

## Short Paper

---

# Secure Key-Evolving for Public Key Cryptosystems Based on the Discrete Logarithm Problem<sup>\*</sup>

CHENG-FEN LU<sup>†</sup> AND SHIUHPYNG SHIEH

<sup>\*</sup>*Department of Computer Science and Information Engineering*

*Ta Hwa Institute of Technology*

*Hsinchu, 307 Taiwan*

*Computer Science and Information Engineering Department*

*National Chiao Tung University*

*Hsinchu, 300 Taiwan*

This paper addresses the security and efficiency of key-evolving protocols in public key encryption and signature schemes, where the security assumption is the intractability of the discrete logarithm problem. We identify the key-independence property as the security goal of key-evolving, so that each periodic secret key is independent of each other. The first protocol operates in  $Z_p^*$  and is efficient for the secret-key holder; the second operates in  $Z_n^*$ , and is efficient for the public-key holder. For both protocols, we provide proofs and analysis for correctness, security and efficiency.

**Keywords:** provable security, discrete logarithm, key management, key evolving, key independence

## 1. INTRODUCTION

Over the past 20 years, public key cryptography has provided many signature and public key encryption schemes. However, if a signing key or decryption key is compromised, it is regarded a total break of the system. To avoid this undesirable situation, one common practice is to assign each key a certain usage period and update the key when entering a new period. Therefore, the security and efficiency of key updating or key-evolving becomes an important topic for key management [1].

Key-evolving, or the synchronized updating of a shared key, is commonly employed for symmetric cryptosystems. When the sender and the receiver of a message share a common master key, they do not use this shared key directly. Instead, they use it to derive a set of sub-keys where each sub-key is valid for a certain period of time or for a specific application [2].

---

Received June 24, 2002; revised November 15, 2002; accepted April 21, 2003.

Communicated by Chi Sung Laih.

<sup>\*</sup> An earlier version of this paper has been published in the RSA Conference 2002, USA. This work is supported in part by Ministry of Education, National Science Council of Taiwan, and Lee & MTI Center, National Chiao Tung University.

<sup>†</sup> This work was done while the first author was with National Chiao Tung University.



For asymmetric cryptosystems, the key-evolving technique was first employed in research on forward-secure signature schemes [3]. Recently, asymmetric key-evolving have been proposed for both signature schemes and public key encryption schemes [4-10]. Asymmetric key-evolving considers the scenario where the sender and receiver do not possess the same master key, but rather some asymmetric system parameters. With the initial parameters, the secret key holder (the signer or the decryptor) and the public holder (the verifier or the encryptor) update their corresponding key without further communication.

Without a key-evolving protocol, certificate retrieval and verification must be performed periodically. With a key-evolving protocol, the certificate retrieval and verification operation is performed only for the initial parameters. Thereafter, both parties update their corresponding public or secret keys for future periods.

The asymmetric key-evolving protocol aims to reduce the damage in case a secret key is compromised. This is also a goal of the research of key management and key agreement [11, 12]. In those papers, the concepts of forward-secrecy, backward-secrecy, and key-independence were defined. Informally, forward-secrecy refers to the situation that the compromise of one or several secret keys does not compromise previous secret keys. Likewise, backward-secrecy refers to the situation that the compromise of one or several secret keys does not compromise future secret keys. Key-independence means that the secret keys used in different periods are basically independent. Thus, even if the attacker discovers the secret key of a certain period, it gives him little advantage in finding the secret keys of other periods.

Two other related properties include perfect forward-secrecy and resistance to known-key attacks. Perfect forward-secrecy assures that the “compromise of long-term keys does not compromise past session keys” [13]. Resistance to known-key attacks assures that the compromise of past session keys will allow neither a passive adversary to compromise future session keys nor an active adversary to forge them [14-16].

