A Handover Security Mechanism Employing Diffie-Hellman PKDS for IEEE802.16e Wireless Networks

Fang-Yie Leu, Yi-Fu Ciou, Yi-Li Huang

Department of Computer Science, Tunghai University, Taiwan
leufy@thu.edu.tw, stevennick@gmail.com, yifung@thu.edu.tw

Abstract. In this paper, we propose a handover authentication mechanism, called handover key management and authentication scheme (HaKMA), which as a three-layer authentication architecture is a new version of our previous work Diffie-Hellman-PKDS-based authentication method (DiHam for short) by improving its key generation flow and adding a handover authentication scheme to respectively speed up handover process and increase the security level for mobile stations (MS). AAA server supported authentication is also enhanced by involving an improved extensible authentication protocol (EAP). According to the analyses of this study, the HaKMA is more secure than the compared schemes, including the PKMv2 and DiHam.

Keywords: HaKMA, DiHam, PKM, WiMax, IEEE802.16, Wireless security

1 Introduction

Recently, wireless networks due to their popularity and the characteristics of convenience and high access speed have been a part of our everyday life. Through wireless systems, people can surf web contents, send emails and watch video program outdoors anytime anywhere. To satisfy the requirements of high-speed mobile wireless networks, the IEEE 802.16 Working Group in 2005 developed the IEEE 802.16e standard, known as the WiMax system, which is an extended version of IEEE 802.16 by adding mobility management and handover scheme so as to provide users with mobile broadband wireless services.

To prevent malicious attacks, the IEEE802.16 standard employs a key management and authorization mechanism called privacy key management (PKM) to authenticate users and wireless facilities [1, 2]. However, several problems have been found [1], like lacking mutual authentication, and having authorization vulnerabilities and key management failures. Also, the high complexity of its authentication mechanism and the involvement of designing errors [1] make the PKM fail to effectively protect a wireless system. To solve these problems, the IEEE Network Group proposed PKMv2 in 2005 to fix the defects of PKMv1 by adding mutual authentication and EAP support. But this enhancement also makes PKMv2 more complicated and difficult to

maintain than PKMv1if someday new shortcomings are found. On the other hand, Leu et al. [3] proposed a Diffie-Hellman-PKDS-based authentication method (DiHam) to improve some of the defects. However, the scheme does not guarantee full security since it only considers the initial network entry without providing handover and user authentication.

Therefore, in this paper, we propose a handover authentication mechanism, called handover key management and authentication system (HaKMA for short), which is an extended version of the DiHam by improving the key generation flow and adding a handover authentication scheme to respectively speed up handover process and increase the security level for mobile stations (MSs). It also enhances the AAA server authentication by employing an improved version of extensible authentication protocol (EAP). To meet different security levels of wireless communication, two levels of handover authentication are proposed. The analytical results show that the HaKMA is more secure than the DiHam, and the PKMv2.

2 Background and Related Work

2.1 The WiMax Network Architecture

Fig. 1 shows a modern multi-layer wireless network configuration. The ASN-GW is connected to a network service provider (NSP) backbone network, and BSs are directly linked to their ASN-GWs. An ASN-GW can not only communicate with other ASN-GWs via the backbone network through R3 reference points, but also directly communicate with other ASN-GWs with direct links via R4 reference points [4]. An NSP may provide many ASN-GWs to serve users. The MS may currently link to a BS, or hand over between two BSs under the same ASN-GW, called Intra-ASN-GW handover, or different ASN-GWs, called Inter-ASN-GW handover. In this study, due to limited pages, we only discuss the Intra-ASN-GW handover.

2.2 Privacy key management protocol

The PKM protocol first specified by the IEEE 802.16-2004 provides device authentication (also known as facility authentication), and PKMv2 proposed in the IEEE 802.16e-2005 is a new version of PKM protocol by correcting designing errors for security found in PKMv1 [4] and supporting user authentication.

PKMv2. PKMv2 uses X.509 digital certificates together with either a RSA public-key encryption algorithm or a sequence of RSA device authentication to authenticate communication facilities. After that, an EAP method is employed to further authenticate users. The encryption algorithms used by PKMv2 for key exchange between MS and BS are more secure than those used by PKMv1. According to IEEE802.16 standard, the optimized handover can skip the security sublayer operation and reuses old keys, such as TEKs [2, 4], or provide handover support through mobile IPv6 with other proposed scheme [5, 6].

PKMv2, allowing mutual authentication or unilateral authentication, also supports periodic re-authentication/re-authorization and key renew. All PKMv2 key derivations are performed based on the Dot16KDF algorithm defined in IEEE802.16e standard.

