Journal of Universal Computer Science, vol. 17, no. 6 (2011), 891-911
submitted: 15/5/10, accepted: 30/11/10, appeared: 28/3/11 © *J.UCS*

Improving Security Levels of IEEE802.16e Authentication by Involving Diffie-Hellman PKDS

Yi-Li Huang
(Department of Computer Science, TungHai University, Taichung, Taiwan
yifung@mail.tbcnet.net)

Fang-Yie Leu
(Department of Computer Science, TungHai University, Taichung, Taiwan
leufy@thu.edu.tw)

Chao-Hong Chiu
(Department of Computer Science, TungHai University, Taichung, Taiwan
g97357015@thu.edu.tw)

I-Long Lin
(Department of Information Management, Central Police University, Taipei, Taiwan
paul@mail.cpu.edu.tw)

Abstract: Recently, IEEE 802.16 Worldwide Interoperability for Microwave Access (WiMAX for short) has provided us with low-cost, high efficiency and high bandwidth network services. However, as with the WiFi, the radio wave transmission also makes the WiMAX face the wireless transmission security problem. To solve this problem, the IEEE802.16Std during its development stage defines the Privacy Key Management (PKM for short) authentication process which offers a one-way authentication. However, using a one-way authentication, an SS may connect to a fake BS. Mutual authentication, like that developed for PKMv2, can avoid this problem. Therefore, in this paper, we propose an authentication key management approach, called Diffie-Hellman-PKDS-based authentication method (DiHam for short), which employs a secret door asymmetric one-way function, Public Key Distribution System (PKDS for short), to improve current security level of facility authentication between WiMAX's BS and SS. We further integrate the PKMv1 and the DiHam into a system, called PKM-DiHam (P-DiHam for short), in which the PKMv1 acts as the authentication process, and the DiHam is responsible for key management and delivery. By transmitting securely protected and well-defined parameters for SS and BS, the two stations can mutually authenticate each other. Messages including those conveying user data and authentication parameters can be then more securely delivered.

Keywords: Diffie-Hellman PKDS, Common secret key, PKMv1, WiMAX security, IEEE802.16e data security
Categories: C.2.3, K.6.5, H.4.3

1 Introduction

In a wireless network, what the users need are generally greater bandwidth, speedy transmission, uninterrupted services and more secure environment. Although WiMAX has farther transmission distance and faster speed than those of IEEE802.11

[IEEE, 04], due to using radio signals to transmit data, it is now facing wireless security issues. In fact, its security is fragile as that of Wi-Fi. So, the IEEE802.16 standard [Barbeau, 05; Johnston, 04] developed a security mechanism, called Privacy Key Management version 1 (PKMv1), which mainly manages keys and defines particular confidential and unidirectional authentication for message delivery. The IEEE802.16e [IEEE, 05; IEEE, 06] owing to performing mobile authentication has practiced 802.16 key management (i.e., PKMv1) [Johnston, 04] and set up PKMv2. In PKMv1, only BS authenticates SS so an SS has the possibility to connect to a fake BS. In fact, mutual authentication can solve this problem. However, the mutual authentication mechanism developed for PKMv2 has its own drawbacks, which will be described later.

Rahman and Kowsar [Rahman, 09] established a one-time authentication key, which can be employed only once to avoid the case that the key once is compromised, the entire system will be in danger. In addition, BS and SS share a common key, and recognize the legitimacy of each other through the key. However, if hackers crack the encryption functions via a reverse engineering process, this scheme will fail to protect the wireless system.

Han et al. [Han, 08] implemented an one-time public key. The system security is basically constructed on this unique key. To prevent the system from a man-in-the-middle attack [Arkko, 04], the authors assumed that there was a one-way function that generates an identifying code. However, they did not describe how the identifying code is generated. So, it is hard for us to evaluate the system security level.

In this paper, we propose a key management approach, called Diffie-Hellman-PKDS-based authentication method (DiHam for short) which manages security keys delivered between SS and BS by involving Diffie-Hellman's public key distribution system (DH-PKDS for short) [Bhattacharya, 05] to provide mutual authentication. We further integrate the PKMv1 and the DiHam as a new authentication method, called PKM-DiHam (P-DiHam for short), in which PKMv1 acts as the authentication process, and the DiHam is responsible for the key management and delivery. With the P-DiHam, BS and SS individually generate the common key used to encrypt messages without the involvement of certificate authority (CA) so the security level of the integration system is higher that of the PKMv1. The preliminary version of this paper is published in [Leu, 10a]. We enhance the paper by increasing the clear description of all items conveyed on messages for key exchange and more detailed security analyses, and give a formal theorem to present the P-DiHam's authentication security.

The contributions of this study are as follows.

(1) We design a high security-level encryption algorithm which can more securely protect encrypted data from being cracked by hackers.
(2) We develop a high security authentication key exchange mechanism with which SS and BS can more securely exchange security parameters. From the parameters, the key used to encrypt data messages is then more securely derived and produced, and consequently cannot be easily cracked.

The rest of this paper is organized as follows. Section 2 describes background and related work of this study. Section 3 presents the PKMv1 model. Section introduces the new method and describes how it provides a more secure and convenient

environment than what PKMv1 can. Simulation and discussions are presented in section 5. Section 6 concludes this paper and outlines areas of future research.

2 Background and Related work

2.1 WiMAX Initiation

The process from when an SS joins a WiMAX network to the time when SS and BS establish a service connection has 10 steps [IEEE, 04], in which the first four steps that should be performed before BS can start authenticating SS and exchanging security keys with SS are as follows.

a) Scanning BS's downlink channels: If several channels are available, SS selects one and performs appropriate actions to connect to and synchronize with BS's downlink.
b) Acquiring uplink parameters: After synchronization, SS receives UL_MAP, Downlink Channel Descriptor (DCD for short), Uplink Channel Descriptor (UCD for short) from BS, and catches the entire channel configuration and settings.
c) Performing ranging: SS scans UL_MAP to acquire the frequency band of the ranging sub-channel, through which SS issues a RNG_REQ message containing its' MAC address as the source MAC address to request ranging parameters. BS based on the MAC address assigns a channel ID (CID), and sends the related parameters to SS. With the parameters, SS adjusts its uplink power and frequency.
d) Negotiating basic capabilities: On receiving the CID, SS exchanges information with BS, telling BS what capabilities that it has.

