step 3: When the original signer is absent, the proxy signer can sign on behalf of the original signer by using any existing discrete logarithm based signature schemes, e.g., the ElGamal scheme [2] or the Schnorr scheme [3]. Now, $\{M, W, Sign_{\sigma_p}(M), K, v_o, v_p\}$ are sent as the complete signature for message M where $Sign_{\sigma_p}(M)$ means the usual signature using σ_p as the signing key.

Step 4: The signature verifier/receiver first computes the signature verifying key v_p' as in Eq.(2), then checks the correctness (validity) of the signature $Sign_{\sigma_p}(M)$ in the usual approach.

2.2 A forgery attack

In the following, it will be shown that the above proxy signature scheme is not "proxy signer nonrepudiable". This means that the original signer can sign on behalf of the proxy signer and can frame the proxy signer if he wishes.

The original signer chooses random integer $k \in_R \mathbb{Z}_{p-1}$ as usual and computes

$$K' = g^k \cdot v_p^{-1} \bmod p \tag{3}$$

where v_p^{-1} is the multiplicative inverse of v_p modulo p. It can be proven that the original signer can use

$$\tilde{\sigma_p} = s_o \cdot e + k \bmod (p-1)$$

as the forged proxy signing key, where $e = h(W, K', v_p)$. Consequently, the corresponding forged public key (i.e., the signature verifying key) is

$$\tilde{v_p'} = v_o^{h(W,K',v_p)} \cdot g^k \bmod p. \quad (4)$$

The original signer can sign and send $\{M, W, Sign_{\tilde{\sigma_p}}(M), K', v_o, v_p\}$ to the receiver as a valid proxy signature using $\tilde{\sigma_p}$ as the signing key. As described in the Step 4 of the scheme, the signature verifier/receiver will compute the signature verifying key as in Eq.(2) before verifying the received signature. The following manipulation proves that $\tilde{v_p'}$ will be the derived verifying key:

$$\begin{aligned} & v_o^{h(W,K',v_p)} \cdot K' \cdot v_p \pmod p \\ \equiv\ & v_o^{h(W,K',v_p)} \cdot g^k \cdot v_p^{-1} \cdot v_p \pmod p \quad \text{(by Eq.(3))} \\ \equiv\ & \tilde{v_p'} \quad \text{(by Eq.(4))} \end{aligned}$$

2.3 Enhancement of the Sung-Hsieh scheme

In this subsection, an enhanced Sung-Hsieh scheme is suggested. In this improved scheme, it is infeasible for the original signer to forge a valid proxy signing key. The scheme is sketched in the following.

Step 1: It is exactly the same as the original Step-1 except that $e = h(W, K, v_p)$ is replaced by $e = h(W, K, v_o, v_p)$.

Step 2: The proxy signer checks the validity of the proxy certificate by computing $g^{\sigma} \stackrel{?}{\equiv} v_o^{h(W,K,v_o,v_p)} \cdot K \pmod p$. If it holds, the proxy signer computes the proxy signing key as

$$\sigma_p = \sigma + s_p \cdot K \bmod (p-1). \quad (5)$$

In this modified scheme, the corresponding public key for verifying signature generated using σ_p is

$$v_p' = g^{\sigma_p} = v_o^{h(W,K,v_o,v_p)} \cdot K \cdot v_p^K \bmod p. \quad (6)$$

Step 3: It is exactly the same as the original Step-3.

Step 4: The signature verifier/receiver first computes the signature verifying key v_p' as in Eq.(6), then checks the correctness (validity) of the signature $Sign_{\sigma_p}(M)$ in the usual approach.

2.4 Security analysis of the enhanced scheme

The following paragraphs discuss the security issue of the enhanced Sung-Hsieh scheme.