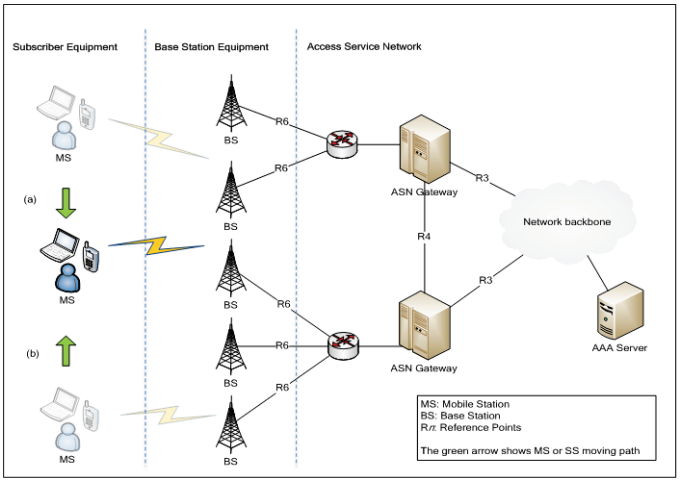

Fig. 1. The WiMax network security architecture in which MS may perform an Inter-ASN-GW handover and an Intra-ASN-GW handover.

2.3 Diffie-Hellman PKDS-based authentication

The DiHam [3] was developed based on PKMv1 by improving the key exchange flow and providing different data security levels. Basically, its key exchange process consists of two phases, authentication phase and TEK exchange phase.

In the authentication phase, AK is individually generated by BS and MS after the delivery of the Authentication-Request message and Authentication-Reply message [3]. In the TEK exchange phase, three security levels of TEK generation processes are proposed to meet users' different security requirements. This phase starts when MS sends a TEK-Exchange-Request message to BS, and ends when BS replies a TEK-Exchange-Reply message.

However, the DiHam as stated above only considered initial network entry, without dealing with handover network re-entry. If it is involved by a handover procedure, the whole process needs to be fully performed on each handover, consequently causing serious service disruption time (SDT).

3 The Proposed Security System

The HaKMA, besides retaining the advantages of DiHam's Key Exchange and AK Generation processes [3], also involves an enhanced version of an EAP method to authenticate users. A handover support to make the HaKMA more suitable for wireless environment is added as well. The HaKMA architecture consists of three isolated processes: CSK Generation process in which ASN-GW and MS mutually authenticate each other, User Authentication process in which AAA server authenticates users, and TEK Generation and Renew process in which TEKs are produced to encrypt data messages. The three processes together are called the HaKMA authentication scheme. As a layered architecture, the changing on one of the three processes does not affect the functions of others, consequently making us easier to develop a new authentication process for IEEE802.16 wireless networks when

some functions in one of the three processes need to be modified. The outputs of the three processes are sequentially CSKs, MSK and CSKs, and TEKs. In this study, we move Authenticator from BS to ASN-GW to simplify the HaKMA architecture and its handover process.

3.1 Initial Process of Network Entry

With the HaKMA, if MS has successfully completed one or two previous processes, but fails in an underlying process, the failed process is resumed from its beginning, instead of re-initiating the CSK Generation process. Of course, if the CSK Generation process fails, the HaKMA authentication scheme should be restarted. Table 1 lists the terms and functions used in this study.

Table 1. The terms and functions used in this study.

Term or function	Explanation
P	A strong prime number
g	The primitive root of P
RS_i and $RA_i, i = 1, 2$	Private keys generated by MS and Authenticator
P_{RSi} and $P_{RAi}, i = 1, 2$	Public keys generated by MS and Authenticator
$CSKi,\ i = 1,2$	Common secret keys
EXOR(x, y)	Exclusive OR function, i.e., $x \oplus y$
SEXOR$(key, data)$	Stream exclusive OR function, repeating key content to match the length of $data$, and performing exclusive OR bit by bit
Certfun(a, b, …)	Modulus function, i.e., $g^{a+b+\cdots} \bmod P$
Encrypt($PubKey$, $message$)	The standard RSA-OEAP-Encryption function for encrypting *message* into *ciphertext* with given public key *PubKey*
Decrypt($PrivKey$, $ciphertext$)	The standard RSA-OEAP-Decryption function that decrypts *ciphertext* into plaintext with given private key *PrivKey*
ADR(a, b)	A binary adder, but ignoring the carry of the greatest significant bit

CSK Generation process. The main objective of the CSK Generation process is to perform mutual authentication between Authenticator and MS, and produce two CSKs.

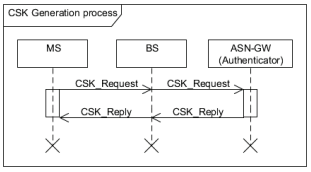

Fig. 2. Sequence chart of the CSK Generation process. BS only relays messages for MS and Authenticator.

Fig. 2 shows the sequence chart. It first establishes a secure communication channel between MS and Authenticator, and completes the following steps for initial network entry. MS and Authenticator first mutually check each other's X.509 certificate, and perform a DiHam-like process to generate CSKs which are only known to MS and Authenticator, and with which both sides encrypt those messages exchanged in User Authentication process and TEK Generation process. BS recognizes a message received by accessing its OP_Code, and relays messages for MS and Authenticator

without providing any authentication functions. This process can be further divided into two phases: Authenticator-CSK phase and MS-CSK phase.

