Complete EAP Method: User Efficient and Forward Secure Authentication Protocol for IEEE 802.11 Wireless LANs

Chun-I Fan, *Member*, *IEEE*, Yi-Hui Lin, and Ruei-Hau Hsu

Abstract—It is necessary to authenticate users who attempt to access resources in Wireless Local Area Networks (WLANs). Extensible Authentication Protocol (EAP) is an authentication framework widely used in WLANs. Authentication mechanisms built on EAP are called EAP methods. The requirements for EAP methods in WLAN authentication have been defined in RFC 4017. To achieve user efficiency and robust security, lightweight computation and forward secrecy, excluded in RFC 4017, are desired in WLAN authentication. However, all EAP methods and authentication protocols designed for WLANs so far do not satisfy all of the above properties. This manuscript will present a complete EAP method that utilizes stored secrets and passwords to verify users so that it can 1) fully meet the requirements of RFC 4017, 2) provide for lightweight computation, and 3) allow for forward secrecy. In addition, we also demonstrate the security of our proposed EAP method with formal proofs.

Index Terms—Wireless local area networks (WLANs), extensible authentication protocol (EAP), forward secrecy, passwords, authentication, lightweight computation

1 Introduction

Authentication is the process of verifying users' identities when they want to access resources from networks. Typically, a user provides his authentication factors to a server, and then the server verifies them. If the factors are correct, the user is authorized to gain the access right to the resources provided by the server, and the server generates a session-key material that is shared with the user. Similarly, it is also crucial for Wireless Local Area Networks (WLANs) to authenticate users and build secure channels with them.

IEEE 802.11 [32], the most widely used standard, contains definitions for the operations of WLANs. The original design in the standard provides only some basic authentication mechanisms, such as preshared key establishment and password verification implemented between a user and a server, called Wired Equivalent Privacy (WEP). WEP is not secure because an attacker can obtain unauthorized access through intercepted messages [22]. The security of IEEE 802.11 was latter amended to include Wi-Fi Protected Access (WPA) and WPA2 [33].

IEEE 802.1x [34] defines the message encapsulation of Extensible Authentication Protocol (EAP). Fig. 1 shows the typical 802.1x message flows of successful authentication, where a supplicant represents a mobile node, an authenticator is usually an Access Point (AP), and the authentication server is usually the RADIUS server [14] that is responsible for authorization, authentication, and accounting. After the supplicant and the authenticator establish a data link, the communication between the supplicant and the authentication server starts.

Extensible Authentication Protocol (EAP), defined in RFC 3748 [1], is a flexible authentication framework that has been frequently utilized in WLANs. For IEEE 802.11, WPA and WPA2 have utilized EAP as their authentication mechanisms, such as EAP-TLS, EAP-TTLS, and EAP-SIM. A network administrator can appropriately choose a desired authentication mechanism, called an EAP method. The requirements for the EAP methods used in WLAN authentication have been defined in RFC 4017 [29].

There are several EAP methods designed for wireless networks. These EAP methods can be classified into three classes: 1) legacy EAP methods defined in RFC 3748 [1], [28], 2) EAP certificate-based methods that establish a tunnel and utilities of certificates [17], [4], [20], [11], [31], and 3) EAP methods that provide a way for two parties to convince each other that they both know a secret without revealing the secret to any third party who might be listening to the conversation [24], [5], [7], [10]. The methods of Class 1 are very efficient because no asymmetric-key computation is involved, but they suffer from some attacks. The security is much improved in the methods of Class 2. EAP-TLS uses certificates in both server and client sides to achieve mutual authentication. However, applying for a certificate is complicated because the client and the server need to expend extra efforts in certificate maintenance and revocation. In order to fix these weaknesses, the other certificate-based methods, such as EAP-TTLS, EAP-PEAP, and EAP-FAST, can choose to use the certificates or the passwords to authenticate the clients. The methods of Class 3 accept preshared secrets as authenticated credentials. They save the cost of verifying certificates while achieving the same

• *The authors are with the Department of Computer Science and Engineering, National Sun Yat-sen University, No. 70, Lienhai Rd., Kaohsiung 80424, Taiwan.*
E-mail: cifan@faculty.nsysu.edu.tw, {yihui1223, xyzhsu}@gmail.com.

Manuscript received 26 Dec. 2011; revised 29 Mar. 2012; accepted 8 May 2012; published online 22 May 2012.
Recommended for acceptance by D. Turgut.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2011-12-0935.
Digital Object Identifier no. 10.1109/TPDS.2012.164.

1045-9219/13/$31.00 © 2013 IEEE Published by the IEEE Computer Society

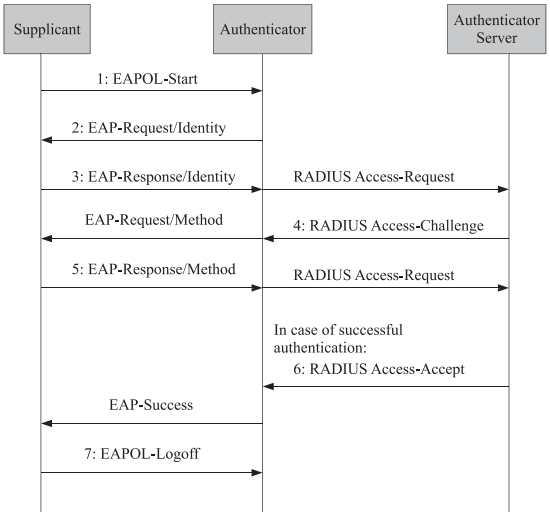

Fig. 1. The message exchange procedure of IEEE 802.1x.

security as the methods in Class 2. However, Class 3 methods still utilize the asymmetric algorithms, such as the Diffie-Hellman algorithm, in the key exchange procedure to provide perfect forward secrecy [19].

EAP-SSC [15] was designed especially for the smart card environment in 2004. The method builds an EAP secured channel between a smart card and an authentication server in both asymmetric and symmetric key-exchange models. The computation is efficient, but it does not fulfill some requirements of RFC 4017 and the security of forward secrecy.

Some two-factor authentication protocols designed for WLAN environments have been proposed in the literature. In 2003, Badra and Serhrouchni [8] proposed a key-exchange protocol to enhance end-to-end security in WAP. In 2004, Park and Park [26] proposed a two-factor authenticated key-exchange protocol for public wireless LANs. In the protocol, users accept passwords and tokens, such as smart cards, to perform authentication with a server. Park et al. claimed that it achieves mutual authentication, identity privacy, half forward secrecy, and low computation. Yoon and Yoo [30] proposed a protocol in 2006 where the number of rounds is fewer than Park et al.'s protocol, but Yoon et al.'s plan did not provide identity privacy.