In the fifth step, BS authenticates SS and exchanges keys with SS by using PKMv1. In this study, as stated above the original PKM process is modified and integrated with the DH-PKDS. The remaining five steps of PKMv1 include performing registration, establishing IP connectivity, calibrating time and days, transferring operational parameters, and setting up connections.

2.2 Diffie-Hellman PKDS

In 1976, Diffie and Hellman [Diffie, 76] proposed the public key distribution system (PKDS for short) which as a specific public key system allows two people to exchange keys without knowing each other's identity. Basically, a single key encryption can truly protect messages from their contexts being known to hackers. However, once the key, very often a fixed-length key, is solved by hackers, they will realize what the messages are. So, a hard to be solved encryption function is required. An exponential function [Elgamal, 85] is a typical example. Thus, a specific exponential function is employed to encrypt private keys for DH-PKDS.

In a DH-PKDS system, to establish a private communication connection between two parties, e.g., A and B shown in Figure 1, A and B first generate two integers: p and g, where p is a big prime, and g is the primitive root of p. Next,

1). Party A randomly selects a large number X_a as its private key [Aboud, 04] [Bien, 10][Lee, 10] which has the same number of bits as *p*, and defines a public key Y_a where $Y_a = g^{X_a} \bmod p$.

2). Party B randomly chooses a large number X_b as its private key which has the same number of bits as *p*, and defines a public key Y_b where $Y_b = g^{X_b} \bmod p$.

3). Party A sends Y_a to party B without telling party B its private key X_a.

4). Party B sends Y_b to party A without telling party A its private key X_b.

5). Party A computes a secrete key K_a, $K_a = Y_b^{X_a} \bmod p = \left(g^{X_b}\right)^{X_a} \bmod p = g^{X_a X_b} \bmod p$.

6). Party B computes a secrete key K_b, $K_b = Y_a^{X_b} \bmod p = \left(g^{X_a}\right)^{X_b} \bmod p = g^{X_a X_b} \bmod p$, i.e., $K_a = K_b$, showing that the two parties share the same secret key, called common secret key K, where $K = g^{X_a X_b} \bmod p$.

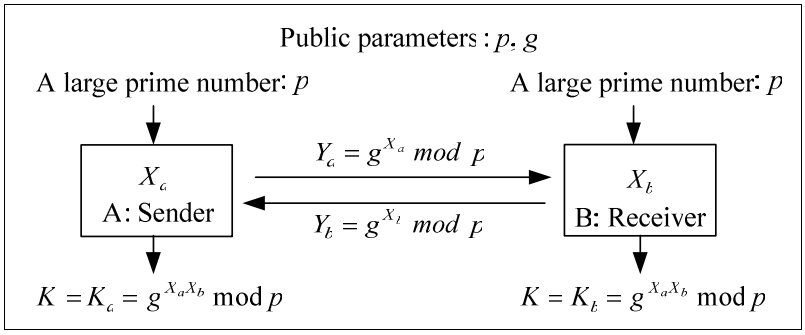

Figure 1: The generation of a common secrete key by the DH-PKDS

In other words, when the DH-PKDS is in use, the two sides of a communication channel can exchange security keys without knowing each other's identity. However, this is also the disadvantage of the DH-PKDS because without user certificates, hackers may issue a man-in-the-middle attack [Arkko, 04].

2.3 Related Work

Rahman and Kowsar [Rahman, 09] used Diffie-Hellman algorithm to establish a one-time authentication key and an exclusive-or function to encrypt messages, and assumed that each legitimate BS and SS has an International Subscriber Station Identity (ISSI for short) authentication ID and a corresponding cryptographic function. The security process is as follows.

SS sends a message to BS to allege that it is a legitimate subscriber. BS sends a random number R_{BS} to challenge SS. SS invokes the cryptographic function to

calculate the value for this random number R_{BS}, and sends the value and its ISSI number to BS. SS further transmits a random number R_{SS} to challenge BS. BS also invokes the cryptographic function corresponding to the ISSI to calculate the value for this random number R_{SS}, and sends the value to SS. Only the legitimate BS and SS know what function corresponding to the ISSI is. The key shared by SS and BS for message encryption is then established. This approach is secure since both R_{BS} and R_{SS} are not delivered through wireless channels, and the cryptographic function and an ISSI are employed to prevent the man-in-the-middle attack. However, once the key is solved by hackers, this approach will lose its protection capability.

Han et al. [Han, 08] used the Diffie-Hellman algorithm to generate a common key PK and proposed a one-way hash function H (TSSI) where TSSI stands for Temporary Subscriber Station Identity. At first, SS sends a message to BS to allege that it is a legitimate subscriber. BS sends a random number R_{BS} to challenge SS. SS calculates H(TSSI) with its own ISSI, cascades H(TSSI), R_{BS} and its public key PK_{SS} to generate the response, H(H(TSSI)|| R_{BS} || PK_{SS}), and sends the response, PK_{SS} and R_{SS} to challenge BS. BS calculates a hash value by involving H(TSSI), R_{BS} and PK_{SS} that it stored beforehand, and compares the calculation result with SS's response to check to see whether SS are legitimate. BS further calculates H(H(TSSI)|| R_{SS} || PK_{BS}), and sends the result and its own public key PK_{BS} to SS. SS checks BS's identity by using the response that it receives. The common key *PK* shared by SS and BS to encrypt messages is then individually generated at both sides. But the authors as stated above did not specify how to generate TSSI from ISSI, and how SS and BS know each other's ISSI. Semantic based Retrieval using Meta data

3 PKMv1 Model

In PKMv1, two important keys, authentication key (*AK*) and traffic encryption key (*TEK*) both generated by BS by invoking the RSA algorithm [Aboud, 04; Aboud, 08; RSA], are involved in PKMv1's authentication process. *AK* as a random number is produced for authorization, and *TEK* as a key derived from *AK* is generated to encrypt transferred data.

3.1 PKMv1 Process

Figure 2 shows the PKMv1 process in which five steps should be performed before *TEK* can be used to encrypt data messages [Sun, 07].