(1) Authenticator-CSK phase

In this phase, MS first sends a CSK_Request message, of which the format is shown in Fig. 3, to Authenticator.

$$\text{OP\_Code}|\text{NS}_{MS}|\text{Cert}(MS)|P_{RM1}|P_{RM2}|\text{Capabilites}|\text{SAID}|\text{HMAC}(PubKey(MS))$$

Fig. 3. Format of a CSK_Request message sent by MS to Authenticator.

In this message, RM_1 and RM_2, two random numbers, are private keys generated by MS, P_{RM1} and P_{RM2} are two public keys where

$$P_{RMi} = g^{RM_i} \bmod P, 1 \le i \le 2 \tag{1}$$

NS_{MS}, the nonce, is a timestamp indicating when this message is created, capabilities field lists the security configurations acceptable by MS, the SAID field contains MS's primary SAID that is currently filled with the basic CID, and the hash-based message authentication code (HMAC) function produces a message signature by inputting all fields of the message as its plaintext and $PubKey(MS)$ as the encryption key. Authenticator on receiving the message checks to see whether the message signature calculated by itself by using $PubKey(MS)$ retrieved from $\text{Cert}(MS)$, and the $\text{HMAC}(PubKey(MS))$ sent by MS are equal or not. If not, implying the message has been altered, the Authenticator discards this message. If yes, it randomly selects two random numbers RA_1 and RA_2 as private keys to generate the corresponding public keys P_{RA1} and P_{RA2} where

$$P_{RAi} = g^{RA_i} \bmod P, 1 \le i \le 2, \tag{2}$$

produces two CSKs, i.e., $CSK1$ and $CSK2$, where

$$CSKi = {P_{RMi}}^{RA_i} \bmod P, 1 \le i \le 2 \tag{3}$$

and calculates the certificate function $\text{Certfun}(PubKey(MS), CSK1, CSK2)$. After that, Authenticator sends a CSK_Reply message of which the format is shown in Fig. 4 to MS. To make sure that the message is securely delivered, the HMAC which is generated by involving all other fields of the message as the inputs and the authenticator's certificate public key (i.e., PubKey(Authenticator)) as the encryption key is also added.

$$\begin{aligned}&\text{OP\_Code}|\text{NS}_{MS}|\text{NS}_{Authenticator}|\text{Cert}(Authenticator)|\\&\text{Encrypt}(PubKey(MS),P_{RA1}|P_{RA2})|\text{Certfun}(PubKey(MS), CSK1, CSK2)|\\&\text{HMAC}(PubKey(Authenticator))\end{aligned}$$

Fig. 4. Format of a CSK_Reply message sent by Authenticator to MS.

(2) MS-CSK phase

MS on receiving the CSK-Reply message checks to see whether the message has been maliciously modified or not by comparing not only the HMAC value calculated and the value received from the CSK_Reply message, but also NS_{MS} retrieved from

the message with previous nonce involved in CSK_Request message. They should be individually equal. Otherwise, the message is discarded. MS further records $\mathrm{NS}_{Authenticator}$ for later authentication, and checks to see whether the Authenticator is trustable or not by comparing the authenticator's certificate $\mathrm{Cert}(Authenticator)$ with the certificate list provided by a trustable network provider and pre-installed in the MS. If yes, MS retrieves the public key P_{RA1} and P_{RA2} by performing RSA decryption function with its own private key, calculates CSKs, i.e., $CSK1$ and $CSK2$, and the certificate function $\mathrm{Certfun}(PubKey(MS), CSK1, CSK2)$, and then compares the calculated Certfun() value with the one conveyed on CSK_Reply message sent by Authenticator where

$$CSKi = {P_{RAi}}^{RM_i} \bmod P, 1 \leq i \leq 2 \tag{4}$$

If two values are equal, the CSK Generation process terminates. MS starts the User Authentication process. Otherwise, MS discards the message and the calculated CSKs, and waits for a valid CSK-Reply message for a predefined time period. If MS cannot receive a reply from the Authenticator after timeout, it assumes the CSK Generation process fails and then restarts the process by sending a CSK-Request message to Authenticator.

User Authentication process. In this process, MS and the Authenticator first negotiate with each other to choose an EAP method. After that, Authenticator communicates with AAA server to check to see whether the user is authorized to access requested services or not.

However, EAP was originally designed for wired networks. When it is applied to wireless networks, hackers can intercept and decrypt sensitive information. To solve this problem, in this study, CSKs are involved to encrypt messages exchanged between MS and Authenticator. In fact, some EAP methods do not provide security mechanism (e.g., EAP legacy methods). Involving our encryption scheme can ensure the security of the originally insecure EAP communication between MS and BS so that hackers cannot easily decrypt sensitive information, even though an insecure EAP authentication method is used. Further, we employ EAP-AKA [7] as an EAP example, and suggest involving EAP-authentication based AK generation flow to balance handover performance and the security of the concerned system. Basically, our modular design strategy results in the fact that any authentication methods designed for wireless authentication [7] can substitute EAP-AKA to perform user authentication. A general EAP authentication process can be found in [2].