Attack 1. The original signer may try to forge a valid proxy signing key $\tilde{\sigma_p}$ as mentioned previously. However, in the above enhanced scheme, it requires the original signer to select two integers (K', a) such that

$$\tilde{\sigma_p} = s_o \cdot h(W, K', v_o, v_p) + a \bmod (p-1) \quad (7)$$

and

$$K' = (v_p{}^{K'})^{-1} \cdot g^a \bmod p. \quad (8)$$

Note that the corresponding signature verifying key now becomes (computed by the signature verifier)

$$\tilde{v_p'} = g^{\tilde{\sigma_p}} = v_o^{h(W,K',v_o,v_p)} \cdot K' \cdot v_p^{K'} \bmod p.$$

This can be justified by the following derivation:

$$\begin{aligned} & v_o^{h(W,K',v_o,v_p)} \cdot K' \cdot v_p^{K'} \pmod p \\ \equiv\ & v_o^{h(W,K',v_o,v_p)} \cdot (v_p{}^{K'})^{-1} \cdot g^a \cdot v_p^{K'} \pmod p \\ & \qquad \text{(by Eq.(8))} \\ \equiv\ & g^{\tilde{\sigma_p}} \pmod p \quad \text{(by Eq.(7))} \\ \equiv\ & \tilde{v_p'} \end{aligned}$$

However, it is infeasible to solve the problem in Eq.(8). The first approach is that K' is selected first and try to solve the exponent a. However, this is well known as the discrete logarithm hard problem. The second approach is that the exponent a is selected first and try to solve K'. This problem seems to be also infeasible to solve or to be even harder than the discrete logarithm problem.

Attack 2. In the Attack 1, the original signer (attacker) computes her public key in an usual approach. However, in some scenarios, an attacker may intend to forge a proxy signing key at the cost of having her usual private key unknown. Suppose that an attacker, say Alice with public key v_a, as the role of an original signer wishes to forge a proxy signature on behalf of a proxy signer, say Peter with public key v_p. Alice needs to compute v_a, K, and σ_p such that $g^{\sigma_p} \equiv v_a^{h(W,K,v_a,v_p)} \cdot K \cdot v_p^K \pmod p$. One of the possibility is to let

$$v_a = v_p^{-K/h(W,K,v_a,v_p)} \cdot g^a \bmod p \quad (9)$$

and

$$K = g^k \bmod p$$

where a is an integer to determine and $k \in_R \mathbb{Z}_{p-1}$. Therefore, the proxy signing key (for Peter) can be computed as

$$\sigma_p = a \cdot h(W, K, v_a, v_p) + k \bmod (p-1). \quad (10)$$

This can be justified as follows: (we denote $e = h(W, K, v_a, v_p)$)

$$\begin{aligned} & v_a^e \cdot K \cdot v_p^K \pmod p \\ \equiv\ & v_p^{(-K/e)\cdot e} \cdot g^{a\cdot e} \cdot g^k \cdot v_p^K \pmod p \quad \text{(by Eq.(9))} \\ \equiv\ & g^{a\cdot e+k} \pmod p \\ \equiv\ & g^{\sigma_p} \pmod p \quad \text{(by Eq.(10))} \end{aligned}$$

It is interesting to note that the problem raised in Eq.(9) seems to be more difficult than that given in Eq.(8), the reason is simply because of the inclusion of a one-way hash function in Eq.(9). Thus, the above forgery fails to work.

Attack 3. Another scenario of forgery attack is that an attacker, as the role of a proxy signer, may intend to forge a proxy signing key at the cost of having her usual private key unknown. Suppose that an attacker, say Alice with public key v_a, wishes to forge a proxy signing key. Alice needs to compute v_a, K, and σ_a such that $g^{\sigma_a} \equiv v_o^{h(W,K,v_o,v_a)} \cdot K \cdot v_a^K \pmod p$. One of the possibility is to let $v_a = v_o^a \cdot g^b \bmod p$ and $K = v_o^c \cdot g^d \bmod p$ where a, b, c, and d are integers to determine. Therefore, the proxy signing key (for Alice) can be computed as

$$\begin{aligned} \sigma_a = s_o \cdot (h(W, K, v_o, v_a) + a \cdot K + c) \\ +(b \cdot K + d) \bmod (p-1). \end{aligned}$$

