Short Paper

Authentication Protocols Using Hoover-Kausik's Software Token[*]

WEI-CHI KU AND HUI-LUNG LEE [+]
Department of Computer Science and Information Engineering
Fu Jen Catholic University
Taipei, 242 Taiwan
E-mail: wcku@csie.fju.edu.tw
[+]*Department of Computer and Information Science*
National Chiao Tung University
Hsinchu, 300 Taiwan

In 1999, Hoover and Kausik introduced a software token using the cryptographic camouflage technique and claimed that it can resist various on-line and off-line guessing attacks. Later, Kwon presented an authentication protocol based on the cryptographic camouflage technique and DSA, and pointed out that this initial protocol is vulnerable to an impersonation attack once a server's secret key or private key is compromised. Then, Kwon proposed a modified version that can resist such an impersonation attack by cryptographically embedding the recipient's identity in the user's signature to ensure that only the intended recipient will accept this signature. However, we find that Kwon's modified protocol still has some drawbacks. In this paper, we first demonstrate the drawbacks of Kwon's modified protocol and then propose an improved authentication protocol based on the cryptographic camouflage technique and RSA. Finally, we show that our improved protocol can provide prefect forward secrecy and can resist the off-line guessing attack, the impersonation attack, the replay attack, and the Denning-Sacco attack. Furthermore, the resistance of our improved protocol to the modification attack is also enhanced by additionally using credit-card sized CD-ROMs.

Keywords: password, authentication, software token, cryptographic camouflage, guessing attack

1. INTRODUCTION

As the use of public-key cryptography in authentication and signatures increases the need for a secure, convenient, and economic mechanism for protecting a user's private key is becoming more pressing than ever before. Employing a tamper-resistant hardware token to store the private key can effectively keep outsiders from extracting its content and resist the guessing attack by locking up after a limited number of attempts to activate

Received January 2, 2004; revised July 9, 2004; accepted August 26, 2004.
Communicated by Chi-Sung Laih.

[*] This work was supported in part by the National Science Council of Taiwan, R.O.C., under grant No. NSC 92-2213-E-030-013.

it with incorrect passwords. Since the private key is protected by two authentication factors, the password and the hardware token, compromising a single authentication factor will not immediately breach the private key. However, hardware tokens require an expensive infrastructure in the form of dedicated readers/writers, and their deployment in large-scale communication is unfavorable. Therefore, researchers are motivated to develop a means of securely storing private keys in software tokens especially suitable for certain constrained environments, e.g., [6, 7, 10, 12]. The *software key container*, which conforms to PKCS #5 [14] and PKCS #8 [13], is a widely used method for storing private keys in software tokens. In the software key container, the private key is encrypted with the user's password. Because the software key container is not tamper-resistant, data stored in it tend to be compromised in practical environments [9]. However, the software key container has been found to be vulnerable to an off-line guessing attack [11] once it is compromised [7-9].

Recently, Hoover and Kausik [7] introduced a software token based on the cryptographic camouflage technique, in which the private key is protected by the user's password in a particular way. The idea behind the cryptographic camouflage technique is to guarantee that the adversary cannot distinguish the user's private key among spurious but plausible private keys. Although the adversary who has got a copy of the software token can try to crack it to obtain the private key, he will only recover many plausible private keys and cannot distinguish the correct private key from the spurious decoys. To camouflage private keys, Hoover and Kausik dictated: (1) do not encrypt a known structure with the password; (2) conceal the public key and do not use it to encrypt verifiable plaintext; (3) do not reveal information about the password; (4) randomize and protect signatures. In practice, the cryptographic camouflage technique has been employed in some commercial products, e.g., ArcotID™ [1]. In 2002, Kwon [9] described an authentication protocol based on cryptographically camouflaged DSA (Digital Signature Algorithm) [3] keys. Kwon then pointed out that this initial protocol is vulnerable to an impersonation attack once a server's secret key (or private key) is compromised. In addition, he proposed a modified version, which will be abbreviated as Kwon's modified protocol hereafter, and claimed that his modified version is strong against the off-line guessing attack even if the software token is compromised. However, we find that Kwon's modified protocol still has two drawbacks. In this paper, we will first review Kwon's modified protocol and then show its drawbacks. Next, we will propose an improved authentication protocol, which is based on camouflaged RSA [15] keys. We will show that the improved protocol can provide prefect forward secrecy and can resist the off-line guessing attack, the impersonation attack, the replay attack, and the Denning-Sacco attack [2]. Additionally, the resistance of the improved protocol to the modification attack will be discussed. Hereafter, we will use 'camouflage' for short instead of 'cryptographic camouflage.'