Before this process starts, the MS first generates a 160-bits random number $RAND$, and derives the EAP encryption key

$$\text{EAP Encryption Key} = \mathrm{ADR}(\mathrm{EXOR}(CSK1, RAND), CSK2) \tag{5}$$

which is generated individually by MS and Authenticator and used to encrypt and decrypt EAP messages by involving a streamed Exclusive OR method/function $\mathrm{SEXOR}(EAP\ Encrypt\ Key, EAP\text{-}messages)$. Fig. 5 illustrates our User Authentication process in which MS sends a PKMv2-EAP-Start message, of which the format is shown in Fig. 6, to notify Authenticator the start of the process. Authenticator on receiving the message extracts $RAND$ by using its private key,

derives the EAP encryption key, and replies an EAP-Request/Identity message to start security negotiation and EAP authentication.

If user is authenticated, AAA server after checking the correctness of the MAC and the RES received sends a Radius-Access/Accept message which contains an MSK 512 bits in length to Authenticator. Authenticator delivers an EAP-Success message to MS [7], indicating the user is authenticated.

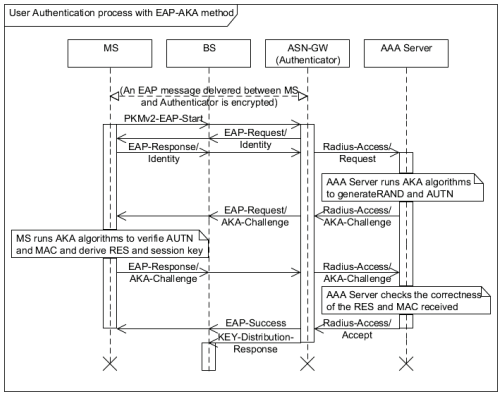

Fig. 5. The HaKMA's User Authentication process with EAP-AKA method.

$$\text{OP\_Code}|\text{NS}_{Authenticator}|\text{NS}_{MS}|\text{Encrypt}(PubKey(Authenticator), RAND)|\text{HMAC}(PubKey(Authenticator))$$

Fig. 6. The format of PKMv2-EAP-Start message sent by MS to Authenticator.

Key Distribution in the Initial Network Entry. In the HaKMA, we design a key distribution message, prefixed by KEY-Distribution, to deliver keys from a former process to current process and among serving BS, target BS and Authenticator. Authenticator sends a KEY-Distribution-Response message which contains MSK and CSKs to BS (see Fig. 5). Fig. 7 shows the format of this message. Note that both TEK_Lifetime and TEK_Count are zeros since TEKs have not been generated. Their usage will be described later. BS on receiving this message extracts MSK and CSKs from this message, and starts its TEK Generation and Renew process. Currently, both BS and MS own MSK and CSKs.

$$\text{OP\_Code}|\text{BSID}|\text{SAID}|\text{CSK1}|\text{CSK2}|\text{MSK}|\text{BS-Random}|\text{TEK\_Lifetime}|\text{TEK\_Count}|\text{TEK1}|\text{TEK2}$$

Fig. 7. The format of a KEY-Distribution-Response message sent by Authenticator to BS.

TEK Generation and Renew process. Like that in the DiHam, TEKs are individually generated by MS and BS by involving required key parameters supplied by Authenticator. Fig. 8 shows the process which starts when MS sends a TEK-Request message, of which the format is shown in Fig. 9, to BS. Both MS and

BS invoke the Dot16KDF algorithm, which accesses the first 160 bits of MSK, to individually derive AK, where

$$\text{AK} = \text{Dot16KDF("AK",2,TRUNCATE(MSK,160),160)} \tag{6}$$

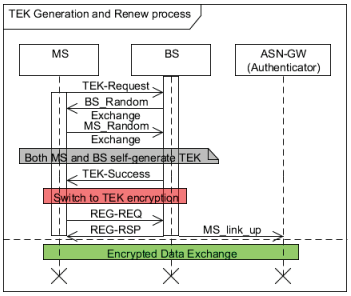

Fig. 8. Sequence chart of the TEK Generation and Renew process.

$$\text{OP\_Code}|\text{NS}_{MS}|\text{SAID}|\text{HMAC(ADR(CSK1,CSK2))}$$

Fig. 9. The format of the TEK-Request message sent by MS to BS.

BS (MS) self-generates a random number called BS_Random (MS_Random) which is also 160-bits in length. BS_Fingerprint (MS_Fingerprint) is then generated by encrypting BS_Random (MS_Random) with CSKs and AK, and delivered to MS (BS) through wireless channels. The propose is to protect the random number from being intercepted by hackers where

$$\text{BS\_Fingerprint} = \text{ADR(EXOR}(BS\_Random, CSK1), AK) \tag{7}$$

and

$$\text{MS\_Fingerprint} = \text{ADR(EXOR}(MS\_Random, CSK2), AK) \tag{8}$$

After that, a BS_Random_Exchange message, of which the format is shown in Fig. 10 and which carries BS_Fingerprint, OP_Code, and lifetime of TEKs, denoted by Key-lifetime, is sent by BS to MS.

$$\text{OP\_Code}|\text{NS}_{MS}|\text{NS}_{BS}|\text{BS\_Fingerprint}|\text{Key-lifetime}|\text{HMAC(ADR(CSK1,CSK2))}$$

Fig. 10. The format of the BS_Random_Exchange message sent by BS to MS.