Step1: SS begins its authentication by sending an authentication-information message (i.e., message 1), which contains SS manufacturer's X.509 certificate, to BS. BS can authenticate the certificate, or just based on Connectivity Service Network (CSN) management policies [Barbeau, 05] ignore this message. Figure 3 shows the

message format. The PKM code, as listed in Table1, is used to identify the type of the PKM message. When a message with an invalid code is received, it will be discarded. The Cert(SS Manufacturer) attribute contains an X.509 CA certificate used to identify the Certificate Authority (CA) that issued the certificate to the manufacturer.

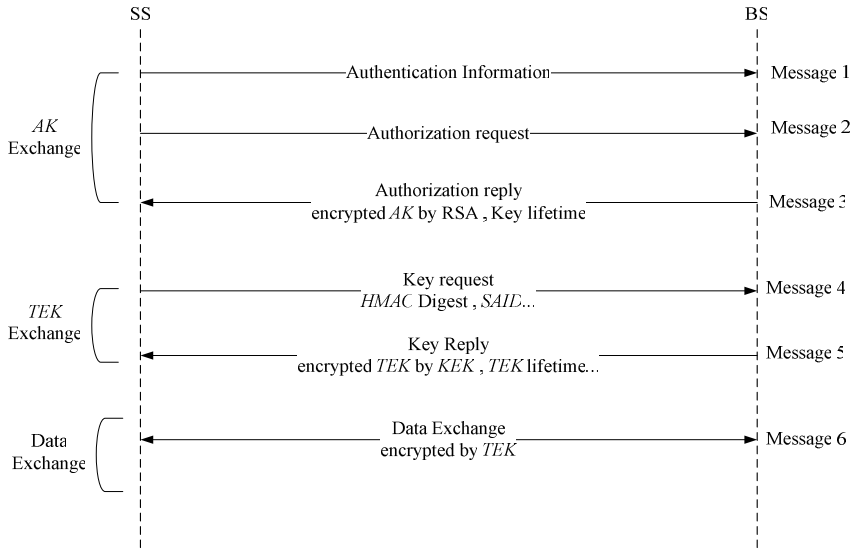

Figure 2: PKMv1 process in PMP mode [Sun, 07]

$$PKM\ Code \mid Cert(SS.Manufacturer)$$

Figure 3: An authentication-information message (message 1)

After sending an authentication-information message to BS, SS immediately delivers an authorization-request message to BS (i.e., message 2). Figure 4 shows the message format in which the *Cert(SS)*, containing an *X.509 SS certificate* issued by SS's manufacturer, is a public key that binds SS's identifying information to its RSA public key in a verifiable manner [IEEE, 04]. The *Security-Capabilities* attribute conveys the data encryption and authentication algorithms that SS supports. The *SAID* attribute contains SS's basic *CID* that BS assigned to SS during the initial ranging step.

PKM code	PKM message type	MAC management message
0-2	Reserved	--
3	SA Add	PKM-RSP
4	Authorization Request	PKM-REQ
5	Authorization Reply	PKM-RSP
6	Authentication Reject	PKM-RSP
7	Key Request	PKM-REQ
8	Key Reply	PKM-RSP
9	Key Reject	PKM-RSP
10	Authentication Invalid	PKM-RSP
11	TEK Invalid	PKM-RSP
12	Authentication Information	PKM-REQ
13-255	Reserved	--

Table 1: PKM message codes

$$PKM\ \ Code \mid Cert(SS) \mid Security - Capabiliti\ es \mid SAID$$

Figure 4: An authorization-request message (message 2)

Step2: On receiving the authorization-request message, BS validates SS's certificate, chooses an encryption algorithm and a protocol specified in the *Security-Capabilities* attribute, generates *AK*s from which one is chosen, encrypts the chosen *AK* with SS's public key and then sends the *AK* back to SS in an authorization-reply message (i.e., message 3), which as shown in Figure 5 also contains the *AK*'s lifetime, a 4bits *AK* sequence number used to distinguish the *AK* from other *AK*s, and SS's *SA-Descriptor*. The *SA-Descriptor* lists descriptors of Static SAIDs that SS is authorized to access. SS on receiving message 3 decrypts the *AK* by using the RSA algorithm and its own private key.

$$PKM\ \ Code \mid AK \mid Key - lifetime \mid Key - sequence - Number \mid SA - Descriptor$$

Figure 5: An authorization-reply message (message 3)

Step3: Each time when SS would like to transfer data to BS, it sends a key-request message (i.e., message 4) to BS. This message as shown in Figure 6 contains a 160 bits *HMAC digest* derived from the *AK* for downlink authentication.

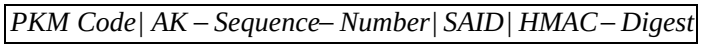

Figure 6: A key-request message (message 4)

Step4: BS on receiving message 4 validates the value of *HMAC-Digest* by invoking the HMAC-Digest algorithm. After the validation, BS generates a *TEK* based on the *AK* and the selected encryption algorithm. It further encrypts the *TEK* by *KEK* and then sends the *TEK* to SS through a key-reply message (i.e., message 5), where the *KEK* is derived from the *AK*. The message also includes an old *TEK* and a new *TEK*. When the old expires, the new one will be used. Both are encrypted by the *KEK*. Figure 7 shows the message.

Figure 7: A key-reply message (message 5)

Step5: SS on receiving message 5 validates the received *HMAC-Digest* value. After the validation, SS decrypts the value by using *KEK*, and recovers the *TEK*s. Data messages are then encrypted by the *TEK* before they are delivered between SS and BS.

3.2 IEEE802.16 Security Analyses

In PKMv1, only BS authenticates SS, and the authentication does not testify the completeness of messages. For example, a fake BS when receiving an authorization-request message replies SS with a PKM code= 10, i.e., an authentication-invalid message, indicating that SS cannot connect to a valid BS. The key reason is that SS does not authenticate BS.