2. DRAWBACKS OF KWON'S MODIFIED PROTOCOL

In 1999, Kwon [9] described an authentication protocol based on cryptographically camouflaged DSA [3] keys. However, Kwon [9] showed that his initial protocol is flawed in that a malicious server can impersonate a user to login to another server by opening interleaved sessions without having to compromise any authentication factor of the user. Then, Kwon [9] proposed a modified version that can resist such an impersonation attack. Unfortunately, we find that Kwon's modified protocol still has some drawbacks. In this section, we will first review Kwon's modified protocol and then show its drawbacks.

2.1 Kwon's Modified Protocol

Each user selects his own DSA system with the parameters (p, q, g, x, y), where p and q are large prime numbers, $g = h^{(p-1)/q} \bmod p$ $(h \in Z)$, x is his private key, and y $(= g^x \bmod p)$ is his public key. To protect x, the user chooses a password π and computes $a = E(k(\pi), x)$, where $k(\ )$ denotes a key derivation function and $E(\ )$ denotes the encryption function of a secret-key cryptosystem. To conceal his public key, the user computes $b = \mathcal{E}(\varphi, (y, g, p, q))$, where φ denotes the server's encryption key and $\mathcal{E}(\ )$ denotes the encryption function of either a secret-key or a public-key cryptosystem. In addition, b is contained in the certificate $Cert_b$ that is signed by a certification authority (CA) to ensure its authenticity [7]. The notation σ represents the server's decryption key. $D(\ )$ represents the decryption function corresponding to $E(\ )$. And $\mathcal{D}(\ )$ represents the decryption function corresponding to $\mathcal{E}(\ )$.

According to the mechanism used to encrypt the user's public key, the authentication protocol can be operated in one of the following models: the *S* model, where the user's public key is encrypted with the secret key shared by all servers; the *P* model, where the user's public key is encrypted with the public key of each server; the *CP* model, where the user's public key is encrypted with the common public key of all the servers. In the *S* model, the encryption key equals the decryption key, i.e., $\varphi = \sigma$. In the *P* and *CP* model, φ denotes the public key, and σ denotes its corresponding private key. In the *CP* model, all the servers share the same key pair. Initially, the user memorizes π and keeps (a, b, g, p, q, φ) in his software token. Kwon's modified protocol can be described as in the following steps:

Step 1: The user sends a login request to the server.

Step 2: The server replies with a random challenge m of L bits to the user. (L represents the security parameter of the system.)

Step 3: The user inputs π to produce $k(\pi)$ and then computes $x = D(k(\pi), a)$. In addition, the user computes $n = h(svr, h(m))$, where $h(\ )$ represents a one-way hash function and svr denotes the server's identity. Next, the user chooses a random integer c within $[1, q-1]$ and computes $r = (g^c \bmod p) \bmod q$ and $s = c^{-1}(n + xr) \bmod q$. Then, the user computes $w = \mathcal{E}(\varphi, (r, s))$ and then sends w to the server along with $Cert_b$.

Step 4: If $Cert_b$ is valid, the server uses σ to decrypt w and b, i.e., $(r, s) = \mathcal{D}(\sigma, w)$ and $(y, g, p, q) = \mathcal{D}(\sigma, b)$. Next, the server computes $n = h(svr, h(m))$, $u_1 = s^{-1}n \bmod q$, and $u_2 = rs^{-1} \bmod q$. Then, the server computes $v = (g^{u_1}y^{u_2} \bmod p) \bmod q$. If $v = r$, the server accepts the user's login request.

2.2 Drawbacks

Since ordinary storage is not tamper-resistant and software tokens can easily be copied and backed up for users' convenience, an accidental compromise of a software token cannot be entirely prevented. Suppose that the user's (a, b, g, p, q, φ) stored in the software token is compromised by an adversary. First, the adversary tries a candidate password π' to compute $k(\pi')$ and then uses $k(\pi')$ to decrypt a, which yields x' $(= D(k(\pi'), a))$. Then, the adversary computes $y' = g^{x'} \bmod p$. Then, the attack for each model can be described as follows.