Since s_o is unknown to Alice, the following approach shall be a reasonable setting to forge σ_a

$$\begin{cases} h(W, K, v_o, v_a) + a \cdot K + c \equiv 0 \pmod{p-1} \\ b \cdot K + d \equiv \sigma_a \pmod{p-1}. \end{cases} \tag{11}$$

It is infeasible to select a, c, and d first (now K is determined) and try to compute v_a to satisfy Eq.(11), the reason is simply because of the inclusion of a one-way hash function in Eq.(11). Furthermore, even a possible v_a is obtained, the attacker needs to compute b in order to obtain the forged proxy signing key σ_a. However, under this situation, to solve b from $v_a = v_o^a \cdot g^b \bmod p$ is equivalent to solve the discrete logarithm problem. Finally, if the attacker let all $\{a, b, c, d\} \in_R \mathbb{Z}_{p-1}$. Then, it is believed that Eq.(11) will be true with only a negligibly small probability.

3 Remark on the Mambo-Usuda-Okamoto proxy signature

In Mambo's paper [4], a cryptanalysis on their scheme was considered in which it was claimed that a proxy signer could forge another proxy signing key. Therefore, cheating conducted by a proxy signer is possible. In this section, it will be shown that the reported attack in Mambo's paper is not practical (and not really correct) and a simple countermeasure can be easily developed.

3.1 Brief review of the Mambo-Usuda-Okamoto scheme

The original signer and the proxy signer select s_o and s_p as their private keys, respectively. The corresponding public keys are $v_o = g^{s_o} \bmod p$ and $v_p = g^{s_p} \bmod p$ where p is a large prime and g is a primitive root modulo p. The Mambo-Usuda-Okamoto scheme is reviewed in the following.

Step 1: The original signer computes $K = g^k \bmod p$ where $k \in_R \mathbb{Z}_{p-1}$. Then, the proxy signing key $\sigma = s_o + k \cdot K \bmod (p-1)$ is computed. The original signer sends $\{\sigma, K\}$, called the proxy certificate, to the proxy signer through a secure channel.

Step 2: The proxy signer checks the validity of the proxy certificate by computing

$$g^\sigma \stackrel{?}{\equiv} v_o \cdot K^K \pmod p.$$

If it holds, σ is treated as the proxy signing key and the corresponding public key for signature verification is $v_p' = g^\sigma \bmod p$.

Step 3: When required the proxy signer can sign on behalf of the original signer by using any existing discrete logarithm based signature schemes. Now, $\{M, Sign_\sigma(M), K\}$ are sent as the complete signature for message M where $Sign_\sigma(M)$ means the usual signature using σ as the signing key.

Step 4: The signature verifier/receiver first computes the signature verifying key $v_p' = v_o \cdot K^K \bmod p$, then checks the correctness (validity) of the signature $Sign_\sigma(M)$ in the usual approach.

3.2 Remarks on a forgery attack

The following forgery attack was considered in [4]. In the attack, a proxy signer holding a proxy certificate

$\{\sigma, K\}$ (with a corresponding public key v_o) may intend to forge another valid proxy certificate $\{\tilde{\sigma}, \tilde{K}\}$ (with a corresponding public key $\tilde{v_o}$).

The forgery attack [4] works as follows. The proxy signer (attacker) selects a random number $u \in \mathbb{Z}_{p-1}$ and computes $U = g^u \bmod p$. Then the attacker computes

$$\begin{cases} \tilde{v_o} = v_o^U \bmod p \\ \tilde{K} = K \cdot U \bmod p \\ \tilde{\sigma} = (\sigma + uK)U \bmod (p-1). \end{cases}$$

It was claimed that $\{\tilde{\sigma}, \tilde{K}\}$ would be a valid proxy certificate issued by some user with a corresponding public key $\tilde{v_o}$, simply because

$$g^{\tilde{\sigma}} \equiv g^{s_o U + (k+u)KU} \equiv \tilde{v_o} \cdot \tilde{K}^{\tilde{K}} \pmod p.$$

It can be easily verified that the above statement requires

$$\tilde{K}^{KU \bmod (p-1)} \equiv \tilde{K}^{\tilde{K}} \pmod p$$

where $(\tilde{K}^{\tilde{K}} \bmod p) = (\tilde{K}^{KU \bmod p} \bmod p)$.