Basically, *AK* is a random number. Its random number generator should be trustable. Otherwise, the hackers can then break through the system. Also, the maximum lifetime of an *AK* is 70 days. If *AK* is updated every 30 minutes which is the shortest lifetime of an *AK*, during its maximum lifetime, up to $3360(=\frac{70*24}{0.5})$ *TEK*s can be gathered to decode messages. WiMAX employs the RSA algorithm to protect *AK*. However, the RSA-768 algorithm has been cracked [Kleinjung, 10]. In fact, the RSA factoring algorithm is rather complex. It is difficult to crack the RSA-1024 [Kleinjung, 10].

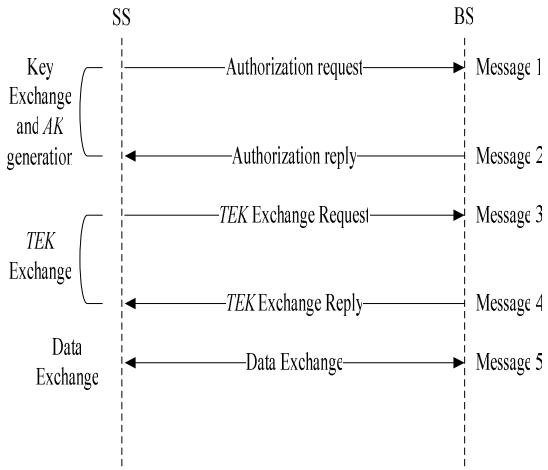

Figure 8: P-DiHam process

4 The Proposed Approach

In this study, the proposed *TEK* exchange method is different from the original PKMv1 standard in that in the PKMv1 *TEK* is encrypted by *KEK* (indirectly by *AK*) and delivered through a wireless channel. However, in the P-DiHam, only encrypted *pre_TEK* is sent via a wireless channel. In the following, we will describe how the PKDS is applied to WiMAX authentication. Figure 8 illustrates the P-DiHam process in which messages 1 and 2 are for *AK* generation, messages 3 and 4 are used to exchange *TEK* and message 5 is for data exchange.

4.1 Parameters, Functions and OP_Codes

The parameters defined by the P-DiHam are as follows.

P : a strong prime number.

g : the primitive root of P.

$RSi, i = 1,2,3$: Private keys generated by SS.

$RBi, i = 1,2,3$: Private keys generated by BS.

$P_{RSi}, i = 1,2,3$: SS's Public keys.

$P_{RBi}, i = 1,2,3$: BS's Public keys.

$CSKi, i = 1,2,3$: Common secret keys.

$pre\_AKi, i = 1,2,3$: Pre-authentication keys.

$pre\_TEKi, 1 \le i \le 15$: Pre-traffic encryption keys.

The functions defined and used include

Encryption function_1: $Encrypt\_1(x, y) = g^{x+y} \bmod P$

Encryption function_2: $Encrypt\_2(x, y) = x \oplus y$

Several MAC management messages which all begin with an Operation Code (*OP_Code* for short) field are also defined. Their descriptions are listed in Table 2.

OP_Code	**P-DiHam message**
1	Authorization Request
2	Authorization Reply
3	Authentication Invalid
4	TEK Exchange Request
5	TEK Exchange Reply
6	TEK Exchange Invalid
0, 7-255	Reserved

Table 2: Operation Codes and their descriptions

4.2 Applying PKDS to WiMAX Authentication

When SS would like to connect to BS, the process is as follows.

Step 1: SS first produces three random numbers, *RS1*, *RS2*, and *RS3*, as private keys, and three public keys P_{RS1}, P_{RS2} and P_{RS3} where $P_{RSi} = g^{RSi} \bmod P, 1 \le i \le 3$. After that, it sends an authorization-request message (i.e., message 1 shown in Figure 8) to BS. Figure 9 illustrates the format in which *OP_Code* =1.

$$OP\_Code \mid Cert(SS.Manufacturer) \mid Cert(SS) \mid P_{RS1} \mid P_{RS2} \mid P_{RS3}$$

Figure 9: An authorization-request message (message 1 from SS to BS) with OP_Code=1.

The *Cert(SS.Manufacture)* and the *Cert(SS)*, like those in PKMv1, contain *X.509 digital certificates* as the facility certificates to respectively identify SS's device manufacturer and SS; P_{RS1}, P_{RS2} and P_{RS3} are sent to BS to produce three common secret keys shared by SS and BS. The delivery of the three public keys can establish the minimum security requirements between SS and BS. If only one public key is sent to BS, which of course generates only one common key, the security level of the system would be lower.

Step 2: On receiving the authorization-request message, BS produces three private keys *RB1*, *RB2* and *RB3*, with which BS generates three public keys P_{RB1}, P_{RB2} and P_{RB3} where $P_{RBi} = g^{RBi} \bmod P, 1 \le i \le 3$. BS then yields three common secret keys *CSK1*, *CSK2* and *CSK3* where $CSKi = P_{RSi}^{RBi} \bmod P = g^{RSi*RBi} \bmod P, 1 \le i \le 3$. Then BS validates SS to see whether or not it is a legitimate user of the system by checking user certificate *Cert(SS)*. If not, BS sends an authentication-invalid message, of which

the format is shown in Figure 10a, with *OP_Code*=3 to SS. Otherwise, BS retrieves *PubKey(SS)* from *Cert(SS)* and invokes its random number generator to produce three *pre_AK*s (i.e., *pre_AK*1, *pre_AK*2 and *pre_AK*3), with which five *AK*s, *AK*1 ~ *AK*5, are generated by invoking HMAC-SHA algorithm where

$$AK1 = HMAC\text{-}SHA(CSK1, pre\_AK1 \mid CSK2 \mid pre\_AK2 \mid SS\_MAC\_Addr \mid BS\_MAC\_Addr)$$

,

$$AK2 = HMAC\text{-}SHA(CSK2, pre\_AK2 \mid CSK3 \mid pre\_AK3 \mid SS\_MAC\_Addr \mid BS\_MAC\_Addr)$$

,

$$AK3 = HMAC\text{-}SHA(CSK3, pre\_AK3 \mid CSK1 \mid pre\_AK1 \mid SS\_MAC\_Addr \mid BS\_MAC\_Addr),$$

$$AK4 = HMAC-SHA(CSK1, CSK2 \mid CSK3 \mid pre\_AK1 \mid pre\_AK2 \mid SS\_MAC\_Addr)$$

,

and

$$AK5 = HMAC-SHA(CSK2, CSK3 \mid CSK1 \mid pre\_AK2 \mid pre\_AK3 \mid BS\_MAC\_Addr)$$

.