Recently, two more related properties such as  $z$ -resilient/key-insulated public encryption schemes or intrusion-resilient signatures have been proposed [6, 9, 10]. The forward-secrecy and backward-secrecy properties in this paper are defined for key-evolving protocols, while  $z$ -resilient/key-insulated or intrusion-resilient properties are defined for specific public encryption schemes or signatures. Therefore, the proposed key-evolving protocols in this paper are applicable to both public key encryption schemes and signature schemes based on the discrete logarithm problem.

This paper is organized as follows. Section 2 describes the model and definitions. Section 3 presents the protocol based the difficulty of computing discrete logarithm in  $Z_p^*$ . Section 4 presents the protocol based on the difficulty of factoring a large composite  $n$  and section 5 concludes this paper.

## 2. MODEL AND DEFINITIONS

The initial public/secret parameter or key base is denoted as  $(PKB, SKB)$ . The public/secret key base is used to derive the periodic key of the  $i$ -th period,  $(PK_i, SK_i)$ , respectively.

**Definition 1** A key-evolving protocol consists of three algorithms  $(KG, g, f)$ :



1. Public/Secret Key Base Generation Algorithm  $KG(k)$   
Given a security parameter  $k$ , the secret key holder generates the public/secret key base  $(PKB, SKB)$ .  $SKB$  is kept secret at the secret key holder and  $PKB$  is then distributed to the users in the form of a public certificate.
2. Public Key-Evolving Algorithm  $g()$   
Given the period  $i$  and  $PKB$ , the public key holder computes  $PK_i = g(PKB, i)$ .
3. Secret Key-Evolving Algorithm  $f()$   
Given the period  $i$  and  $SKB$ , the secret key holder computes  $SK_i = f(SK_{i-1})$  or  $f(SKB, i)$ .

Next, the desirable properties of the key-evolving protocols are as follows:

**Definition 2** A key-evolving protocol is *forward-secret* if the compromise of  $SK_i$  does not compromise  $SK_j$  for all  $j < i$ .

**Definition 3** A key-evolving protocol is *backward-secret* if the compromise of  $SK_i$  does not compromise  $SK_j$  for all  $j > i$ .

**Definition 4** A key-evolving protocol is *key-independent* if it is *forward-secret* and *backward-secret*.

**Definition 5** A key-evolving protocol is  $t$ -bounded key-independent if a set of periodic keys  $C$  are compromised and the following conditions hold.

1. If  $C$  is large enough ( $|C| > t$ ), all the periodic secret keys will be compromised.
2. If  $C$  is not large enough ( $|C| \leq t$ ), the remaining periodic secret keys are still secret.

### 3. PROTOCOL 1

The first protocol is based on the difficulty of computing the discrete logarithm in  $Z_p^*$ . It employs the construction of a non-interactive verifiable secret sharing scheme proposed by Feldman [17], which is based on the secret sharing scheme of Shamir [18]. The following properties of polynomials are the basis of the Shamir's scheme.

- I. Given  $t + 1$  distinct points on the polynomial  $m(x)$  of degree  $t$ , namely  $(x_0, y_0), (x_1, y_1), \dots, (x_t, y_t)$  and  $y_i = m(x_i)$ , all the  $t + 1$  coefficients can be determined. In other words, the polynomial can be uniquely determined.
- II. Given  $t$  distinct points on the polynomial  $m(x)$  of degree  $t$ , namely  $(x_1, y_1), (x_2, y_2), \dots, (x_t, y_t)$  and  $y_i = m(x_i)$ , the  $t + 1$  coefficients cannot be uniquely determined.

The protocol, presented in Fig. 1, consists of three algorithms. First, the secret key holder executes the key base generation algorithm and publishes the public key base  $PKB = (P_0, P_1, \dots, P_t) = (g^{a_0}, g^{a_1}, \dots, g^{a_t})$ , where  $a_0, a_1, \dots, a_t$  are the coefficients of a polynomial  $f(x)$ . In contrast, the secret key base  $SKB = (a_0, a_1, \dots, a_t)$  is kept secret. Also, the secret key holder publishes a hash function  $h$ , which works as a randomizer of the index  $i$ . For example, this  $h(\cdot)$  can be a permutation in  $Z_q$ .