The above equation can be described in an alternative way as

$$\tilde{K}^{KU \bmod (p-1)} \equiv \tilde{K}^{KU \bmod p} \pmod p. \qquad (12)$$

However, for a very large prime number p of the form $p = 1 + 2q$ (where q is also a large prime), it is easy to avoid the occurrence of Eq. (12).

For the trivial case of $KU < p - 1$, it is evident that $(KU \bmod (p-1)) = (KU \bmod p)$. Therefore, $\tilde{K}^{KU \bmod (p-1)} \equiv \tilde{K}^{KU \bmod p} \pmod p$ is true for any $\tilde{K}$. It is also easy to verify the case of $KU = p - 1$, in which $\tilde{K}^0 \equiv \tilde{K}^{p-1} \pmod p$. However, it is easy to counteract the above forgery by setting $K \geq p/2$, so that the only possible value of U to conduct the attack is one. Note that $U = 1$ implies $u = p - 1$ since g is a primitive root modulo p. Interestingly, the requirement of $U = 1$ and $u = p - 1$ result in $\tilde{v_o} = v_o$, $\tilde{K} = K$, and $\tilde{\sigma} = \sigma$. So, no forgery is possible.

For the nontrivial case of $KU > p - 1$, we first note from the following Lemma 1 that $KU \bmod (p-1) \neq KU \bmod p$.

Lemma 1. *Given two nonzero integers K and U selected from $[1, p-1]$ and that $KU > p-1$. It results in that $KU \bmod (p-1)$ and $KU \bmod p$ are different.*

Proof. Let $KU \bmod (p-1) = R_1$ and $KU \bmod p = R_2$. Therefore, $KU = t_1(p-1) + R_1$ and $KU = t_2 p + R_2$ for two existing integers t_1 and t_2. Note also that $KU \leq (p-1)^2$ since both K and U are less than or equal to $p - 1$.

Suppose that $R_1 = R_2$, then it results in $t_1(p-1) = t_2 p$. Based on $KU > p-1$ and $t_1(p-1) = t_2 p$, it results in that $t_1 = np$ and $t_2 = n(p-1)$ for an integer $n \geq 1$.

However, the substitution of t_1 and t_2 into the representation of KU contradicts $KU \leq (p-1)^2$. This proves that $KU \bmod (p-1) \neq KU \bmod p$. □

In this case, the requirement for a possible proxy certificate forgery is analyzed below. In the following discussions, it is assumed that p $(= 1 + 2q)$ is a very large prime number and q is also a large prime. Let the *order* of $\tilde{K}$ modulo p be r. Denote $\delta = |KU \bmod (p-1) - KU \bmod p|$. It can be obtained from the following Lemma 2 that $\delta \leq p - 2$ if $KU > p - 1$.

Lemma 2. *Given any positive integer $x > p - 1$, then $|x \bmod (p-1) - x \bmod p| \leq p-2$ where p is an integer greater than two.*

Proof. Let $\delta = |x \bmod (p-1) - x \bmod p| = |x \bmod p - x \bmod (p-1)|$. To maximize δ, it needs to maximize $(x \bmod p)$ and at the same time to minimize $(x \bmod (p-1))$.