After that, it encrypts *pre-AKi* with *CSKi* by using $Encrypt\_2(CSKi, pre\_AKi), i = 1,2,3$, for further authentication. At last, it delivers an authorization-reply message (i.e., message 2), of which the format is shown in Figure 10b and *OP_Code*=2, to SS. P_{RB1}, P_{RB2} and P_{RB3} are included since SS needs them to generate common secret keys *CSK1*, *CSK2* and *CSK3*. *Encrypt_1* (*CSK1, PubKey(SS)*) is used by SS to authenticate BS. *Encrypt_2*(*CKSj, pre_AKj*), called a carrier, carry the security parameters *pre_AK1*, *pre_AK2* and *pre_AK3* to SS , 1≤j≤3.

Basically, the *AK* generation formulas follow the ones used by the IEEE 802.16 PKM, but we increase the number of parameters to improve the system security. A total of five *AK*s are generated to support the production of the following 75 *TEK*s.

$OP\_Code \mid P_{RB1} \mid P_{RB2} \mid Encrypt\_1(CSK1, CSK2)$

(a) An authentication-invalid message

$OP\_Code \mid P_{RB1} \mid P_{RB2} \mid P_{RB3} \mid Encrypt\_1(CSK1, PubKey(SS)) \mid Encrypt\_2(CSK1, pre\_AK1) \mid$ $Encrypt\_2(CSK2, pre\_AK2) \mid Encrypt\_2(CSK3, pre\_AK3)$

(b) An authorization-reply message

Figure 10: An authentication-invalid message with OP_Code=3 (message 2 from BS to SS) and an authorization-reply message (message 2 from BS to SS) with OP_Code=2.

Step 3: SS on receipt of message 3 retrieves P_{RB1} and P_{RB2} from the message to respectively derive *CSK1* and *CSK2* and calculate *Encrypt*_1(*CSK*1,*CSK*2). If the

calculation result is not equal to the *Encrypt*_1(*CSK*1,*CSK*2) value retrieved from the message, implying the BS is a fake one, it discards the message and waits for a valid BS's reply.

SS on receiving message 2 retrieves P_{RB1}, P_{RB2} and P_{RB3} from the message to generate its common secret keys, $CSKi = P_{RBi}^{RSi} \bmod P = g^{RBi*RSi} \bmod P, 1 \le i \le 3$. SS determines whether *Encrypt_1(CSK1,PubKey(SS))*s sent by BS and calculated inside in SS are equal or not? If not, then SS discards the fake message and waits for a valid BS's reply. Otherwise SS acquires the three *pre-AKs* by invoking $Encrypt\_2(CSKi, pre\_AKi), i = 1,2,3$. At last, SS generates *AK*1 ~ *AK*5 by using the same HMAC-SHA algorithm.

After that, SS sends a *TEK*-exchange-request message (i.e., message 3) with *OP_Code*=4 to BS before data transmission. In this study, we divide the *TEK*s into three levels based on SS's computation ability and the required communication security levels.

1) Level-1 *TEK*: SS sends a level-1 *TEK*-exchange-request message (i.e., level-1 message 3), of which the format is shown in Figure 11 with *Security-capabilities*= 1, telling BS that SS requests a level-1 security. The role of *Encrypt_1*(*CSK1*, *pre_AK1*) as mentioned above is to request BS to generate a *TEK*.

$$OP\_Code \mid Encrypt\_1(CSK1, pre\_AK1) \mid Security-capabilities$$

Figure 11: A level-1 TEK-exchange-request message (i.e., level-1 message 3 from SS to BS) with OP_Code=4 and Security-capabilities=1.

2) Level-2 *TEK*: SS generates a random number as a *pre_TEK*, and then produces five *TEK*s where $TEKi = g^{AKi+pre\_TEK} \bmod P, 1 \le i \le 5$. SS encrypts the pre_TEK by invoking $Encrypt\_2(CSK2, pre\_TEK)$ and sends a level-2 *TEK*-exchange-request message (i.e., level-2 message 3) to BS. The message format is shown in Figure 12 in which *OP_Code*=4 and *Security-capabilities* = 2, indicating that SS requests a Level-2 security. The roles of *Encrypt_1*() and *Encrypt_2*() are individually mentioned above.

$$OP\_Code \mid Encrypt\_1(CSK1, pre\_AK1) \mid Encrypt\_2(CSK2, pre\_TEK) \mid Security-capabilities$$

Figure 12: A level-2 TEK-exchange-request message (i.e., level-2 message 3 from SS to BS) with OP_Code=4 and Security-capabilities=2.

3) Level-3 *TEK*: SS produces 15 pre_TEKs, and then yields 75 *TEKs* where

$pre\_TEK_{(i-1)\times 5+j} = CSKi \oplus AKj, 1 \le i \le 3; 1 \le j \le 5;$

$TEK_{(i-1)\times 15+j} = g^{AKi+pre\_TEK_j} \bmod P, 1 \le i \le 5, 1 \le j \le 15.$

It consequently sends a level-3 *TEK*-exchange-request message (i.e., level-3 message 3) in which *OP_Code*=4 and *Security-capabilities* = 3 to BS. The message format is illustrated in Figure 13.

$$OP\_Code \mid Encrypt\_1(CSK1, pre\_AK1) \mid Security-capabilities$$

Figure 13: A level-3 TEK-exchange-request message (i.e., level-3 message 3 from SS to BS) with OP_Code=4 and Security-capabilities=3.

Step 4: BS on receiving a *TEK*-exchange-request message checks the authentication code $Encrypt\_1(CSK1, pre\_AK1)$ contained in the message to see whether SS is legal or not. If yes, BS checks the *security-capabilities* and generates the corresponding *TEK* or *TEK*s to synchronize the following data transmission with SS.

The processes for BS to generate different security-level *TEK*s are as follows.