In 2008, Juang and Wu [25] pointed out that Park et al.'s protocol did not achieve identity privacy and could not withstand dictionary attacks due to low entropy of identities. Therefore, Juang and Wu proposed two protocols with mutual authentication, identity privacy, and half forward secrecy. The difference between the two protocols is the way they achieve identity privacy. The first protocol protects the user's identity via a hash function, and the second one achieves this by using a new pseudoidentity for each session. However, we find that the first protocol also suffers from dictionary attacks on users' identities. Moreover, all of the above protocols [30], [25], [26] do not satisfy the requirements defined in RFC 4017 and they just achieve half forward secrecy instead of perfect forward secrecy.

In this paper, we propose an authentication protocol for wireless LAN environments that fully satisfies the EAP method requirements of RFC 4017 and perfect forward secrecy. Furthermore, since mobile devices are not suitable for high-cost computation, such as exponentiation computation, it is necessary to adopt a lightweight and secure authentication protocol in WLANs. Therefore, our work avoids exponentiation computations by applying symmetric-key algorithms, and it reduces the communication cost by employing a two-round trip, so the performance is enormously improved. We also prove that our protocol is secure.

Since 802.11 wireless LANs have been widely used in computation restricted mobile devices, such as smart phones and tablet computers, an energy-saving EAP method is desired. Some EAP methods can fulfill the security requirements in RFC4017 and achieves perfect forward secrecy, but they consume much more computation energy than our proposed EAP method. We avoid using asymmetric cryptosystems such that our method saves about 10,000 times computation cost in an Arm Cortex-A8 machine while it can still achieve the same security level.

The rest of the paper is organized as follows: in Section 2, we summarize the contribution of our research. In Section 3, we demonstrate our EAP method. In Section 4, we provide the security model and summarize the security properties of the proposed protocol, where the formal proofs for the security properties are shown in the supplementary file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2012.164. In Section 5, we compare widely used EAP methods and other authentication protocols designed for WLANs with our proposed EAP method. Finally, we make conclusions about this research in Section 6.

2 Contributions

Our proposed protocol will achieve the following goals.

1. **Filling security requirements in RFC 4017.** RFC 4017 defined the EAP requirements for EAP methods in WLANs. It has been considered as a standard for WLANs. Our proposed EAP method fills the following security requirements for wireless LANs:
   a. **Generation of symmetric keying material.** Our EAP method can generate a session key. The session key is produced during authentication used to protect exchanged data
   b. **Mutual authentication support.** Our EAP method provides mutual authentication. A client authenticates a server and vice versa. The client can verify if the server is rogue, while the server will know whether the client is legitimate.
   c. **Resistance to dictionary attacks.** The proposed EAP method provides a mechanism to withstand dictionary attacks when it accepts passwords as a credential. That is, an attacker cannot determine whether a guessed password is correct through the intercepted messages.
   d. **Protection against man-in-the-middle attacks.** In the EAP method, an attacker cannot intercept, modify, and send the modified messages to the receiver without being detected. This protection is relevant to cryptographic binding, integrity protection, replay protection, and session independence.

e. **Protected cipher suite negotiation.** The EAP method can negotiate a cipher suite to protect an EAP conversation in which the cipher suite negotiation is also protected.
f. **End-user identity hiding.** The identities of the users who participate in the exchange are encrypted during an EAP conversation.
g. **Fast reconnect.** The current security association is established in a smaller number of round trips when a security association has been previously established.

2. **Low computation and communication cost.** Due to hardware-limited resources, low computation and communication costs are required when using mobile devices. Since public key encryptions usually execute more computations, we adopt symmetric encryptions to reduce the computation cost. Furthermore, the number of round trips between a user and a network system is minimized to two, which decreases the communication cost.
3. **Forward secrecy.** Forward secrecy, one of the most important security properties in communication, makes it impossible for attackers to derive previous session keys even though the long-term key material has been revealed. We provide this capability in our proposed protocol.
4. **Formal security proofs.** To guarantee security, the security of an authentication protocol should be formally defined and proved. Some EAP methods, such as EAP-MD5 and EAP-LEAP, were believed to be secure until some security flaws were found later. We formally define secure mutual authentication, secure session key exchange, and forward secrecy according to [3], [9] and theoretically prove the security of our protocol based on that of the secure cryptographic components we adopted.

3 The Proposed EAP Method

In our proposed EAP method, we define three participants, U, AP, and AS, which denote a user, an access point, and an authentication server, respectively. We assume that the communication channel between AS and AP is secure. The execution flow of the proposed method is shown in Fig. 2. In the registration phase, U and AS negotiate shared credentials for future authentication. The registration phase is performed offline. After registering with AS, U first normally authenticates AS and AP. In the normal authentication process, AS updates some parameters shared with U and helps U and AP share time-limited credentials for a fast reconnect process. To accelerate the following authentication processes, U is able to perform the fast reconnect process with frequently visited APs. To deal with an AP that is not frequently visited, U and AP periodically delete their expired credentials if the time interval between the last visit and the present visit exceeds a certain predefined limit. Thus, U can perform the fast reconnect process only with frequently visited APs if the shared credentials exist. Otherwise, U performs the normal authentication process. Besides, if the fast connect process fails due to some reasons (e.g., man-in-the-middle attacks, asynchronous updating of shared credentials, or a crashed AP), U will perform the normal authentication process to refresh the shared credentials with the AP.

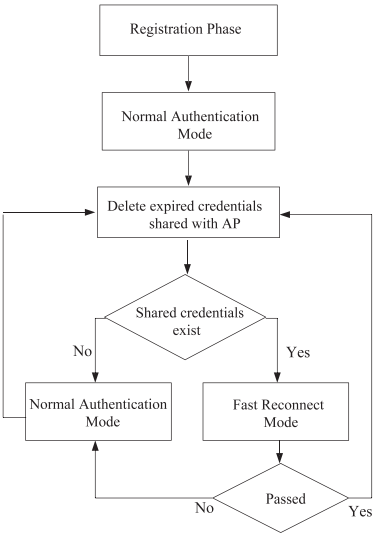

Fig. 2. User execution flowchart of the proposed method.