---

### 1. Public/Secret Key Base Generation Algorithm $KG(1^n, t)$

- (1) The secret key holder chooses an  $n$ -bit prime  $p = 2q + 1$  where  $q$  is a prime of at least 160-bits, i.e.  $q > 2^{160}$ . Let  $G_q$  denote the subgroup of the quadratic residues modulo  $p$  and  $g$  is the generator of  $G_q$ .
- (2) The secret key holder chooses a  $t$ -degree polynomial  $f(x)$  with coefficients randomly chosen from  $Z_q$ , and  $f(x) = \sum_{j=0}^t (a_j x^j) \pmod{q}$ .  $SKB = (a_0, a_1, \dots, a_t)$ , the coefficients of  $f(x)$ , will be kept secret and used to derive the secret keys for later periods.
- (3) The secret key holder publishes the public key base information and a hash function, including

$$PKB = (P_0, P_1, \dots, P_t) = (g^{a_0}, g^{a_1}, \dots, g^{a_t}), \text{ and } h: N \rightarrow Z_q.$$

### 2. Public Key-Evolving Algorithm

Given index  $i$ , the public key holder will update its public key of period  $i$  as

$$PK_i = \prod_{j=0}^t (P_j)^{h(i)^j} \pmod{p}.$$

### 3. Secret Key-Evolving Algorithm

Given index  $i$ , the secret key holder will update the key of period  $i$  as

$$SK_i = f(h(i)) \pmod{q}.$$


---

Fig. 1. Key-evolving protocol using the Feldman's technique.

The public key holder retrieves and verifies  $PKB$  and  $h(\cdot)$ . Suppose that he needs the public key of period  $i$ . He would then perform  $PK_i = \prod_{j=0}^t (P_j)^{h(i)^j} \pmod{p}$ .

$SKB$  is the long-term system secret, protected separately from  $SK_i$  at the secret-key holder. Its protection follows the paper of Shamir's secret-sharing scheme [18]. At the beginning of the period  $i$ , the secret key holder evaluates the secret key as  $SK_i = f(h(i)) \pmod{q}$ . The analysis of this protocol is as follows:

**Correctness** The relation of  $PK_i = g^{SK_i}$  is established because

$$\begin{aligned} PK_i &= \prod_{j=0}^t (P_j)^{h(i)^j} \pmod{p} = \prod_{j=0}^t (g^{a_j})^{h(i)^j} \pmod{p} \\ &= g^{\sum_{j=0}^t a_j \cdot h(i)^j \pmod{q}} = g^{f(h(i)) \pmod{q}} = g^{SK_i}. \end{aligned}$$

**Security** The following lemmas summarize the relationship between compromising a secret key for a given public key and solving a discrete logarithm problem. It is straightforward to solve the discrete logarithm problem, given a successful attacker.



**Lemma 1** The attacker, who is able to compute the corresponding  $SK$  of a given  $PK$ , is able to solve of the Discrete Logarithm problem in  $Z_p^*$ .

Given  $t + 1$  sets of keys of  $(SK_{i_0}, i_0), (SK_{i_1}, i_1), \dots, (SK_{i_t}, i_t)$ , the attacker can determine the coefficients of  $f(x)$  and thus the secret keys of all periods. In contrast, if less than  $t + 1$  secrets keys are compromised, then the polynomial for the secret key-evolving cannot be uniquely determined. Therefore, even if  $t$  secret keys are compromised, it seems difficult to derive even one additional secret key out of the  $t$  secret keys. It seems that the only way to compute one extra secret key is to compute the discrete logarithm in  $Z_p^*$  directly. Therefore, we have the following conjecture.