1) Level-1 *TEK*: To respond to the level-1 *TEK*-exchange-request message, BS randomly generates a *TEK*, encrypts the *TEK* by using $Encrypt\_2(CSK, TEK)$ and delivers a level-1 *TEK*-exchange-reply message (i.e., level-1 message 4) with *OP_Code*=5 to SS. The format is shown in Figure 14. *Encrypt_1*(*CSK2, pre_AK2*) is the third authentication key (besides *Cert(SS)* and *Encrypt_1*(*CSK1, pre_AK1*)) between SS and BS. The two *Encrypt_2*()s are used to carry *old_TEK* and *new_TEK* to SS.

$$OP\_Code \mid Encrypt\_1(CSK2, pre\_AK2) \mid Encrypt\_2(CSK2, old\_TEK) \mid Encrypt\_2(CSK3, new\_TEK)$$

Figure 14: A level-1 TEK-exchange-reply message (i.e., a level-1 message 4 from BS to SS) with OP_Code=5.

2) Level-2 *TEK*: To respond to the level-2 *TEK*-exchange-request message, BS decrypts the *pre_TEK* and generates the same five *TEK*s also by using the formulas $TEKi = g^{AKi + pre\_TEK} \bmod P,\ 1 \le i \le 5$. Next, it chooses one of the *TEK*s and sends a level-2 *TEK*-exchange-reply message (i.e., level-2 message 4) with *OP_Code*=5 to SS. The message format is shown in Figure 15 in which the *TEK* sequence number (a number between 1~5) is to tell SS which *TEK* is chosen.

$$OP\_Code \mid Encrypt\_1(CSK2, pre\_AK2) \mid TEK\ seq\ num$$

Figure 15: A level-2 TEK-exchange-reply message (i.e., a level-2 message 4 from BS to SS) with OP_Code=5.

3) Level-3 *TEK*: To respond to the level-3 *TEK*-exchange-reply message, BS generates 75 *TEK*s by using the following formulas.

$$pre\_TEK_{(i-1)\times 5+j} = CSKi \oplus AKj, 1 \le i \le 3; 1 \le j \le 5;$$
$$TEK_{(i-1)\times 15+j} = g^{AKi + pre\_TEKj} \bmod P,\ 1 \le i \le 5, 1 \le j \le 15 .$$

After that, BS chooses one of the *TEK*s, and sends a level-3 *TEK*-exchange-reply message (i.e., level-3 message 4) which contains the *TEK* sequence number (a number between 1~75) to SS. The message format is shown in Figure 16.

$$\boxed{OP\_Code \mid Encrypt\_1(CSK2, pre\_AK2) \mid TEK\ \ seq\ num}$$

Figure 16: A level-3 TEK-exchange-reply message (i.e., level-3 message 4 from BS to SS) with OP_Code=5.

Step 5: After finishing one of the level-*i TEK* procedures, *i*=1,2,3, both SS and BS can now use the *TEK* to encrypt data messages (i.e., message 5s).

4.3 TEK Security Analyses

When the user of SS is performing a low-secret activity, such as surfing the web pages, he/she can choose a level-1 *TEK* generated by BS. SS only needs to decrypt the *TEK*. This gives the least burden to SS's hardware, but, the security level is lower than those of the other two since the *TEK* is delivered through a wireless channel, even though the key is encrypted. Hence, it is relatively easier to be cracked.

When the user would like to perform a middle-level secret activity, such as communicating with other SS or receiving e-mails, he/she can choose a level-2 *TEK*. In this level, SS generates a random number as a *pre-TEK*, and calculates *TEK*s. The hardware consumption cost is then higher than that of a level-1 *TEK*. But a level-2 *TEK* is more secure because SS transfers the *pre-TEK* instead of the chosen *TEK* to BS, and BS only sends a *TEK* sequence number back to SS. Even both the *pre-TEK* and the sequence number are known to hackers, without the *AK*s and *CSK*s the hackers cannot obtain the chosen *TEK*.

While the user is doing something that requires high-level security, such as e-commerce or secret file transferring, SS can use a level-3 TEK. In this level, SS and BS employ *CSK*s, *AK*s and *pre_AK*s to individually produce a set of *TEK*s. To achieve their synchronization, BS sends a level-3 *TEK*-exchange-reply message to notify SS which *TEK* is chosen. In this case, SS needs to invoke several algorithms so the hardware burden is relatively higher. But hackers cannot directly retrieve any keys from intercepted packets. All are indirect information so that the security level is relatively higher. Besides, due to involving finite number of attributes, the number of generated *TEK*s is finite. So if level-3 connections are frequently established, it is possible that *TEK*s are used repeatedly. Such will slightly lower a level-3 *TEK*'s security level. Table 3 summarizes the characteristics of the three levels.

Item / TEK	**Perform.**	**Hardware**	**Security**
Level 1	High	Low	Low
Level 2	Middle	Middle	Middle
Level 3	Low	High	High

Table 3: Summary of the characteristics of the three levels of TEKs

4.4 The P-DiHam Security Analyses

From security viewpoint, the drawback of a wireless system is transmitting data via wireless channels since data can easily be intercepted, resulting in information leakage. Hackers can even issue pseudo-communication [Deininger, 07] to receive more information. The P-DiHam uses *Encrypt_2* (*X, Y*) to carry and protect the transmitted data, and *Encrypt_1* (*X, Y*) to provide authentication. When hackers intercept *Encrypt_2*(*X, Y*), the probability that they can solve *Y* from *Encrypt_2*(*X, Y*) on one trial is $\frac{1}{2^n}$, where *Y* as mentioned above may be a *pre_AK*, *pre_TEK* or *TEK*.

Theorem 1:
Assume that X and Y are both *n* bits in length, then the probability *p* that we can obtain *Y* on one trial from illegally received *Encrypt_2* (*X, Y*) is $p = \frac{1}{2^n}$.

Proof: Let $X = x_{n-1}...x_2x_1x_0$, $Y = y_{n-1}...y_2y_1y_0$ and $Encrypt\_2(X,Y) = z_{n-1}...z_2z_1z_0$ where x_i, y_i, z_i are binary digits, and $z_i = x_i \oplus y_i$, $0 \le i \le n-1$. If $z_i = 0$, the possible (x_i, y_i) pair is (0,0) or (1,1). Otherwise, the possible (x_i, y_i) pair is (0,1) or (1,0). Hence, when z_i is known, for each *i*, the probability to obtain the correct y_i on one trial is $\frac{1}{2}$, i.e., either 0 or 1. Then the probability to correctly recover original *Y* on one trial is $\frac{1}{2^n}$. QED.