In the registration phase, the server AS and the user U share credentials, which is performed offline. The credentials are 1) a long-term key k, 2) a password pw chosen by the user, and 3) a random one-time key y for protecting the long-term key in the current session. Then, the user stores (UID,SID,k,y) and the server stores (UID,k,y,pw,τ) and $(\bar{y},\bar{\tau})$ as a registration record of the user in its database, where UID and SID are the identities of the user and the server, respectively, $\tau = E_{k\oplus y}(UID)$, and the variables, $(\bar{y},\bar{\tau})$, are initially null. The cryptographic functions utilized in the method are 1) H, a one-way hash function and 2) (E_K, D_K), symmetric encryption and decryption functions with a key K.

In the normal authentication process of our EAP method (see Fig. 3), there are eight steps in the authentication after EAP-Request/Identity (i.e., Step 2 in the procedure of IEEE 802.1x shown in Fig. 1).

1. $U \rightarrow AP$: The user computes $tid_1 = E_{k\oplus y}(UID)$ and $tid_2 = E_{k\oplus y}(UID\|N_C)$, where N_C is a string randomly chosen by the user, and sends a temporal identity $TID = \{tid_1, tid_2\}$ to respond to the identity request.
2. $AP \rightarrow AS$: AP forwards $TID = \{tid_1, tid_2\}$ to AS.
3. $AS \rightarrow AP$: AS searches its database to find UID and then extracts the correct encryption key. There are two cases of finding the user U and the correct encryption key.

   Case 1: If $\tau = tid_1$ and UID is a prefix of $D_{k\oplus y}(tid_2)$, AS 1) retrieves the suffix of $D_{k\oplus y}(tid_2)$ as N_C, 2) sets $(\bar{y},\bar{\tau}) \leftarrow null$ if they are not null, and 3) continues performing the processes.

   Case 2: If $\bar{\tau} = tid_1$ and UID is a prefix of $D_{k\oplus\bar{y}}(tid_2)$, AS 1) retrieves the suffix of $D_{k\oplus\bar{y}}(tid_2)$ as N_C, 2) sets $y_N \leftarrow y$, where y_N is the one-time key used in the next session of the normal authentication process, $y \leftarrow \bar{y}$, $\tau \leftarrow \bar{\tau}$, and $(\bar{y},\bar{\tau}) \leftarrow null$, and 3) continues performing the processes.

   Otherwise, AS aborts the session.

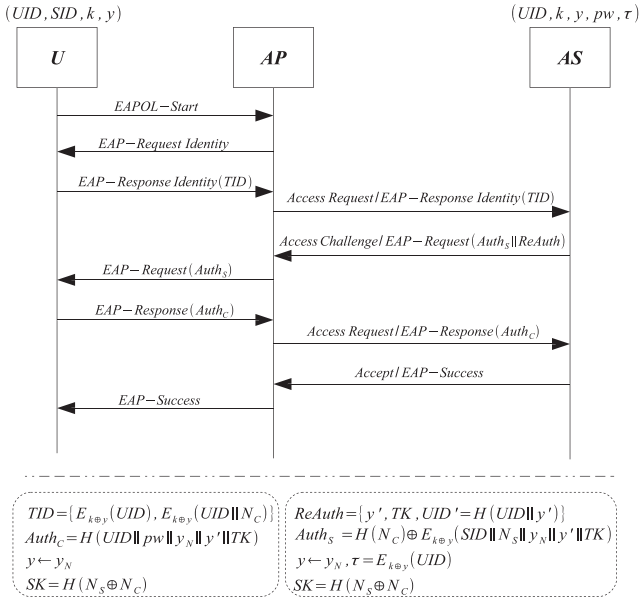

Fig. 3. The normal authentication process.

Then, AS randomly chooses 1) a nonce N_S, 2) y_N if it has not been assigned, and 3) another one-time key y' and a temporal key TK, where (y', TK) will be shared between AP and U for the fast reconnect process, and computes $Auth_S = H(N_C) \oplus E_{k\oplus y}(SID\| N_S\|y_N\|y'\|TK)$. Finally, AS sends $Auth_S$ and $ReAuth$ to AP, where $ReAuth$ contains y', TK, and a temporal identity $UID' = H(UID\|y')$.

4. $AP \rightarrow U$: AP saves $ReAuth$ and forwards $Auth_S$ to U.
5. $U \rightarrow AP$: U computes $D_{k\oplus y}(Auth_S \oplus H(N_C))$. If a prefix of the decryption result is SID, U stores (N_S, y_N, y', TK) obtained from the decryption result and sets $y \leftarrow y_N$ and then sends $Auth_C = H(UID\|pw\|y_N\|y'\|TK)$ to AP. Otherwise, U aborts the session.
6. $AP \rightarrow AS$: AP forwards $Auth_C$ to AS.
7. $AS \rightarrow AP$: If AS receives $Auth_C$ and $Auth_C = H(UID\|pw\|y_N\|y'\|TK)$ holds, AS sets $y \leftarrow y_N$ and $\tau \leftarrow E_{k\oplus y}(UID)$ and sends "Accept/EAP-Success" to AP. The session key SK is $H(N_S \oplus N_C)$. Otherwise, AS sets $\bar{y} \leftarrow y$, $y \leftarrow y_N$, $\bar{\tau} \leftarrow \tau$, and $\tau \leftarrow E_{k\oplus y}(UID)$ and aborts the session.
8. $AP \rightarrow U$: AP forwards "EAP-Success" to U if it receives an acceptance notification from AS; otherwise, AP deletes $ReAuth$.

The shared credential y is updated in every session in the proposed scheme, so it is required to synchronize the updating of y between AS and U. In our protocol, AS is in charge of the synchronization. If AS does not receive $Auth_C$ or $Auth_C = H(UID\|pw\|y_N\|y'\|TK)$ does not hold in Step 7 of the normal authentication process, AS has no idea whether U has updated y and thus an asynchronous updating of y might occur.

To deal with the asynchronous updating, AS keeps both y and y_N by setting $\bar{y} \leftarrow y$ and $y \leftarrow y_N$ and determines which one is correct in the next run of the normal authentication.

In the next session, AS can determine whether U has updated y in Step 3. If TID is successfully decrypted with $k \oplus y$, AS knows that U has set $y \leftarrow y_N$. If TID is successfully decrypted with $k \oplus \bar{y}$, U uses $\bar{y}$ as the current credential. AS does not know that TID is sent from U or replayed by the adversary until it receives the correct $Auth_C$.

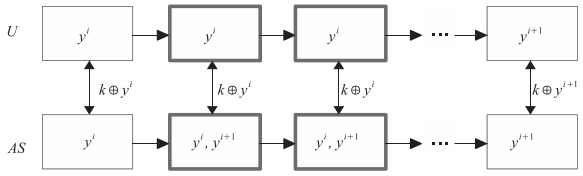

Fig. 4. The resynchronization procedure for the one-time key.