MS on receiving the message decrypts the BS_Random by using one of the following formulas:

$$BS\_Random = \begin{cases} (\text{BS\_Fingerprint} - AK) \oplus CSK1 & , \text{if BS\_Fingerprint} \geq AK \\ (\text{BS\_Fingerprint} + \overline{AK} + 1) \oplus CSK1 & , \text{if BS\_Fingerprint} < AK \end{cases} \tag{9}$$

After that, MS sends MS_Random_Exchange message, of which the format is shown in Fig. 11, to BS.

$$\text{OP\_Code}|\text{NS}_{BS}|\text{NS}_{MS}|\text{MS\_Fingerprint}|\text{HMAC(ADR(CSK1,CSK2))}$$

Fig. 11. The format of the MS_Random_Exchange message sent by MS to BS. Note that Key-lifetime is not involved since it is determined by BS.

Following that, MS generates TEKs where

$$TEK_i = \text{EXOR}(\text{ADR}(\text{EXOR}(\text{MS\_Random}, AK),\text{BS\_Random}), CSK_i), 1 \le i \le 2 \quad (10)$$

BS on receiving the MS_Random_Exchange message retrieves the MS-Random number, and generates TEKs with same formula where MS_Random is calculated by using one of the following formulas:

$$MS\_Random = \begin{cases} (\text{MS\_Fingerprint} - AK) \oplus CSK2 & \text{, if MS\_Fingerprint} \ge AK \\ (\text{MS\_Fingerprint} + \overline{AK} + 1) \oplus CSK2 & \text{, if MS\_Fingerprint} < AK \end{cases} \quad (11)$$

Now both sides own the TEKs. A TEK-Success message shown in

Fig. 12 is sent by BS to inform MS the success of the TEK generation. MS registers its terminal device with BS by sending a REG-REQ message, and BS replies a REG-RSP message to finish this process. The data exchange can now be started.

$$\text{OP\_Code}|\text{NS}_{BS}|\text{NS}_{MS}|\text{HMAC(ADR(TEK1,TEK2))}$$

Fig. 12. The format of the TEK-Success message sent by BS to MS.

3.2 Handover Process of Network Re-entry

The main objective of a handover process is minimalizing the handover delay by key reuse and pre-distribution in the employed security scheme. If a user is authenticated in the initial network entry, as stated above, we assume that he/she is still authenticated after handover. This means we can reuse MSK and CSKs generated in previous processes to avoid the latency of re-authentication. For security reason, we can also renew TEKs on each handover. That means each time when MS moves to a target BS, new TEKs are required to substitute the two TEKs used by the serving BS, called previous TEKs (prev_TEKs for short to avoid confusing with the term pre_TEKs used in PKMv2).

In this study, two security levels of handover are proposed to meet different security requirements. With Level-1 handover, before prev_TEKs expires, the target BS after MS's handover reuses the same TEKs to encrypt data messages so as to shorten SDT. With Level-2 handover, on each handover, target BS temporarily reuses prev_TEKs to communicate with MS, generates new TEKs, and encrypts data messages with the new TEKs.

Summary of Key Distribution in the Network Re-entry. To deliver key information between MS and Authenticator, the KEY-Distribution-HOInfo message shown in Fig. 13 is designed to provide MS with handover support. In this message, the TEK_Count indicates the number of generated TEK pairs, and the TEK_Lifetime shows TEKs' remaining time in minutes.

OP_Code|Serving_BSID|Target_BSID|SAID|CSK1|CSK2|MSK|
TEK_Lifetime|TEK_Count|TEK1|TEK2

Fig. 13. The format of the KEY-Distribution-HOInfo message sent by a serving BS to Authenticator or Authenticator to another Authenticator.

SAID	BSID	CSK1	CSK2	MSK	TEK_Lifetime	TEK_Count	TEKs

Fig. 14. The fields of an AK Table. `TEKs` field stores TEKs.

Before MS hands over to the target BS, serving BS sends a KEY-Distribution-HOInfo message to its Authenticator. Authenticator stores the concerned keys in the MS's corresponding tuple in its Authentication key table (AK Table), a table used to keep authentication keys including MSK and CSKs for the authenticator's subordinate MSs. AK Table is indexed by SAID to identity which MS the concerned keys belong to. Fig. 14 shows the fields of this table. Authenticator further checks to see whether or not the target BS that it should newly associate with is in its BS Table, a table for recording the BSIDs of the authenticator's subordinate BSs, including those of its own and those subordinated by all its successor Authenticators, implying Authenticators are organized as a hierarchy. The table has only one field BSID. If yes, a KEY-Distribution-Response message that carries CSKs, MSK, and prev_TEKs is then sent to the target BS. If not, Authenticator needs to relay the KEY-Distribution-HOInfo message to other ASN-GW that subordinates the target BS.