The most effective method to attack the P-DiHam is to get both sides' public keys and then decipher the keys into private keys, i.e., $RS1 \sim RS3$ or $RB1 \sim RB3$. Once either set is successfully recovered, *CSK1*, *CSK2*, *CSK3* and *AK* can then be produced. However, the P-DiHam security solution is based on the difficulty of solving discrete logarithm problem. The time complexity of solving the problem currently known is $O(\exp\sqrt{cm\ln m})$ [Elgamal, 85], where c=0.69 and m is the length of public key. If the length of the public key is 256 bits, the time required is 3.9093×10^{13} s (=1.2 million years). In other words, the P-DiHam is secure and safe.

4.5 Comparison of P-DiHam, IEEE 802.16e PKMv1 and PKMv2

To increase PKMv1's security and mitigate its shortcomings, PKMv2 which includes two stages of authentication is proposed [Scarfone, 09; Johnston, 04]. The first stage employs a PKMv1-compatible RSA device mutual authorization protocol as its

network facility authentication. The second deploys Extensible Authentication Protocol (EAP), which is performed on an Authentication, Authorization, Accounting (AAA) server, to authenticate users and collect users' network usage information. The BS acts as a relay station. After authentication, the master session key (MSK) 512 bits in length is then delivered to the BS to tell the BS whether the underlying SS is allowed to utilize the network facilities or not. The RSA device mutual authorization protocol [Scarfone, 09] is defined as

auth_req: SS -> BS: *SS-Random* | *Cert*(*SS*) | *Capabilites* | *Basic CID*

auth_reply: BS -> *SS: SS-Random* | *BS-Random* | *RSA-OAEP-Encrypt(PubKey(SS), PAK* | *Id(SS))* | *Lifetime* | *SeqNo* | *SAIDList* | *AAID* | *Cert(BS)* | *Sig(BS)*

auth_ack: SS -> BS: *BS-Random* | *SS_MAC_Address* | *OMAC (Auth-Key, BS_Random* | *SS_MAC_Address)*

in which the authorization reply (auth-reply) involves *Cert(BS)* which together with *Cert(SS)* are respectively the issues used by SS and BS to authenticate each other so as to achieve mutual authentication and make PKMv2 more secure than PKMv1. However, hackers may access the BS several times with legal SSs to acquire *Cert(BS)* and related information, which will hugely reduce the security levels between the SS and BS. But in our approach, *Encrypt_1(CSK1,PubKey(SS))* is employed to authenticate BS by SS. Only legal BS and SS have correct *PubKey(SS)* So that only they can establish the *CSK*s, generate accurate *Encrypt_1(CSK1,PubKey(SS))* and then successfully finish the authentication. Therefore, we dare to say the proposed approach is more secure than PKMv2's RSA device mutual authorization protocol.

Furthermore, EAP has four types of authentication versions [WiMAX-Forum] in which EAP-TLS, EAP-TTLS and EAP-AKA respectively employ Certificate Authority (CA), Tunnel and USIM subsystems to support their authentication. Each of the four types needs to work together with AAA server. Therefore, EAP's performance is not better than our approach since to achieve mutual authentication between SS and BS, our approach only requires a round trip communication, i.e., Authorization request and Authorization reply.

5 System Experiments and Discussion

To verify the proposed process to see whether it is feasible in practice or not, we simulate the P-DiHam on NS-2 [NS-2, 09; Neves, 08]. In this study, three experiments were performed. The first measured the times required to process an authorization-request message and an authorization-reply message. The second evaluated the times required to process level-1 to level-3 *TEK*-exchange-request messages. Both experiments were performed on different lengths of *TEK*s, including 256, 512, 768 and 1024 bits. The third studied the total authentication delays for PKMv1, PKMv2 and our approach. Each experiment was executed 50 times. The specifications of the simulation hardware are listed in Table 4.

Component	Specification
CPU	Intel Pentium Dual CPU E2180 2GHz
Ram	2GB
O/S	Windows XP SP2

Table 4: Specifications of the simulation hardware

5.1 Results of the First Experiment

The results of the first experiment are shown in Figures 17 and 18. Figure 17 depicts the fact that longer key lengths cause SS to spend a longer time to process an authorization-request message. P_{RSi}, i=1,2,3, is generated by invoking an discrete-logarithm function. Hence, the lengths and processing time of the corresponding ciphertext increase sharply.

As shown in Figure 18, the time BS requires to generate an authorization-reply message given a 512-bit AK is several times that required when a 256-bit *AK* is given since BS needs to generate three P_{RBi}'s and three CSK_i's, and invoke HMAC-SHA algorithm to produce five *AK*s. The relationships between 1024-bit *AK*s and 768-bit *AK*s and between 768-bit *AK*s and 512-bit *AK*s are similar to that between 512-bit *AK*s and 256-bit *AK*s.

In addition, on receiving an authorization-reply message, SS needs to certify BS by decrypting parameters conveyed on the message to generate the *AK*s. So, the costs shown in Figure 18 are respectively higher than those plotted in Figure 17.

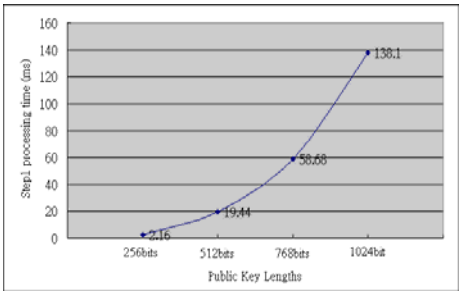

Figure 17: The times consumed by SS to generate an authorization-request message (i.e., step 1).