- For the S model: Since the adversary has obtained the servers' shared secret key φ $(= \sigma)$ from the user's software token, he can compute $(y, g, p, q) = D(\varphi, b)$. If $y' = y$, the adversary has guessed the correct password π' $(= \pi)$, which also implies that he has obtained the user's private key x' $(= x)$.
- For P and CP model: As the adversary knows the server's public key φ, he can compute $b' = \mathcal{E}(\varphi, (y', g, p, q))$. If $b' = b$, then the adversary has guessed the correct password π' $(= \pi)$, which also implies x' $(= x)$.

Therefore, Kwon's modified protocol fails to resist the off-line guessing attack as described above. Clearly, Kwon's modified protocol operated in the P and CP models can be easily improved by implementing $\mathcal{E}(\ )$ with an encoding method that embeds sufficient randomness, e.g., the EME-PKCS1-v1.5 specified in PKCS #1 [15]. To avoid implementation dependency, another improvement is to embed sufficient randomness when computing b such that $b = \mathcal{E}(\varphi, (y, g, p, q))$ is modified to obtain $b = \mathcal{E}(\varphi, (y, g, p, q, R))$, where R is a large random integer. Because the adversary does not know R, he cannot compute $b' = \mathcal{E}(\varphi, (y', g, p, q))$ and compare it with b. However, in the S model, the adversary can directly use the decryption key φ, which is stored in the software token in plaintext, to mount an off-line guessing attack. At this point, we cannot recommend a simple improvement for Kwon's modified protocol operated in the S model.

In some situations, the content of the software token can be easily modified, e.g., when the software token is contained within the rewritable storage of a public computer. We will show that the adversary can obtain the user's password by performing such a modification attack. Suppose that the adversary has compromised the software token and obtained (a, b, g, p, q, φ). The adversary can modify φ to get φ', which is the adversary's secret key for the S model or public key for the P/CP model. Once the user uses the modified $(a, b, g, p, q, \varphi')$, the adversary can obtain the user's password as follows. When the user sends a login request to the server, the adversary interrupts the request and then sends a random m to the user. Next, the user uses $k(\pi)$ to decrypt a, which yields x. Then, he chooses c at random within $[1, q - 1]$ and computes $r = (g^c \bmod p) \bmod q$, $n = h(svr, h(m))$, and $s = c^{-1}(n + xr) \bmod q$. Next, the user encrypts (r, s) with the adversary's secret key (or public key) φ' and sends out w along with $Cert_b$. Upon receiving w, the adversary uses his secret key (or private key) σ' to decrypt it, which yields (r, s), and computes $n = h(svr, h(m))$, $u_1 = s^{-1}n \bmod q$, and $u_2 = rs^{-1} \bmod q$. Next, the adversary can guess a password π' to decrypt a, which yields the corresponding x', and compute $y' = g^{x'}$ mod p and $v' = (g^{u_1}y'^{u_2} \bmod p) \bmod q$. If $v' = r$, then the adversary has guessed the correct password π' $(= \pi)$, which also implies that he has obtained the user's private key x' $(= x)$.

3. THE IMPROVED PROTOCOL

RSA [15] is one of the most widely used public-key cryptosystems and has been regarded as a de-facto standard that is extremely important for the development of a digital economy. In addition, adoption of RSA has grown to the extent that standards are being written to accommodate RSA. For example, the U.S. government changed FIPS PUB 186-1 [4], which is the corrected version of FIPS PUB 186 [3], to FIPS PUB 186-2 [5] with the emphasis on RSA digital signatures to support the de-facto standard of official and financial institutions. Because RSA is more popular than DSA in practice and can be used for both encryption and digital signatures, we will describe an improved authentication protocol based on RSA and Hoover-Kausik's cryptographic camouflage technique instead of improving Kwon's modified protocol, which is based on camouflaged DSA keys. In contrast to Kwon's modified protocol, the improved protocol additionally provides mutual authentication and session key establishment, and can resist the off-line guessing attack, the impersonation attack, the replay attack, and the Denning-Sacco attack. Furthermore, the improved protocol provides prefect forward secrecy. In addition, the resistance of the improved protocol to the modification attack is enhanced by also using credit-card sized CD-ROMs.