The server can cope with the problem of asynchronous updating on the one-time key in the client side because the server always saves both the old and the new one-time keys when it does not receive the correct response $Auth_C$. Fig. 4 shows an example. In a situation where the client fails to update the old one-time key y^i to the new one, the server can use the saved y^i to communicate with the client until the client finally updates the one-time key to y^{i+1}.

In the fast reconnect process (see Fig. 5), (UID', y', TK, τ') has been shared between AP and U in advance, where $\tau' = E_{TK\oplus y'}(UID')$. There are four steps in the fast reconnect process after EAP-Request/Identity takes place.

1. $U \rightarrow AP$: The user computes and sends a temporal identity $TID' = \{E_{TK\oplus y'}(UID'), E_{TK\oplus y'}(UID'\|\bar{N}_C)\}$, where $\bar{N}_C$ is a randomly chosen string, to respond to the identity request.
2. $AP \rightarrow U$: After receiving $TID' = \{tid'_1, tid'_2\}$, AP searches its database to check whether $\tau' = tid'_1$ exists, and it computes $D_{TK\oplus y'}(tid'_2)$ to check if the corresponding UID' is a prefix of the decryption result. If not, AP aborts the session. Then, AP randomly chooses a nonce $\bar{N}_S$ and a one-time key y'_N, which will be used in the next session of the fast reconnect process with the same AP to replace y', and then computes $ReAuth_S = H(\bar{N}_C) \oplus E_{TK\oplus y'}(SID\|\bar{N}_S\|y'_N)$. Finally, AP sends $ReAuth_S$ to U.
3. $U \rightarrow AP$: U computes $D_{TK\oplus y'}(ReAuth_S \oplus H(\bar{N}_C))$. If the prefix of the decryption result is SID, U stores $(\bar{N}_S, y'_N)$ and then sets $y' \leftarrow y'_N$ and sends $ReAuth_C = H(UID'\|y'_N)$ to AP. Otherwise, U aborts the session and starts a new session of the normal authentication process.
4. $AP \rightarrow U$: AP verifies whether $ReAuth_C = H(UID'\| y'_N)$. If the condition holds, AP sets $y' \leftarrow y'_N$ and $\tau' \leftarrow E_{TK\oplus y'}(UID')$ and sends "EAP-Success" to U. The session key SK is $H(\bar{N}_S \oplus \bar{N}_C)$. Otherwise, AP aborts the session.

4 Formal Security Model

In this section, we define a formal security model for mutual authentication, secure key exchange, and forward secrecy.

4.1 The Security Model

Bellare and Rogaway first proposed a theoretical security proof for an authentication and key agreement protocol with a symmetric two-party setting case, called BR93-Model

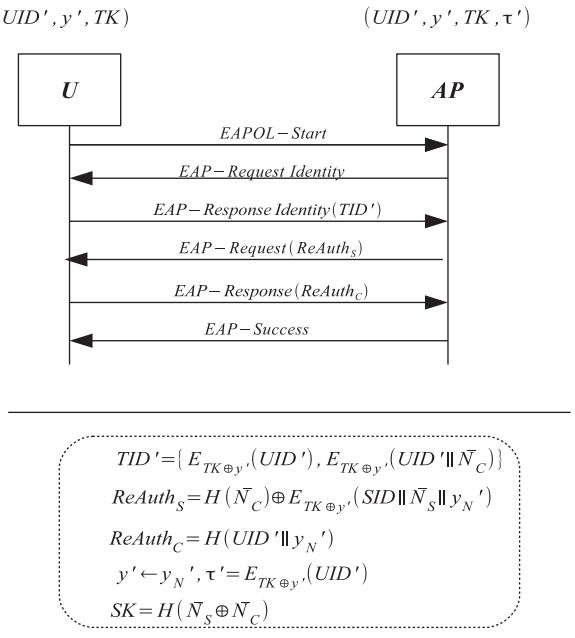

Fig. 5. The fast reconnect process.

[9]. We use BR93-Model as a foundation to prove that our protocol meets all of the EAP security requirements.

$\Pi^i_{A,B}$ denotes the client oracle that plays the role A to interact with B in the ith session, and $\Pi^j_{B,A}$ denotes the server oracle that plays the role B to interact with A in the jth session. Let P be the proposed symmetric-based authentication protocol. During the execution of P, there exists an adversary E, which is a probabilistic polynomial-time Turing machine. Our communication model and security notions follow BR93-Model. In P, there are two partner oracles, $\Pi^i_{A,B}$ and $\Pi^j_{B,A}$, and an adversary E that can control the entire network and obtain the transmitted data in the past processes. We define the capability of adversary E, which can be captured by the following queries:

- $Execute(\Pi^i_{A,B}, \Pi^j_{B,A})$. This query models all kinds of passive attacks in which a passive adversary can eavesdrop all transmitted data between $\Pi^i_{A,B}$ and $\Pi^j_{B,A}$ in P.
- $Send(\Pi^i_{A,B}, m)$. This query models active attacks in which an adversary sends a message to $\Pi^i_{A,B}$. The adversary gets the response message according to the sending message m in P. An adversary can also initiate a session by setting $m = \lambda$.
- $Send(\Pi^j_{B,A}, m)$. This query models active attacks in which an adversary sends a message to $\Pi^j_{B,A}$. The adversary gets the response message according to the sending message m in P .
- $Reveal(\Pi^i_{A,B})$. This query models the exposure of the session key of instance i. This query is only valid to A when the role actually holds a session key.
- $Corrupt(\Pi^i_{A,B})$. This query models the compromise of the long-term key of instance i. This query is only valid to A when the role actually holds a long-term key.
- $Test(\Pi^i_{A,B})$. When $\Pi^i_{A,B}$ accepts and shares a session key with the partner oracle $\Pi^i_{B,A}$, adversary E can make this query and try to distinguish a real session key from a random string. This query models an adversary's query to the $Test$ oracle. It will return the real session key or a randomly chosen string to the adversary according to the value of a random coin bit.