**Conjecture 1** The protocol in Fig. 1 is  $t$ -bounded key-independent.

**Efficiency** This protocol is efficient for the secret key holder, requiring only the evaluation of a  $t$ -degree polynomial over  $Z_q$ . The computational complexity is the same as that of Tzeng's work. In contrast, other related key-evolving protocols for the secret key require modular exponentiations [8-10].

For a public key holder, he needs to compute  $t$  modular exponentiations and  $t$  modular multiplications, which is the same as in Tzeng's scheme. However, the on-line computational load is reduced if pre-computation is employed.

#### 4. PROTOCOL 2

This protocol is based on the difficulty of computing the discrete logarithm (DL) in  $Z_n^*$ , where  $n$  is the product of several large primes. It uses the same technique of Maurer-Yacobi in the design of non-interactive public-key distribution systems [19]. All operations are performed in  $Z_n^*$ , where the factoring of  $n$  is hard and  $g$  is the generator of  $QR_n$ , the quadratic residues of  $Z_n^*$ .  $QR_n = (g)$  is employed because it is a large cyclic subgroup of  $Z_n^*$ , where the computing of the discrete logarithm is hard.

This protocol is presented in Fig. 2 and consists of three algorithms. First, the secret key holder executes the key base generation algorithm and publishes the public key base  $PKB = (n, g, H, PK_0)$ , where  $n$  and  $g$  are defined as above,  $H$  is a cryptographical hash function, and  $PK_0$  is a random number. The secret key base  $SKB$ , consisting of the factors of  $n$ , is kept secret.

The public key holder retrieves and verifies the public key base  $PKB$ . Suppose that he needs the public key of period  $i$ . He would perform a series of hash-and-square operations to get  $PK_1 = (H(PK_0))^2$ ,  $PK_2 = (H(PK_1))^2$ , ...,  $PK_i = (H(PK_{i-1}))^2$ . All operations are performed in the group of  $Z_n$ . If he could store some of these intermediate values, he can reduce the hash-and-square computations.

$SKB$  is the long-term system secret, protected separately from  $SK_i$  at the secret holder. Its protection follows the paper of Maurer and Yacobi [19]. At the beginning of period  $i$ , the secret key holder performs the hash-and-square  $PK_i = (H(PK_{i-1}))^2 \pmod{n}$  and then calculates the corresponding secret key  $SK_i = \log_g(PK_i)$ . This task can employ the Pohlig-Hellman algorithm as follows. First, the secret key holder computes the discrete logarithm of  $PK_i$  in  $QR_{p_i}$  and then uses the Chinese Remainder Theorem (CRT) to construct the discrete logarithm of  $PK_i$  in  $QR_n (\equiv QR_{p_1} \times QR_{p_2} \dots \times QR_{p_r})$ .



---

### 1. Public/Secret Key Base Generation Algorithm

The secret key holder chooses a  $k$ -bit composite  $n = p_1 p_2 \dots p_r$  where the factoring of  $n$  is intractable and  $q_1 = (p_1 - 1)/2, q_2 = (p_2 - 1)/2, \dots, q_r = (p_r - 1)/2$  are pairwise relatively prime.

Then he broadcasts the public key base  $PKB = (n, g, H, PK_0)$ , where the following requirements are met.

- (a)  $SKB = (p_1, p_2, \dots, p_r)$ , the set of factors of  $n$ , is kept secret to the secret key holder,
- (b)  $g$  is the generator of the quadratic residues  $QR_n$ .
- (c)  $H(\cdot): \{0, 1\}^* \rightarrow Z_n^*$  is a cryptographical hash function,
- (d)  $PK_0$  is a random number.