As previously explained, the authentication protocol operated in the *S* model is vulnerable to an off-line guessing attack; therefore, we will only describe the authentication protocol that can be operated in the *P* and *CP* models. Initially, each user selects his own RSA system with parameters (p, q, n, e, d), where p and q are large prime numbers, $n = p \times q$, e is the user's public exponent, and d is the user's private exponent. In addition, the server selects the common Diffie-Hellman algorithm parameters (r, g), where r is a large prime number and g $(< r)$ is a primitive root of r, for all users. Other notations used in the improved protocol are listed in Table 1.

Table 1. Notations of the improved protocol.

Notation	Description
svr	server's identity
π	user's password
$h_1(\ )$, $h_2(\ )$	one-way hash function ($h_1(\ )$ are $h_2(\ )$ are uncorrelated)
sk	session key
$k(\ )$	key derivation function
φ	server's public key
σ	server's private key
$Cert_b$	certificate of b issued by certificate authority (CA)
$E(\ )$	the encryption function of a secret-key cryptosystem
$D(\ )$	the decryption function corresponding to $E(\ )$
$\mathcal{E}(\ )$	the encryption function of RSA
$\mathcal{D}(\ )$	the decryption function corresponding to $\mathcal{E}(\ )$
a	$a = E(k(\pi), (d, n))$
b	$b = \mathcal{E}(\varphi, (e, n))$

To protect (d, n), the user chooses a password π and computes $a = E(k(\pi), (d, n))$. To conceal his public key (e, n), the user encrypts it with φ to derive $b = \mathcal{E}(\varphi, (e, n))$. The user memorizes the password π and keeps (a, b, g, r, φ) in a software token. Additionally, the software token is stored in a credit-card sized CD-ROM, which can fit into the center recess of most CD-ROM and DVD-ROM drives nowadays. Since the credit-card sized CD-ROM does not require any special equipment, expensive infrastructure in the form of dedicated readers/writers is avoided, i.e., the deployment cost remains low. To resist the modification attack, the user should carry his CD-ROM with him and ensure that it will not be stealthily replaced by others. Note that Arcot System Inc. [1] also gives a similar suggestion for using its ArcotID™.

Step 1: The user chooses a random number x and computes $R_1 = g^x \bmod r$. Next, the user computes $T_1 = \mathcal{E}(\varphi, R_1)$ and sends T_1 to the server.

Step 2: Upon receiving T_1 from the user, the server chooses a random number y and computes $R_2 = g^y \bmod r$. Subsequently, the server decrypts T_1 with the server's private key σ, which yields R_1, and then computes the session key $sk = h_2((R_1)^y \bmod r)$ $(= h_2(g^{xy} \bmod r))$. Next, the server computes $T_2 = E(sk, R_1)$, and then sends R_2 and T_2 to the user.

Step 3: Upon receiving R_2 and T_2, the user computes the session key $sk = h_2((R_2)^x \bmod r)$ $(= h_2(g^{xy} \bmod r))$. If $R_1 = D(sk, T_2)$, the user authenticates the server. Otherwise, the user terminates this session. Next, the user inputs his password π to compute $k(\pi)$ and uses $k(\pi)$ to decrypt a, which yields his private key (d, n). Then, the user computes $m = h_1(svr, h_1((R_2)^x \bmod r))$ and $c = m^d \bmod n$. Subsequently, the user encrypts c with sk and sends the result w to the server along with $Cert_b$.

Step 4: Upon receiving w and $Cert_b$, the server first verifies $Cert_b$. If $Cert_b$ is valid, the server computes $(e, n) = \mathcal{D}(\sigma, b)$ and $c = D(sk, w)$. Then, the server computes $m = h_1(svr, h_1((R_1)^y \bmod r))$ and $v = c^e \bmod n$. If $m = v$, the server accepts the user's login request and enables sk for communicating with the user securely. Otherwise, the user's login request is rejected.

4. SECURITY ANALYSIS OF THE IMPROVED PROTOCOL

Suppose that the adversary has obtained the user's software token. We will now analyze the security strength of the improved protocol.

Resistance to the Off-Line Guessing Attack

The adversary can use the guessed password π' to decrypt a, which yields (d', n'), but he cannot produce the corresponding public key (e', n'). Hence, the adversary cannot verify whether he has obtained the correct (e, n) by using $b = \mathcal{E}(\varphi, (e, n))$, i.e., he cannot verify whether $\pi' = \pi$ and $(d', n') = (d, n)$. Therefore, the improved protocol can resist the off-line guessing attack without relying on its implementation.