4.2 The Security Definitions

Definition 1 (Matching conversations). *Fix a number of moves $R = 2\rho - 1$ and an R-move protocol P. Run P in the presence of an adversary E and consider two oracles $\Pi^s_{A,B}$ and $\Pi^t_{B,A}$ that engage in conversations K and K', respectively. $(\tau_m, \alpha_m, \beta_m)$ encodes that some oracle was asked α_m and responded with β_m at time τ_m.*

1. *We say that K' is a matching conversation to K if there exists $\tau_0 < \tau_1 < \cdots < \tau_R$ and $\alpha_1, \beta_1, \ldots, \alpha_\rho, \beta_\rho$ such that K is prefixed by $(\tau_0, \lambda, \alpha_1), (\tau_2, \beta_1, \alpha_2), (\tau_4, \beta_2, \alpha_3), \ldots, (\tau_{2\rho-4}, \beta_{\rho-2}, \alpha_{\rho-1}), (\tau_{2\rho-2}, \beta_{\rho-1}, \alpha_\rho)$ and K' is prefixed by $(\tau_1, \alpha_1, \beta_1), (\tau_3, \alpha_2, \beta_2), (\tau_5, \alpha_3, \beta_3), \ldots, (\tau_{2\rho-3}, \alpha_{\rho-1}, \beta_{\rho-1})$.*
2. *We say that K is a matching conversation to K' if there exists $\tau_0 < \tau_1 < \cdots < \tau_R$ and $\alpha_1, \beta_1, \ldots, \alpha_\rho, \beta_\rho$ such that K' is prefixed by $(\tau_1, \alpha_1, \beta_1), (\tau_3, \alpha_2, \beta_2), (\tau_5, \alpha_3, \beta_3), \ldots, (\tau_{2\rho-3}, \alpha_{\rho-1}, \beta_{\rho-1}), (\tau_{2\rho-1}, \alpha_\rho, *)$ and K is prefixed by $(\tau_0, \lambda, \alpha_1), (\tau_2, \beta_1, \alpha_2), (\tau_4, \beta_2, \alpha_3), \ldots, (\tau_{2\rho-4}, \beta_{\rho-2}, \alpha_{\rho-1}), (\tau_{2\rho-2}, \beta_{\rho-1}, \alpha_\rho)$.*

Definition 2 (Mutual authentication). *We say that Π is a secure mutual authentication protocol if for any polynomial time adversary E.*

1. *Matching conversation implies acceptance. If oracles $\Pi^s_{A,B}$ and $\Pi^t_{B,A}$ have matching conversations, both oracles accept.*
2. *Acceptance implies matching conversation. The probability of $No - Matching^E(k)$ is negligible, where k is a security parameter and $No - Matching^E(k)$ is the event that there exists s, t, P, and Q, such that $\Pi^s_{P,Q}$ accepted and there is no oracle $\Pi^t_{Q,P}$ which engaged in a matching conversation.*

A secure mutual authentication protocol also withstands man-in-the-middle attacks since [9] has proved that the probability of $Multiple - Matching^E(k)$ is negligible for a secure mutual authentication protocol, where $Multiple - Matching^E(k)$ is the event that there are at least two oracles $\Pi^t_{Q,P}$ and $\Pi^{t'}_{Q,P}$ which have matching conversations with $\Pi^s_{P,Q}$.

Definition 3 (Secure key exchange). *A protocol Π is a secure mutual authentication and key exchange protocol if Π is a secure mutual authentication protocol and the following properties are satisfied.*

1. *An adversary engages in the execution of Π with $\Pi^s_{A,B}$ and its partner $\Pi^t_{B,A}$. Then both oracles always share the same session key.*
2. *For any polynomial-time adversary*

$$E, advantage^E(k) = (Pr[Good - Guess^E(k)] - 1/2)$$

is negligible where k is the security parameter and $Good - Guess^E(k)$ is the event that the adversary E outputs the right answer to the Test query.

Definition 4 (Forward secrecy). *A protocol provides forward secrecy if, for any polynomial-time adversary E, the advantage $advantage^E(k) = (Pr[Good - Guess^E(k)] - 1/2)$ of distinguishing a previous session key from a random string is negligible after the long-term private keys of all parties are compromised.*

4.3 Security Against an Adaptively Chosen Ciphertext Attack

We introduce the adaptively chosen ciphertext attack in a symmetric encryption scheme. Let $\Gamma = (K, E, D)$ be a symmetric encryption scheme where K is a key generation algorithm that takes a security parameter k as an input, E is a probabilistic encryption algorithm, and D is the deterministic decryption algorithm.

Step 1: The key generation algorithm generates a secret key k.

Step 2: The adversary makes a series of encryption/decryption queries by sending a sequence of ciphertexts/plaintexts, such as $(y_0, ``decrypt'')$, $(y_1, ``encrypt'')$, $\ldots$, $(y_i, ``encrypt'')$, to the encryption/decryption oracles. The oracles make use of the secret key k and returns the corresponding ciphertext/plaintext to the adversary for every input.

Step 3: The adversary selects two messages, x'_0 and x'_1, as plaintexts and sends them to the encryption oracle. The encryption oracle chooses $b \in \{0, 1\}$ by tossing a coin, and then encrypts x'_b and returns the result y'_b to the adversary.

Step 4: The adversary performs Step 2 again by making a series of encryption/decryption queries. But the adversary is not allowed to query the decryption oracle with the input y'_b.

Step 5: Finally, the adversary outputs $b' \in \{0, 1\}$ as the guess of the value of b.

A symmetric encryption scheme is secure against the adaptively chosen ciphertext attack if, for any polynomial-time adversary, $(Pr[b' = b] - 1/2)$ is negligible.

4.4 The Security Properties

We can formally prove that the proposed protocol possesses the properties of mutual authentication, secure key exchange, and forward secrecy if the underlying symmetric encryption scheme is secure against the adaptively chosen ciphertext attack.

Theorem 1 (Mutual authentication). *If the symmetric encryption scheme used in the proposed protocol Π is secure against the adaptively chosen ciphertext attack, then protocol Π is a secure mutual authentication protocol.*

Theorem 2 (Secure key Exchange). *If the encryption scheme used in the proposed protocol Π is secure against the adaptively chosen ciphertext attack, then protocol Π is a secure key exchange protocol.*

Theorem 3 (Forward secrecy). *If the encryption scheme used in the proposed protocol Π is secure against the adaptively chosen ciphertext attack, then protocol Π satisfies forward secrecy.*

The proofs for the three theorems are shown in the supplementary file, which is available online.