### 2. Public Key-Evolving Algorithm

Given period  $i$ , the public key holder evaluates for  $j = 1$  to  $i$ ,

$$PK_j = (H(PK_{j-1}))^2 \pmod{n} \in QR_n.$$

### 3. Secret Key-Evolving Algorithm

At the beginning of period  $i$ , the secret key holder first computes the public key by one hash-and-square

$$PK_i = (H(PK_{i-1}))^2 \pmod{n} \in QR_n.$$

With the factoring knowledge of  $n$ , the secret key holder computes

$$SK_i = \log_g (PK_i).$$


---

Fig. 2. Key-evolving based on Maurer-Yacobi scheme.

**Correctness**  $PK_i = g^{SK_i}$  is established because  $SK_i$  is computed by the secret key holder with the trapdoor knowledge of factoring.

**Security** The security analysis consists of two computational reductions. The first reduction (proposition 1) shows that in the random oracle model a successful attacker can be used to construct a DL oracle in  $Z_n^*$ . The second reduction (proposition 2) shows that a DL-oracle can be used to construct a factoring oracle. The two reductions are summarized in Theorem 1.

**Proposition 1** The successful attacker can be turned into a discrete logarithm (DL) oracle in the random oracle model.

**Proof:** The goal of attacker  $A$  is to break the key-independence property. In other words, when the periodic secret  $SK_i$  is compromised, the successful attacker will compromise one additional signing key  $SK_j$  and  $j \neq i$ . By the following simulation, we will use the successful attacker to compute  $x = \log_g y$ .



First, we generate a set of secret/public key pairs

$$\{(SK_i, PK_i) \mid PK_i = g^{SK_i}, SK_i \text{ is even}, i = 1, 2, \dots, T, i \neq k\}.$$

Also, we control the output of the hash query as follows. If attacker  $A$  makes a hash query of period  $i$ ,  $i \neq k$ , the hash returns  $H(PK_i) = g^{\frac{SK_i}{2}}$ . If attacker  $A$  makes a hash query of period  $k$ , the hash returns  $H(PK_k) = yg^a$ .

Next, we simulate the case of key-compromise by revealing some  $(PK_i, SK_i)$  to the attacker. If  $A$  succeeds in breaking the forward-secret and forward-secret properties,  $A$  would eventually return  $(PK_j, SK_j)$  for some  $j \neq i$ .

The chance that  $j = k$  is  $\frac{1}{T}$ . If this happens, we have  $SK_j = SK_k = \log_g PK_k = \log_g y^2 g^{2a} = 2\log_g y + 2a = 2x + 2a$ . Note that  $SK_j$  is even. Though the order of  $QR_n$  is unknown, the discrete logarithm of  $y$  can be computed as follows:

$$x = \frac{1}{2}SK_j - a. \quad \square$$

**Proposition 2** A discrete logarithm (DL) oracle in  $Z_n^*$  can be turned into a factoring oracle for  $n$ .

**Proof:** Fig. 3 presents this algorithm. It basically follows the work of Maurer and Yacobi [19] but uses a different cyclic group in  $Z_n^*$ . In their paper,  $g$  is the generator of the maximal cyclic group of order  $\lambda(n) = 2q_1 q_2 \dots q_r$ . In this paper,  $g$  is the generator of  $QR_n$  of order  $\lambda(n)/2$ . We denote  $\lambda(n)/2$  as  $Q_n$ .  $DL$  is the discrete logarithm oracle that returns  $\log_g y < Q_n$  when the input is  $y$ . Given  $n$ ,  $g$ , and access to  $DL$ , this algorithm returns the factors of  $n$ .

---

Algorithm DL-Factoring

Input:  $DL$  oracle,  $n$ ,  $g$ : the generator of  $QR_n$ ,

Output: the factors of  $n$

1. Repeat (a)-(c)  $m$  times

(a) Find  $t_i$  such that  $Q_n \leq t_i \leq n$ .

Since  $Q_n$  is unknown,  $t_i$  is chosen not far from  $n$ .

(b) Send  $g^{t_i}$  to  $DL$ .