Resistance to the Impersonation Attack

Upon receiving w, the server can compute $D(sk, w)$ to derive c. Next, the server can compute $c^e \bmod n$ to derive $m = h_1(svr, h_1((R_1)^y \bmod r))$. However, the server, with identity *svr*, cannot use *m* to impersonate the user to login to another server, with identity *svr′*, by opening interleaved sessions. That is, by cryptographically embedding the recipient's identity in the user's signature, we can ensure that only the intended recipient will accept this signature. Thus, the improved protocol can resist the impersonation attack.

Resistance to the Replay Attack

Suppose that the adversary has obtained a previously used x' and captured the corresponding protocol messages. Then, the adversary can try to mount a replay attack as follows. The adversary can replay x' and $T_1' = \varepsilon(\varphi, R_1')$, where $R_1' = g^{x'} \bmod r$, to the server. Then, the server will decrypt T_1' with σ to obtain R_1'. In addition, the server will choose a random number y and compute $R_2 = g^y \bmod r$ and the session key $sk = h_2((R_1')^y \bmod r)$ $(= h_2(g^{x'y} \bmod r))$. Next, the server will compute $T_2 = E(sk, R_1')$ and then send R_2 and T_2 back to the adversary. The adversary can compute $sk = h_2((R_2)^{x'} \bmod r)$ $(= h_2(g^{x'y} \bmod r))$ and $m = h_1(svr, h_1((R_2)^{x'} \bmod r))$. However, the adversary cannot compute the correct $c = m^d \bmod n$ that can be accepted by the server because he does not know the user's private key (d, n). Alternatively, if the adversary encrypts the previously captured c' with sk and sends the result w to the server, his login request will be rejected because the decrypted c' $(= (m')^d \bmod n)$ does not equal the expected c $(= m^d \bmod n)$. Since the adversary cannot generate the correct response that will be accepted by the server, the improved protocol can resist the replay attack.

Prefect Forward Secrecy

Suppose that the adversary has obtained the user's password π. Since $sk = h_2((R_1)^y \bmod r) = h_2((R_2)^x \bmod r)$ $(= h_2(g^{xy}))$, the adversary can compute a previously used session key sk and then derive the messages encrypted with it only if he knows either x or $\{R_1, y\}$. Knowing π, the adversary can compute the user's private key (d, n). Although R_2 is public, the adversary cannot compute y unless the Diffie-Hellman problem is solved, which is computationally infeasible with current techniques. In addition, since R_1 $(= g^x \bmod r)$ is encrypted by the server's public key φ, the adversary cannot compute R_1, which also implies that he has no chance to compute x no matter whether the Diffie-Hellman problem is solved or not. Hence, the improved protocol provides perfect forward secrecy.

Resistance to the Denning-Sacco Attack

If an old session key sk' is compromised by the adversary, a Denning-Sacco attack [2] can be attempted on obtaining π. The adversary can obtain the corresponding signature c' by using sk' and then guess password π^* to decrypt a, which yields (d^*, n^*). However, the adversary cannot compute m' from c' $(= m'^d \bmod n)$ because he does not know the user's public key (e, n). Since the adversary cannot compute c^* $(= m'^{d^*} \bmod n^*)$, it is infeasible for him to verify whether $\pi^* = \pi$ by comparing c^* with c'. Therefore, the improved protocol can resist the Denning-Sacco attack.

Resistance to the Modification Attack

The user's software token is stored in a credit-card sized CD-ROM, and the user only retrieves his software token from this CD-ROM. If the user can ensure that the CD-ROM containing his software token will not be stealthily replaced by others, he will not be fooled into using the bogus software token forged or modified by the adversary. In such a situation, the improved protocol can resist the modification attack. In practice, Arcot System Inc. [1] also gives a similar suggestion for using its ArcotID™. However, it may be questionable in real environments whether the user can effectively safeguard the CD-ROM containing his software token. Further research is needed to find a better solution.