Suppose that $x \bmod p = p-1$, then $x = tp + (p-1) = (t+1)(p-1) + t$ for an existing integer $t \geq 1$. So, $x \bmod (p-1) = t$ and totally $\delta \leq p - 2$. □

It follows from the Fermat's theorem [13] that $\tilde{K}^{KU \bmod (p-1)} \equiv \tilde{K}^{KU \bmod p} \pmod p$ is true if the order r divides δ, i.e., $r|\delta$. For $p = 1 + 2q$, the possible order modulo p are $\{1, 2, q, 2q\}$ and the number of elements with this listed order are $\{1, 1, \phi(q), \phi(2q) = \phi(q)\}$, respectively. We discard the two special cases of $\tilde{K} = 1$ of order 1 and $\tilde{K} = p - 1$ of order 2 when modulo p; simply because of their special form and it is easy to prevent. For $\tilde{K}$ of order $2q = p - 1$, it is impossible for $\tilde{K}^{KU \bmod (p-1)} \equiv \tilde{K}^{KU \bmod p} \pmod p$ to be true because $2q = p - 1$ never divides δ (recall that $\delta \leq p - 2$ if $KU > p - 1$).

For the last case, $r = q$, the probability of finding a value u (and therefore U) so that $q|\delta$ is negligibly small because δ will be a random integer over $[1, p-2] = [1, 2q-1]$. The only possibility for $q|\delta$ is when $\delta = q$.

This concludes our claim that the forgery attack reported in [4] is not practical and can be prevented easily.

4 Concluding remarks

In this paper, it is shown that the Sun-Hsieh nonrepudiable proxy signature scheme is not sound. A forgery attack is demonstrated in which the original signer can

forge a valid proxy signing key and can sign messages on behalf of the proxy signer.

We also give remarks on a forgery attack described in Mambo's paper on proxy signature. The attack is not practical and simple countermeasures are easy to develop.

5 Acknowledgments

This work was partly supported by the National Science Council of the Republic of China under contract NSC89-2213-E-008-049 and was partly sponsored by MOEA and supported by Institute for Information Industry, R.O.C.

References

[1] R.L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystem," *Commun. of ACM*, Vol.21, No.2, pp.120–126, 1978.

[2] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trains. Inf. Theory* Vol.IT-31, No.4, pp.469–472, 1985.

[3] C.P. Schnorr, "Efficient identification and signatures for smart cards," *Proc. of Crypto '89*, Lecture Notes in Computer Science 435, Springer Verlag, pp.239–252, 1990.

[4] M. Mambo, K. Usuda, E. Okamoto, "Proxy signatures: Delegation of the power to sign messages," *IEICE Trans. Fundamentals* Vol.E79-A, No.9, pp.1338–1354, 1996.

[5] S. Kim, S. Park, and D. Won, "Proxy signatures, revisited," *Information and Communications Security (ICICS '97)*, Lecture Notes in Computer Science 1334, Springer-Verlag, pp.223–232, 1997.

[6] K. Zhang, "Threshold proxy signature schemes," *Proc. of 1997 Information Security Workshop*, Japan, pp.191–197, Sept. 1997.

[7] N.Y. Lee, T. Hwang, and C.H. Wang, "On Zhang's nonrepudiable proxy signature scheme," *Advances in Cryptology ASICCRYPT '98 (ACISP 98)*, Lecture Notes in Computer Science 1438, Springer-Verlag, pp.415–422, 1998.

[8] H.M. Sun and B.T. Hsieh, "Remarks on two nonrepudiable proxy signature schemes," *Proc. of the 9th National Conference on Information Security*, pp.241–246, 1999.

[9] H.M. Sun and B.J. Chen. "Time-stamped proxy signatures with traceable receivers," *Proc. of the 9th National Conference on Information Security*, pp.247–253, 1999.

[10] S.M. Yen, "Design and Computation of Public Key Cryptosystems," Ph.D. thesis, Department of Electrical Engineering, National Cheng Kung University, R.O.C., April 1994.

[11] R. Rivest, "The MD5 message digest algorithm," *RFC 1321*, Apr. 1992.

[12] FIPS 180-1, "Secure Hash Standard," NIST, US Department of Commerce, Washington D.C., April 1995.

[13] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, *Handbook of applied cryptography*, CRC Press, 1997.