TABLE 1
The Comparison Table

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
MD5	No	No	No	No	No	No	No	Yes	No	2
TLS	Yes	Yes	Yes	Yes	Yes*	Yes	Yes	No	No	4
TTLS	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	5
PEAP	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	7
LEAP	Yes	Yes	No	Yes	No	No	Yes	Yes	No	4
FAST	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	5‡
SPEKE	Yes	Yes	Yes	Yes	No	No	Yes	Yes	No	3
TLS-SEM	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	2
double-TLS	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	6
SRP	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	4
SSC	Yes	Yes	Yes	Yes	No	No	No	No**	No	2
Park	Yes	Yes	Yes	Yes	No	No	No†	Yes	No	-
Yoon	Yes	Yes	Yes	Yes	No	No	No†	Yes	No	-
Juang1	Yes	Yes	Yes	Yes	No	No	No†	Yes	No	-
Juang2	Yes	Yes	Yes	Yes	Yes	No	No†	Yes	No	-
Ours	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	2§

C1: Mutual authentication
C2: Generation of session key
C3: Resistance to dictionary attacks
C4: Resistance to man-in-the-middle attacks
C5: End-user identity hiding
C6: Fast reconnect
C7: Forward secrecy
C8: No requirement for certificate maintenance
C9: Provable security
C10: The number of EAP request/response round trips

* [6] and [18] provide EAP-TLS for end-user identity hiding.
** The asymmetric key exchange model requires for certificate maintenance.
† Only half forward secrecy
‡ Requirement of five EAP request/response round trips at least
§ It takes 2 round trips if the user selects the EAP type and then indicates it in the EAP-response identity message while 0.5 additional round trip will be needed if the server selects the EAP type.

5 Comparisons

We compare the previous EAP methods and authentication mechanisms for WLANs with our proposed EAP method. We compare them from the viewpoints of the EAP method requirements defined in RFC 4017 and other key properties, including forward secrecy, and maintenance of certificates in Table 1, and we also show the computation time comparison among the methods achieving all security properties in Table 2

The mandatory requirements. EAP-MD5 does not achieve both mutual authentication and session key generation. Besides, it is vulnerable to dictionary attacks and man-in-the-middle attacks [23]. Certificate-based EAP methods, such as EAP-TLS, EAP-TTLS, EAP-PEAP, and EAP-FAST provide mutual authentication and session-key generation. These methods can withstand dictionary attacks and man-in-the-middle attacks. EAP-LEAP has been shown to be vulnerable to dictionary attacks [36]. In addition, the symmetric-based methods, such as EAP-SPEKE, EAP-TLS-SEM, EAP-double-TLS, EAP-SRP use the Diffie-Hellman key exchange [27] to generate session keys, which provide mutual authentication and are also immune to man-in-the-middle attacks and dictionary attacks. Besides, EAP-SSC, and the protocols of Park et al., Yoon et al., and Juang et al. are compliant with the mandatory requirements. Our proposed EAP method also satisfies the mandatory requirements defined in RFC 4017.

End-user identity hiding. End-user identity hiding means that a user's identity is encrypted during the

TABLE 2
Comparison of Computation Time

	Computation Time of Authentication		Decrement of Computation Time	
	Client	Server	Client	Server
Our Proposed Protocol	$2T_{HASH128} + 2T_{AES128} = 2.636 \times 10^3$ CPU cycles = 3.73×10^{-3} msec	$3T_{HASH128} + 2T_{AES128} = 3.199 \times 10^3$ CPU cycles = 4.53×10^{-3} msec	-	-
EAP-TLS	$T_{DH2048} + T_{RSA_S2048} + T_{RSA_V2048} = 8.236 \times 10^7$ CPU cycles = 114.39 msec	$T_{DH2048} + T_{RSA_V2048} = 2.559 \times 10^7$ CPU cycles = 35.54 msec	99.99%	99.99%
EAP-TTLS, EAP-PEAP, EAP-FAST	$T_{DH2048} + T_{RSA_V2048} = 2.559 \times 10^7$ CPU cycles = 35.54 msec	$T_{DH2048} = 2.472 \times 10^7$ CPU cycles = 34.33 msec	99.99%	99.99%
EAP-TLS-SEM, EAP-double-TLS, EAP-SRP	$T_{DH2048} = 2.472 \times 10^7$ CPU cycles = 34.33 msec	$T_{DH2048} = 2.472 \times 10^7$ CPU cycles = 34.33 msec	99.99%	99.99%

$T_{HASH128}$: the computation time of generating a 128-bit message digest by using the SHA-256 algorithm
T_{AES128}: the computation time of a 128-bit AES-CBC encryption/decryption
T_{RSA_S2048}: the computation time of signing an RSA signature with a 2048-bit key
T_{RSA_V2048}: the computation time of verifying an RSA signature with a 2048-bit key)
T_{DH2048} : the computation time of a Diffie-Hellman key agreement with a 2048-bit key
$T_{HASH128} \approx 5.63 \times 10^2$ (cycles per operation) $T_{AES128} \approx 7.55 \times 10^2$ (cycles per operation)
$T_{RSA_S2048} \approx 56.77 \times 10^6$ (cycles per operation) $T_{RSA_V2048} \approx 0.87 \times 10^6$ (cycles per operation)
$T_{DH2048} \approx 24.72 \times 10^6$ (cycles per operation)

authentication processes. EAP-TTLS, EAP-PEAP, and EAP-FAST all establish secure tunnels after the server is authenticated by the client. Then the client is authenticated by the server using a legacy method via the secure tunnel. Since the user's identity is transmitted in the secure tunnel before she/he is authenticated by the server, the user's identity is protected by encryption. Therefore, these EAP methods are able to hide the end-user identities. The user identities in EAP-SEM and EAP-double-TLS are also protected because they use TLS tunnels. EAP-MD5, EAP-LEAP, EAP-SPEKE, and EAP-SSC do not provide identity privacy due to the lack of establishing secure tunnels. Badra and Urien [6] and Dierks and Rescorla [18] can provide EAP-TLS for end-user identity hiding. Besides, the protocols of Park et al. and Yoon et al. and the first protocol of Juang et al. are vulnerable to an offline dictionary attack upon identity privacy. A user's identity will be known by attackers owing to a low entropy of each of all possible users' identities. Therefore, their schemes do not meet identity privacy. Our method can provide identity privacy because the user identity UID is encrypted in the communication.

Fast reconnect. Certificate-based EAP methods support fast reconnections to improve performance. These methods quickly establish a connection between a client and a server. This capability can reduce the number of exchanged messages or trips. EAP-FAST, EAP-SEM, EAP-double-TLS, and EAP-SRP are able to support fast reconnections and our method also achieves this.

Forward secrecy. The protocols of Park et al., Yoon et al., and Juang et al. only support half forward secrecy. If adversaries know the long-term keying material on the client's side, the adversaries can compute the past session keys. Therefore, they only provide half forward secrecy. In addition, EAP-MD5 and EAP-SSC do not support forward secrecy.