The AK table should be updated dynamically each time when MS performs network entry or re-entry, MS is going to hand over, and MS key's lifetime expires. When MS initially enters a network, AK Table is updated when MS completes the User Authentication process. Authenticator saves MS's keys, leaving the TEKs field empty. The field will be filled after Authenticator receives KEY-Distribution-HOInfo message from its BS and then stores them in its AK Table. In fact, when MS is going to hand over, Authenticator extracts MS's TEKs from KEY-Distribution-HOInfo message received from serving BS and stores them in the AK Table. Finally, if the TEK_Lifetime expires, the TEK Generation and Renew process should be reinitiated to reproduce TEKs. After that Authenticator replaces the TEKs with the new TEKs in its AK Table.

Summary of TEK Generation and Renew process on Handover. Both process of the two handover security levels start when MS sends a HO_IND message to its serving BS (see Fig. 15). The serving BS then sends a MSHO_link_down message to inform ASN-GW to start transferring data messages received from MS's corresponding node (CN) to both the serving BS and the target BS, and delivers a KEY-Distribution-HOInfo message, which contains MS security attributes, such as CSKs, MSK and TEKs that serving BS currently uses, to its Authenticator.

Authenticator stores the keys in its AK Table if the target BS is under the Authenticator. Otherwise, it sends the keys to another ASN-GW during MS handover. No matter on which case, the Authenticator delivers a KEY-Distribution-Response message to the target BS. The target BS on receiving the message retrieves security

keys and saves them for future use. Now, data message transfer can be resumed before TEK Generation and Renew process starts, i.e., target BS can relay data messages to MS before new random number exchange, i.e., exchanging new MS_Random and BS_Random, is completed.

After the completion of the TEK Generation and Renew process, an MSHO_link_up message will be sent by target BS to its Authenticator to terminate sending data messages to serving BS, and the transmission of encrypted data messages can be continued.

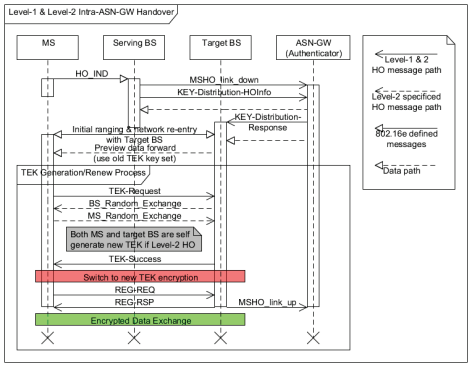

Fig. 15. The process of Level-1 & Level-2 Intra-ASN-GW handover.

Summary of Level-1 Intra-ASN-GW Handover: TEK reuse mode. Once MS chooses Level-1 handover, the KEY-Distribution-Response message sent to target BS by Authenticator includes TEKs used by serving BS. Target BS then waits for MS to complete its network re-entry, and it on receiving the TEK-Request message sent by MS as shown in Fig. 15 delivers a TEK-Success message to MS, indicating the success of TEK reuse mode. MS then sends a REG-RSP message to register itself with the BS. BS replies a REG-RSP message and sends an MSHO_link_up message to inform the ASN-GW the termination of the handover service.

Summary of Level-2 Intra-ASN-GW Handover: TEK regeneration mode. If MS selects Level-2 handover, the steps with which MS completes the network re-entry are the same as those of Level-1 Intra-ASN-GW handover. The following steps are a little different. The target BS generates a new $BS\text{-}Random$, extracts CSKs and MSK from KEY-Distribution-Response message received from Authenticator, uses Dot16KDF algorithm to generate AK, and then sends a BS_Random_Exchange message containing a newly generated BS-Fingerprint (See Eq.(7)) to MS. MS then generates a new MS-Random and sends a MS_Random_Exchange message which contains a newly generated MS-Fingerprint (See Eq.(8)) to the target BS. The BS and MS individually generate new TEKs by using the new $MS\text{-}Random$ and the $BS\text{-}Random$. After that, MS and the target BS which is now MS's serving BS communicate with each other by using the new TEKs. The following steps are the same as those of Level-1 handover. Now, previous serving BSs can no longer communicate with MS since the prev_TEKs are out of date.

4 Security Analyses and Performance Analyses

4.1 Message Integrity and Replay Attack Avoidance

Message integrity is to ensure that a message has not been changed during its delivery. In this study, all authentication messages are unchangeable once they are sent. The receiving end on receiving a message uses HMAC function to detect data tampering, retrieves the nonce conveyed on the message and saves it. The output of HMAC code conveyed on a message received can act as a verification code for that message. If at least one parameter has been changed, including the nonce, the HMAC code varies. The message will be discarded. If HMAC code passes the verification, we further verify the nonce.