(c)  $DL$  computes the discrete logarithm of  $g^{t_i}$  and outputs  $t'_i < Q_n$ .

2. Compute  $Q_n = GCD(t_1 - t'_1, t_2 - t'_2, \dots, t_m - t'_m)$ .

3. Given  $Q_n$ , compute the factors of  $n$  using the Miller algorithm and return the results.

---

Fig. 3. Algorithm DL-factoring: build a factoring oracle using a DL oracle.

The following iteration is repeated  $m$  times (i.e.  $i = 1$  to  $m$ ). We choose random  $t_i$ , not far from  $n$ , compute  $g^{t_i}$  and submit it to  $DL$ . As the input  $g^{t_i}$  arrives,  $DL$  computes its discrete logarithm and outputs  $\log_g g^{t_i} = t'_i < Q_n$ .



In each iteration,  $t_i$  and  $t'_i$  satisfy  $g^{t_i} = g^{t'_i}$ . In other words,  $g^{t_i - t'_i} = 1$  and  $Q_n$ , the order of  $g$ , divides  $t_i - t'_i$ . As  $t_i - t'_i$  is a small multiple of  $Q_n$ ,  $Q_n$  can be computed as the greatest common divisor of all  $t_i - t'_i$ . Once we know  $Q_n$ , the factors of  $n$  can be computed using Miller's algorithm [20].  $\square$

The above two propositions are summarized in the following theorem.

**Theorem 1** In the random oracle model, the key-evolving protocol in Fig. 2 is key-independent if the factoring of  $n$  is hard.

**Efficiency** This protocol is efficient for the public key holders, requiring only one hash-and-square operation per period. In comparison, the related cryptosystems need modular exponentiations either in key-evolving protocols [6, 10] or in the encryption scheme [9]. Thus, this is suitable for public key holders with low computation power such as mobile devices or smart cards.

However, the computation load for the secret key holder increases. It requires the secret key holder to compute the discrete logarithm in several smaller cyclic groups, which requires moderate computation power [19].

## 5. CONCLUSIONS

In this paper, we first presented the security notions of key-evolving protocol, including forward-secrecy, backward-secrecy and key-independence. Next, two concrete constructions were proposed. One protocol is based on a scheme of Feldman and is efficient for the secret-key holder; the other is a variant of the Maurer-Yacobi protocol, and is efficient for the public-key holder. Also, we provide proofs and analysis for correctness, security and efficiency.

## REFERENCES

1. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, Boca Raton, 1997.
2. M. Abdalla and M. Bellare, "Increasing the lifetime of a key: a comparative analysis of the security of re-keying techniques," in *Proceedings of Advances in Cryptology (ASIACRYPT 2000)*, Vol. 1976, 2000, pp. 546-559.
3. M. Bellare and S. K. Miner, "A forward-secure digital signature scheme," in *Proceedings of Advances in Cryptology Conference (CRYPTO '99)*, Vol. 1666, 1999, pp. 431-448.
4. M. Abdalla and L. Reyzin, "A new forward-secure digital signature scheme," in *Proceedings of Advances in Cryptology (ASIACRYPT 2000)*, Vol. 1976, 2000, pp. 116-129.
5. H. Krawczyk, "Simple forward-secure signatures from any signature scheme," in *Proceedings of the 7th ACM Conference on Computer and Communications Security (CCS '00)*, 2000, pp. 108-115.
6. W. Tzeng and Z. Tzeng, "Robust key-evolving public key encryption schemes," Record 2001/009, Cryptology ePrint Archive, 2001.