Maintenance of certificate. All of the certificate-based EAP methods rely on certificate authorities issuing certificates to the servers, but only EAP-TLS requires that all clients must apply the certificates. Each client must install a certificate on its device. This will greatly increase the cost of administration, and maintaining certificate revocation lists adds an additional heavy load. In symmetric-key-based methods, AS can simply revoke the users by discarding the shared secrets.

The number of EAP request/response round trips. EAP-MD5 only requires two EAP request/response round trips for authentication, but it does not achieve mutual authentication. EAP-TLS takes four EAP request/response round trips for authentication, but it requires that the certificates should be installed on the server and the clients. In EAP-TTLS and EAP-PEAP, it is unnecessary to install certificates on the clients, and after finishing server authentication using a TLS handshake, legacy EAP methods, such as EAP-MD5, are used to authenticate the clients. Therefore, EAP-TTLS and EAP-PEAP require five and seven EAP request/response round trips for authentication, respectively. EAP-LEAP performs the Microsoft Challenge Handshake Authentication Protocol (MS-CHAP) twice for mutual authentication. Therefore, it requires four EAP request/response round trips. EAP-FAST adopts the TLS handshake to establish a tunnel key. Then the actual authentication uses MS-CHAP or One-Time Password (OTP) [11]. Therefore, EAP-FAST needs at least five EAP request/response round trips for authentication. EAP-SPEKE contains three EAP request/response round trips. The protocols of Park et al., Yoon et al., and Juang et al. are not based on the EAP format. Our proposed EAP

method satisfies all of the properties with only two EAP request/response round trips.

Computation time comparison. EAP-TLS, EAP-TTLS, EAP-PEAP, EAP-FAST, EAP-SEM, and EAP-double-TLS allow a client and a server to communicate securely using a handshake procedure. In the handshake procedure, both the client and the server perform the key exchange algorithm. Moreover, the client must verify the certificate of the server in EAP-TLS, EAP-TTLS, EAP-PEAP, and EAP-FAST and the server must verify the signature signed by the client in EAP-TLS. These procedures rely on public-key computations, such as RSA, DSA, or Diffie-Hellman computations. In Table 2, we compare the computation time of the handshake performed in EAP-TLS, EAP-TTLS, EAP-PEAP, EAP-FAST, EAP-SEM, EAP-double-TLS, and EAP-SRP with that of our proposed method.

We assume that the key exchange algorithm and certificate verification are implemented by Diffie-Hellman and RSA, respectively. We adopted AES and SHA-256 to implement our proposed method. All simulations are performed by the CryptoPP cryptographic library on an Arm Cortex-A8 machine with CPU frequency 0.72 GHZ. It is important to note that a 2,048-bit DH key and a 2,048-bit RSA key are approximately with the same strength as a 128-bit AES key. Therefore, we compare the performance of the proposed method with that of 2,048-bit DH key agreement and 2,048-bit RSA verification.

These methods require at least an asymmetric computation in the server and the client. Our protocol performs only symmetric encryption/decryption without asymmetric ones, which are time-consuming or energy-consuming. Thus, the computation cost of our protocol is greatly reduced by more than 99.9 percent as compared to the other protocols that have the same level of security as ours.

6 Conclusion

The proposed EAP method satisfies the requirements defined in RFC 4017. Other important features, such as forward secrecy, were also included in our proposed EAP method. In addition, we took advantage of secure symmetric encryption schemes and hash functions to avoid exponentiation computations and to achieve security requirements without maintaining certificates. Finally, we have also provided formal security proofs to demonstrate that our EAP method is truly secure.

Acknowledgments

This work was partially supported by the National Science Council of the Taiwan under grants NSC 101-2219-E-110-003, NSC 101-2219-E-110-005, and "Aim for the Top University Plan" of the National Sun Yat-sen University and Ministry of Education, Taiwan, R.O.C.

References

[1] B. Adoba, L. Blunk, J. Vollbrecht, J. Carlson, and E. Levkowetz, "Extensible Authentication Protocol (EAP)," RFC 3748, June 2004.

[2] B. Adoba, D. Simon, and R. Hurst, "The EAP-TLS Authentication Protocol," RFC 5216, Mar. 2008.

[3] R. Anderson, *Proc. Fourth ACM Ann. Conf. Computer and Comm. Security,* Invited Lecture, 1997.

[4] H. Andersson, S. Josefsson, G. Zorn, D. Simon, and A. Parlekar, "Protected EAP Protocol (PEAP)," *IETF Draft,* draft-josefsson-pppext-eap-tls-eap-04.txt, Sept. 2002.

[5] M. Badra and I. Hajjeh, "Key-Exchange Authentication Using Shared Secrets," *Computer,* vol. 39, no. 3, pp. 58-66, 2006.

[6] M. Badra and P. Urien, "Adding Client Identity Protection to EAP-TLS SmartCards," *Proc. IEEE Wireless Comm. and Networking Conf.,* 2007.

[7] M. Badra and P. Urien, "EAP-Double-TLS Authentication Protocol," http://tools.ietf.org/html/draft-badra-eap-double-tls-04, Oct. 2005.

[8] M. Badra and A. Serhrouchni, "A New Secure Session Exchange Key Protocol for Wireless Communications," *Proc. IEEE 14th Int'l Symp. Personal, Indoor and Mobile Radio Comm. (PIMRC),* pp. 2765-2769, 2003.

[9] M. Bellare and P. Rogaway, "Entity Authentication and Key Distribution," *Proc. 13th Ann. Int'l Cryptology Conf. Advances in Cryptology,* pp. 22-26, 1993.

[10] J. Carlson, B. Aboba, and H. Haverinen, "EAP SRP-SHA1 Authentication Protocol," July 2001.

[11] N. Cam-Winget, D. McGrew, J. Salowey, and H. Zhou, "The Flexible Authentication via Secure Tunneling Extensible Authentication Protocol Method (EAP-FAST)," *RFC 4851,* May 2007.

[12] J.C. Chen, M.C. Jiang, and Y.W. Liu, "Wireless LAN Security and IEEE 802.11i," *IEEE Wireless Comm.,* vol. 12, no. 1, pp. 27-36, Feb. 2005.

[13] J. Chen and Y. Wang, "Extensible Authentication Protocol (EAP) and IEEE 802.1x tutorial and empirical experience," *IEEE Comm. Magazine,* vol. 43, no. 12, pp. 26-32, Dec. 2005.

[14] P. Congdon, B. Aboba, A. Smith, G. Zorn, and J. Roese, "IEEE 802.1X Remote Authentication Dial in User Service (RADIUS)," *RFC 3580,* Sept. 2003.