The first time a message is sent, the receiving end *R* records the nonce contained in the message. If *R* receives the same or similar message again, it confirms that this is not a replay attack by comparing the nonce previously saved and the one retrieved from the message. If the nonce received is smaller than the one saved, then the message is considered as an illegal one and will be discarded. All messages delivered in the CSK Generation process and TEK Generation and Renew process are detected by this method. The DiHam scheme provides key integrity, rather than message integrity [3]. All messages exchanged in the authentication phase and TEK generation phase could be maliciously altered, but the receiving end cannot discover the change. The DiHam also lacks the involvement of nonce. Hence, it cannot discover replay attacks issued by resending an intercepted Authentication-reply message. The PKMv2 uses cipher-based message authentication code (CMAC) or HMAC to authenticate authentication messages, and detects replay attacks by employing CMAC_KEY_COUNT after the success of EAP authentication or re-authentication. However, due to involving no nonce, it cannot avoid replay attacks during the EAP authentication session.

4.2 Confidentiality

In our scheme, we analyze the confidentiality by checking to see whether exchanged information can be decrypted easily or not, and estimating the probability that a concerned message is cracked.

CSK Confidentiality. The HaKMA uses the key exchange process of the DiHam to produce two CSKs. In this process, two public keys should be exchanged between MS and Authenticator for each CSK. However, only the MS public keys P_{RM1} and P_{RM2} will be transferred through wireless channels (see Fig. 3), the Authenticator public keys P_{RA1} and P_{RA2} have been encrypted by using MS's certificate public key $PubKey(MS)$ (see Fig. 4) which can only be decrypted by using MS's certificate private key $PrivKey(MS)$. Therefore, hackers who only know P and P_{RMi} cannot easily derive $CSK1$ and $CSK2$ where

$$CSK_i = x \bmod P = {P_{RMi}}^{y} \bmod P, 1 \le i \le 2 \tag{12}$$

in which $x = {P_{RAi}}^{RM_i}$ (see Eq.(4)) and $y = RA_i$ (see Eq.(3)) are known and need to be determined, thus

$$x = {P_{RMi}}^{y}, 1 \leq i \leq 2 \tag{13}$$

and the possible combinations of x and y pair are infinite. Due to the difficulty of determining the real values for x and y, hackers can only generate CSKs by other methods, e.g., the brute-force method.

However, the number of possible 160-bit CSK values is $2^{160} \approx 1.4615 \times 10^{48}$. The probability of successfully guessing the CSK on one trial is $1/2^{160}$ which approaches to zero. However, two CSKs are used in the HaKMA. The probability will be one second of that when only one CSK is employed. Therefore, CSK confidentiality is high.

EAP Encryption Key Confidentiality. In this study, we use the EAP encryption key to encrypt and decrypt messages exchanged between MS and Authenticator. Since this encryption key is static and may be illegally decrypted, we involve the random number $RAND$, which is encrypted by Authenticator's public key (see Eq.(5)) during its delivery to generate encryption keys. Hackers cannot directly access $RAND$. Hence, it is hard to derive the EAP encryption key. Furthermore, each EAP encryption key is used only by one session, i.e., it is not used again, making hackers more difficult to collect EAP messages, and then accordingly decrypt the key. Therefore, our scheme has high EAP encryption key confidentiality.

TEK Confidentiality. Since TEKs are used to encrypt data messages, we need to keep it secure. TEKs are self-generated by MS and BS. Two random numbers *BS_Random* and *MS_Random* are also involved in the key generation process. To prevent hackers from collecting random numbers so as to derive TEKs, the two random numbers are encrypted to BS_Fingerprint and MS_Fingerprint. Since our TEK generation scheme involves ADR function [3] (see Eq.(10)), which ignores the carry, to calculate TEKs from BS_Fingerprint and MS_Fingerprint, hackers have to face four different mathematical equations (see Eq.(9) and Eq.(11)). Since each equation has up to $2^{160} \approx 1.4615 \times 10^{48}$ solutions, and all the four equations involve AK,$CSK1$ and $CSK2$ as parameters which are unknown to hackers, the possible parameter combinations for each equation is $2^{160\times3} \approx 3.1217 \times 10^{144}$. Thus, we can conclude that the TEK confidentiality is high.

If hackers try to decrypt data messages, they must find the two correct TEKs for uploading and downloading streams. If we assume that the time required to try a possible TEK is only one instruction, which takes about 1.4573×10^{29} years on a 159,000 MIPS machine. In other words, the HaKMA is a secure and safe system.

4.3 Forward and Backward Secrecy on Handover

The forward (backward) secrecy means key K_n used in session n cannot be used in session $n+1$ (session $n-1$). In Level-1 Intra-ASN-GW Handover, we reuse TEKs during and after the handover, implying Level-1 Intra-ASN-GW Handover does not provide the forward and backward secrecy. In the Level-2 Intra-ASN-GW Handover, we temporarily reuse TEKs for shortening the SDT, and generate new TEKs random numbers exchanged between MS and BS, this handover by involving the process provide the forward and backward secrecy.

PKMv2, due to considering performance optimization on Fast BS Switch (FBSS) and reusing all security attributes including TEKs does not provide the forward and backward secrecy. The DiHam process due to providing no handover support does not have forward and backward secrecy.