5. CONCLUSIONS

Authentication protocols based on public-key cryptographic systems usually simply assume that the private keys are securely protected and can be easily retrieved for use. Some authentication protocols further assume that the private keys are stored in smart cards or some other dedicated tamper-resistant hardware tokens. However, storing private keys in tamper-resistant hardware tokens usually requires an expensive infrastructure in the form of dedicated readers/writers and their deployment in large-scale communication is unfavorable making them unsuitable for some constrained environments. Developing more convenient and cheaper methods for securely storing private keys has been a subject of recent researches, and among which the software token introduced by Hoover and Kausik has been paid with much attention. By using the cryptographic camouflage technique, we can ensure that the private key can be protected by the user's password in such a way that the adversary who has obtained the user's software token cannot distinguish the user's private key from spurious but plausible private keys. Recently, Kwon described an authentication protocol based on the cryptographic camouflage technique and DSA, and then pointed out that it is vulnerable to an impersonation attack. To improve the protocol's resistance to the impersonation attack, he also proposed a modified version. In this paper, we have demonstrated that Kwon's modified protocol still has some drawbacks. Additionally, we have described an improved protocol based on the cryptographic camouflage technique and RSA. In the improved protocol, the software token is stored in a credit-card sized CD-ROM. Since the credit-card sized CD-ROM does not require any special equipment, expensive infrastructure in the form of dedicated readers/writers is avoided, i.e., the deployment cost is low. In addition, we have shown that the improved protocol can resist the off-line guessing attack, the impersonation attack, the replay attack, and the Denning-Sacco attack. Furthermore, the improved protocol provides prefect forward secrecy. However, the resistance of the improved protocol to the modification attack depends on whether the user can prevent the CD-ROM containing his software token from being stealthily replaced by others. In the next stage of our research, we will try to find a superior solution.

REFERENCES

1. Arcot Systems Inc., available: http://www.arcot.com/.
2. D. Denning and G. Sacco, "Timestamps in key distribution protocols," *Communications of the ACM*, Vol. 24, 1981, pp. 533-536.
3. *DSS: Digital Signature Standard*, Federal Information Processing Standards Publication 186, FIPS 186, 1994.
4. *DSS: Digital Signature Standar*d, Federal Information Processing Standards Publi-

cation 186, FIPS 186-1, 1998.

5. *DSS: Digital Signature Standard*, Federal Information Processing Standards Publication 186, FIPS 186-2, 2000.
6. W. Ford and B. S. Kaliski, "Server-assisted generation of a strong secret from a password," in *Proceedings of 5th IEEE International Workshop on Enterprise Security*, 2000, pp. 176-180.
7. D. Hoover and B. Kausik, "Software smart cards via cryptographic camouflage," in *Proceedings of the IEEE Symposium on Security and Privacy*, 1999, pp. 208-215.
8. T. Kwon and J. Song, "Security and efficiency in authentication protocols resistant to password guessing attacks," in *Proceedings of 22nd IEEE Conference on Local Computer Networks*, 1997, pp. 245-252.
9. T. Kwon, "Impersonation attacks on software-only two-factor authentication schemes," *IEEE Communications Letters*, Vol. 6, 2002, pp. 358-360.
10. P. MacKenzie and M. Reiter, "Networked cryptographic devices resilient to capture," in *Proceedings of the IEEE Symposium on Security and Privacy*, 2001, pp. 12-25.
11. R. Morris and K. Thompson, "Password security: a case history," *Communications of the ACM*, Vol. 22, 1979, pp. 594-597.
12. R. Perlman and C. Kaufman, "Secure password-based protocol for downloading a private key," in *Proceedings of the Network and Distributed System Security Symposium*, 1999, pp. 1-9.
13. *PKISS: Private-Key Information Syntax Standard*, PKCS#8, Version 1.2, RSA Labs Tech. Note, 1993.
14. *Password-Based Encryption Standard*, PKCS #5, Version 2.0, RSA Labs Technical Note, 1999.
15. *RSA Cryptography Standard*, PKCS #1, Version 2.1, RSA Labs Technical Note, 2002.

Wei-Chi Ku (顧維祺) was born in Taiwan, R.O.C., in 1967. In 2000, he received the Ph.D. degree in Electrical Engineering from National Taiwan University. In 2001, Dr. Ku joined the faculty of the Department of Computer Science and Information Engineering at Fu Jen Catholic University, where he is currently an Associate Professor. His research interests include cryptography and information security.

Hui-Lung Lee (李惠龍) was born in Taoyuan, Taiwan, R.O.C., on June 16, 1976. He received the M.S. degree in Computer Science and Information Engineering from Fu Jen Catholic University in 2003. He is now a Ph.D. student of the Department of Computer and Information Science at National Chiao Tung University. His current research interests include image cryptography and information security.