[15] M.T. Dandjinou and P. Urien, "EAP-SSC Protocol," *Proc. Third Int'l Conf. Networking (ICN '04),* 2004.

[16] R. Dantu, G. Clothier, and A. Atri, "EAP Methods for Wireless Networks," *Computer Standards and Interfaces,* vol. 29, no. 3, pp. 289-301, Mar. 2007.

[17] T. Dierks and C. Allen, "The TLS Protocol Version 1.0," *RFC 2246,* Jan. 1999.

[18] T. Dierks and E. Rescorla, "The TLS Protocol Version 1.2," *RFC 5246,* Aug. 2008.

[19] P. Eronen and H. Tschofenig, "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)," *RFC 4279,* Dec. 2005.

[20] P. Funk and B.W. Simon, "EAP Tunneled TLS Authentication Protocol Version 0 (EAP-TTLSv0)," *IETF Draft,* draft-funk-eap-ttls-v0-00.txt, Feb. 2005.

[21] M. Gast 802.11 Wireless Network: The Definitive Guide, O'REILLY, 2002.

[22] R. Housley and W. Arbaugh, "Security Problems in 802.11-Based Networks," *Comm. ACM,* vol. 46, no. 5, pp. 35-39, 2003.

[23] H. Hwang, G. Jung, K. Sohn, and S. Park, "A Study on MITM (Man in the Middle) Vulnerability in Wireless Network Using 802.1X and EAP," *Proc. Int'l Conf. Information Systems Security,* pp. 164-170, 2008.

[24] D. Jablon, "The SPEKE Password-Based Key Agreement Methods," *IETF Draft,* draft-jablon-speke-02.txt, Oct. 2003.

[25] W.S. Juang and J.L. Wu, "Two Efficient Two-Factor Authenticated Key Exchange Protocols in Public Wireless LANs," *Computers and Electrical Eng.,* vol. 35, no. 1, pp. 33-40, 2009.

[26] Y.M. Park and S.K. Park, "Two Factor Authenticated Key Exchange (TAKE) Protocol in Public Wireless LANs," *IEICE Trans. Comm.,* vol. E87-B, no. 5, pp. 1382-1385, 2004.

[27] E. Rescorla, "Diffie-Hellman Key Agreement Method," *RFC 2631,* June 1999.

[28] W. Simpson, "PPP Challenge Handshake Authentication Protocol (CHAP)," *RFC 1994,* Aug. 1996.

[29] D. Stanley, J. Walker, and B. Aboba, "Extensible Authentication Protocol (EAP) Method Requirements for Wireless LANs," *RFC 4017,* Mar. 2005.

[30] E.J. Yoon and K.Y. Yoo, "An Optimized Two Factor Authenticated Key Exchange Protocol in PWLANs," *Proc. Sixth Int'l Conf. Computational Science (ICCS '06),* pp. 1000-1007, 2006.

[31] H. Zhou, N. Cam-Winget, J. SaloweyFlexible, and S. Hanna, "Authentication via Secure Tunneling Extensible Authentication Protocol Version 2," http://tools.ietf.org/html/draft-ietf-emu-eap-tunnel-method-00, May 2011.

[32] ANSI/IEEE Standard 802.11, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," Dec. 1999.
[33] IEEE 802.11i-2004, "Amendment 6: Wireless LAN Medium Access Control (MAC) Security Anhancements," July 2004.
[34] IEEE Standard 802.1X-2001, "Port-Based Network Access Control," June 2001.
[35] Cisco SAFE, "WLAN Security in Depth," http://www.cisco.com/warp/public/cc/so/cuso/epso/sqfr/safwl_wp.pdf, 2012.
[36] Dictionary Attack on Cisco LEAP, http://www.cisco.com/warp/public/707/cisco-sn-20030802-leap.shtml, 2012.

Chun-I Fan received the MS degree in computer science and information engineering from National Chiao Tung University, Taiwan, in 1993, and the PhD degree in electrical engineering at National Taiwan University in 1998. From 1999 to 2003, he was an associate researcher and project leader of Telecommunication Laboratories, Chunghwa Telecom Co. Ltd, Taiwan. In 2003, he joined as a faculty of the Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan, and has been a full professor since 2010. He was also the editor-in-chief of Information Security Newsletter and is an executive director of Chinese Cryptology and Information Security Association. His current research interests include applied cryptology, cryptographic protocols, information and communication security, and he has published more than 100 technical papers. He won the Dragon PhD Thesis Award from Acer Foundation, Best PhD Thesis Award from Institute of Information & Computing Machinery in 1999, Best Student Paper Awards in National Conference on Information Security 1998 and 2007, Best Master Thesis Award from Taiwan Association for Web Intelligence Consortium in 2011, Outstanding Master Dissertation Award from Taiwan Institute of Electrical and Electronic Engineering in 2011, and Master Thesis Award from Chinese Cryptology and Information Security Association in 2012. He is a member of the IEEE and the IEEE Computer Society.

Yi-Hui Lin received the BS degree in computer science and information engineering from Tung Hai University, Taichung, Taiwan in 2004, and the MS degree in computer science and engineering from National Sun Yat-sen University, Kaohsiung, Taiwan in 2006. Currently, she is working toward the PhD degree at National Sun Yat-sen University. Her current research interests include information security and privacy, authentication protocols, and security proofs. From June to September 2010 and March 2011 to February 2012, she earned scholarships, granted by Deutscher Akademischer Austausch Dienst (DAAD), Germany, and National Science Council (NSC), Taiwan, as a visiting scholar at Center for Advanced Security Research Darmstadt (CASED) in Technische Universität Darmstadt.

Ruei-Hau Hsu received the BS and MS degrees in computer science from Tunghai University, Taiwan, in 2002 and 2004, respectively. Currently, he is working toward the PhD degree in computer science and engineering at National Sun Yat-sen University. From 2004 to 2005, he was a technical engineer of the Computer Center of Hsiuping Institute of Technology, Dali, Taiwan. From August to December 2007, he joined the International Collaboration for Advancing Security Techonology (iCAST) program to be a visiting researcher in Carneige Mellon University, America. His current research interests include information security, information privacy, and cryptographic primitives of wireless related authentication protocols and signature schemes. From June to September 2010 and March 2011 to February 2012, he earned scholarships, granted by Deutscher Akademischer Austausch Dienst (DAAD) Germany and National Science Council (NSC) Taiwan, as a visiting scholar at Center for Advanced Security Research Darmstadt (CASED) in Technische Universität Darmstadt.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.