4.4 Man-in-the-middle attack avoidance

Man-in-the-middle attack means hackers can stay between valid MS and Authenticator to act as a legitimate Authenticator and MS. In the CSK Generation process and User Authentication process, MS and Authenticator exchange device certificate and determine whether the other side is legitimate or not. But in the CSK Generation process, we use MS's and Authenticator's public keys to encrypt important keys, such as P_{RA1} and P_{RA2} in the CSK_Reply message (see Fig. 4), and $RAND$ in the PKMv2-EAP-Start message (see Fig. 5). Receiving end needs its own private key to decrypt those encrypted messages and keys. Now we assume that a hacker, *M*, is standing between valid MS and Authenticator and wishes to steal EAP user passwords by eavesdropping EAP messages. *M* also needs to act as an Authenticator so that it can get the valid CSK to continue the following User Authentication process since our EAP messages are all encrypted by using CSKs and other parameters, like $RAND$. To complete the CSK Generation process, *M* besides relaying MS's and Authenticator's certificates also needs to replace the Authenticator certificate with its own so that it can decrypt $RAND$ for User Authentication process. However, if *M* replaces the certificate with its own, this illegal certificate will not be recognized by MS and this session will be terminated. On the other hand, if *M* continues using the real authenticator's certificate, it will not be able to decrypt the $RAND$ carried on the next PKMv2-EAP-Start message sent by MS since this value can only be decrypted by Authenticator's private key that *M* currently does not have, and the User Authentication process is still secure because all EAP messages are encrypted with both the CSKs and $RAND$. As a result, our scheme can prevent man-in-the-middle attacks.

4.5 Performance Analysis on Key Generation

Generally, in a Diffie-Hellman based authentication method, exponential operations dominate decisive performance differences [3]. In this study, two CSKs are individually generated by MS and Authenticator, and only Diffie-Hellman based public keys, with which the strong security is ensured based on the difficulty of solving discrete logarithm problem [8], are transmitted through wireless channels. In the DiHam, Diffie-Hellman style keys are widely used, e.g., the generation of the CSK, AK and TEK. These keys provide a very secure method to protect the communication system, but the cost of key calculation is high.

Others important algorithms employed in HaKMA and PKMv2 are HMAC, CMAC and Dot16KDF. But since those algorithms perform fast operations, their costs which are very smaller than those of exponential operations can be ignored. Table 2 summarizes the costs of the evaluated schemes. We can see that the cost of the HaKMA operations is between those of PKMv2 with EAP-AKA method and DiHam with level-1 TEK.

Table 2. Lists of modular operations for different security schemes.

Security scheme	Exponential operations			
	CSK	EAP	TEK	Total
DiHam with level-1 TEK	7	-	1	8
DiHam with level-2 TEK	7	-	6	13
DiHam with level-3 TEK	7	-	76	83
PKMv2 with EAP-AKA	-	2	-	2
HaKMA with EAP-AKA	5	2	-	7

5 Conclusions and Future Work

In this paper, the HaKMA security scheme which provides fast and secure key generation process, mutual authentication and EAP based user authentication is proposed. The three-layer architecture simplifies key generation flows compared to those proposed in the DiHam and PKMv2. It further provides a fast and secure key renew process for handover. We also introduce two levels of handover processes to minimize SDT, give connections between MS and BS forward and backward secrecy and analyze the HaKMA's security and performance. From which we can conclude that the HaKMA provides low-cost and effective handover, and its authentication approach is more secure than those of the DiHam and PKMv2.

In the future, we would like to enhance the HaKMA by developing its error handling capability. In the handover support, we will design a flexible MS keys' routing scheme to deliver keys between/among Authenticators, and develop behavior and reliability models so that users can predict the HaKMA's behavior and reliability before using it. The handover authentication between two heterogeneous networks, such as IEEE 802.11 or 3GPP LTE, will also be developed. Those constitute our future research.

References

1. Johnston, D., Walker, J.: Overview of IEEE 802.16 security. IEEE Security & Privacy 2, 40-48 (2004)
2. WiMAX Forum Network Architecture. Stage 2: Architecture Tenets, Reference Model and Reference Points - Part 2, pp. 167. WiMAX Forum (2009)
3. Leu, F.Y., Huang, Y.F., Chiu, C.H.: Improving Security Levels of IEEE802.16e Authentication by Involving Diffie-Hellman PKDS. In: Conference Improving Security Levels of IEEE802.16e Authentication by Involving Diffie-Hellman PKDS, pp. 391-397. (2010)
4. Ergen, M.: Mobile broadband including WiMAX and LTE. Springer Science+Business Media, LLC, Boston, MA (2009)
5. Bernardos, C.J., Gramaglia, M., Contreras, L.M., Calderon, M., Soto, I.: Network-based Localized IP mobility Management: Proxy Mobile IPv6 and Current Trends in Standardization. Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications 1, 16-35 (2010)
6. Yan, Z., Zhou, H., You, I.: N-NEMO: A Comprehensive Network Mobility Solution in Proxy Mobile IPv6 Network. Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications 1, 52-70 (2010)
7. Arkko, J., Haverinen, H.: Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA). RFC. Internet Engineering Task Force: Network Working Group (2006)
8. Elgamal, T.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. Ieee T Inform Theory 31, 469-472 (1985)