7. C. F. Lu and S. P. Shieh, "Secure key-evolving protocols for discrete logarithm schemes," in *Topics in Cryptology, CT-RSA 2002, LNCS2271*, 2002, pp. 300-309.
8. G. Itkis and L. Reyzin, "Forward-secure signatures with optimal signing and verifying with gene itkis," in *Proceedings of Advances in Cryptology (CRYPTO '01)*, Vol. 2139, 2001, pp. 332-354.
9. Y. Dodis, J. Katz, S. Xu, and M. Yung, "Key-insulated public key cryptosystems," in *Advances in Cryptology (Eurocrypt 2002)*, Vol. 2332, 2002, pp. 65-82
10. G. Itkis and L. Reyzin, "Intrusion-resilient signatures, or towards obsolescence of certificate revocation," in *Crypto 2002*, available from IACR ePrint, 2002.
11. A. Perrig, "Efficient collaborative key management protocols for secure autonomous group communication," in *International Workshop on Cryptographic Techniques and E-Commerce (CrypTEC '99)*, 1999, pp. 192-202.
12. A. Perrig, Y. Kim, and G. Tsudik, "Simple and fault-tolerant key agreement for dynamic collaborative groups," in *Proceedings of the 7th ACM Conference on Computer and Communications Security (CCS '00)*, 2000, pp. 235-244.
13. C. G. Guenther, "An identity-based key-exchange protocol," in *Proceedings of Advances in Cryptology (Eurocrypt '89)*, Vol. 434, 1989, pp. 29-37.
14. D. E. Denning and M. S. Sacco, "Timestamps in key distribution protocols," *Communications of the ACM*, Vol. 24, 1981, pp. 533-536.
15. Y. Yacobi and Z. Shmueli, "On key distribution systems," in *Advances in Cryptology (CRYPTO '89)*, 1989, pp. 268-273.
16. Y. Yacobi, "A key distribution 'paradox'," in *Advances in Cryptology (CRYPTO '90)*, 1990, pp. 268-273.
17. P. Feldman, "A practical scheme for non-interactive verifiable secret sharing," in *28th Symposium on Foundations of Computer Science (FOCS)*, 1987, pp. 427-437, IEEE Computer Society Press.
18. A. Shamir, "How to share a secret," *Communication of ACM*, Vol. \_\_\_\_, 1979, pp. 612-613.
19. U. M. Maurer and Y. Yacobi, "A non-interactive public-key distribution system," *Design, Codes and Cryptography*, Vol. 9, 1996, pp. 305-316.

**Cheng-Fen Lu (呂正榮)** received her M.S. degree in Electrical and Computer Engineering Department from the University of Texas at Austin and Ph.D. degree in Department of Computer Science and Information Engineering from National Chiao Tung University in 2003. There she was with the lab of distributed system and network security, directed by Dr. Shiuh-Pyng Shieh. Also she was a NSC/DAAD exchange student at two German institutes for two years (1998-2000), including the Computer Science Department at University of Saarland and the Mathematics Department at University of Frankfurt. Currently, she is an assistant professor at the Department of Computer Science and Information Engineering, Ta Hwa Institute of Technology. Her research interests include cryptography, coding theory, and network security.



**Shiuh-Pyng Shieh (謝續平)** received the M.S. and Ph.D. degrees in Electrical and Computer Engineering from the University of Maryland, College Park, in 1986 and 1991, respectively. He is currently a professor and the chairman of the Department of Computer Science and Information Engineering, National Chiao Tung University; the vice chairman of Chinese Cryptology & Information Security Association; director of Cisco Internetworking Technology Lab. From 1988 to 1991 he participated in the design and implementation of the B2 Secure XENIX for IBM, Federal Sector Division, Maryland, U.S.A. He is also the designer of Secure Network Protocols (SNP), a popular security shareware on the Internet. He has been consultants in the areas of network security and distributed operating systems for many institutes, such as Industrial Technology Research Institute, and National Security Bureau, Taiwan. He was on the organizing committees of numerous conferences, and is currently an editor of Journal of Computer Security, and Journal of Information Science and Engineering. Recently, he has received two outstanding research awards, honored by National Chiao Tung University and Executive Yuan of Taiwan, respectively. His research interests include internetworking, distributed systems, and network security.