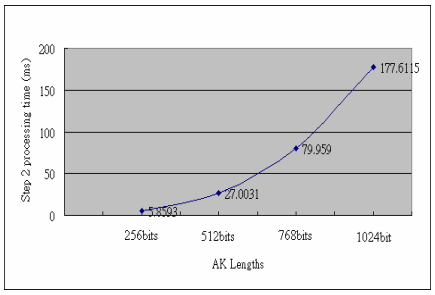

Figure 18: The times consumed by BS from the time point when it receives an authorization-request message to the time point when it generates an authorization-reply message (i.e., step 2).

5.2 Results of the Second Experiment

Figures 19-21 illustrate how different *TEK* security levels and lengths of *TEK*s affect the processing time of the *TEK*-exchange-request and *TEK*-exchange-reply messages. With a level-1 *TEK*, SS only produces an authentication code without producing the

TEK, so the trend of the processing times on different lengths of level-1 *TEK*s as shown in Figure 19 is not very longer than to those of processing an authorization-request message and an authorization-reply message shown in Figures 17 and 18, respectively, since the main tasks of processing the three messages are identifying BS or SS. With a level-2 *TEK*, SS generates a *pre_TEK* by using random number generator and five *TEK*s so the costs are relatively higher than those of invoking a level-1 *TEK*. When a level-3 *TEK* is in use, SS generates 15 *pre_TEK*s and 75 *TEK*s. That is why the costs of the level-3 *TEK* are very much higher than those of a level-2 *TEK*. This meets our description above.

Figure 20 illustrates the times required by BS from the time point when it receives a *TEK*-exchange-request message to the time point when it generates a *TEK*-exchange-reply message. Using a level-3 *TEK* to encrypt data messages is the most time-consuming process because BS also needs to reproduce the 15 *pre_TEK*s and 75 *TEK*s. The costs required are very much higher than those of using a level-1 and a level-2, particularly when the *TEK* is longer.

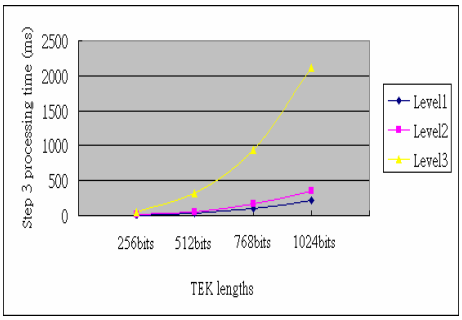

Figure 19: The times consumed by SS from the time point when it receives an authorization-reply message to the time point when it sends out a TEK-exchange-request message (i.e., step 3).

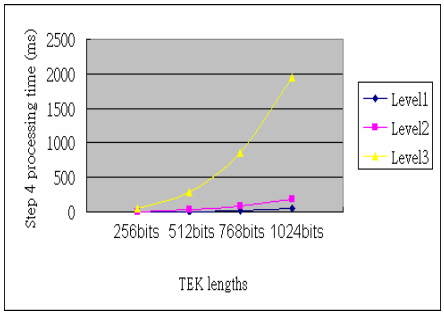

Figure 20: The times consumed by BS from the time point when it receives a TEK-exchange-request message to the time point when it generates a TEK-exchange-reply message (i.e., step 4).

Figure 21 illustrates the fact that SS on receiving a *TEK*-exchange-reply message, no matter a level-1, level-2 or level-3 *TEK* is in use, spends most of its time to identify the legality of BS, i.e., evaluating *Encrypt_1*(*CSK2, pre_AK2*). So, the costs of the three levels of *TEK*s are similar.

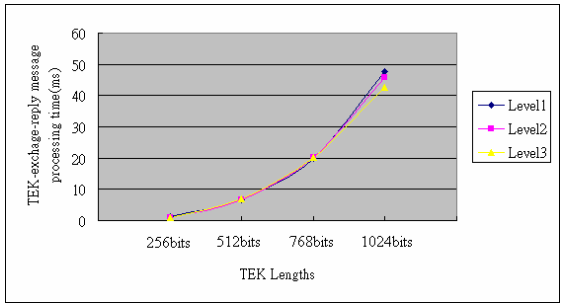

Figure 21: The times required by SS to process a TEK-exchange-reply message after it receives the message.

Scheme		Authentication delay		
		For facility (ms)	For user (ms)	Total delay (ms)
PKMv1		87.4	-	87.4
PKMv2		108.8	135.1	243.9
P-DiHam	Level1	92.3	-	92.3
	Level2	101.5	-	101.5
	Level3	124.6	-	124.6

Table 5: The delays of the compared schemes on the entire authentication process

5.3 Results of the Third Experiment

In the third experiment, PKMv2 authentication was divided into facility authentication and user authentication to meet the real situation. The *AK*s were 160 bits and *TEK*s were 128 bits. Table 5 lists the results in which we can see that PKMv1 is better than P-DiHam since P-DiHam employs DH-PKDS which deploys an exponential function to calculate public keys, i.e., P_{RB1}, P_{RB2}, P_{RB3}, P_{RS1}, P_{RS2} and P_{RS3}, and common secret keys *CSK1*, *CSK2* and *CSK 3*, consequently consuming a longer time. PKMv2 due to conducting mutual authentication requires longer time to perform EAP.

6 Conclusions and Future Research

In this article, we focus on the lift of the security level of the WiMAX authentication, and develop an authentication mechanism to improve WiMAX facility authentication by employing the P-DiHam that integrates the DH-PKDS and the PKMv1, and in which, like that employed by PKMv2, the mutual authentication instead of the unidirectional authentication of PKMv1 is used. When the authentication fails, the

message will be dropped to avoid wireless facilities from being attacked, particularly when a Dos/DDoS attack or a man-in-the-middle attack is launched by a fake BS or SS during the network facility authentication phase. Further, based on the analysis above, P-DiHam's security level is higher than that of PKMv2.

Basically, this study only modifies the PKMv1 and integrates the new one with DH-PKDS. In 802.16e standards, the more secure and well-organized PKMv2 has been released [IEEE, 05]. Our opinion is the DH-PKDS is also applicable to the PKMv2 to further enhance its security. We would also like to derive a reliability model [Leu, 10b] and a behavior model for the integrated system so the users can realize its behavior and reliability before using it. Furthermore, the 802.16e has been added a hand-off mechanism which is also a point that can be easily penetrated by a hijacking attack [Shi, 06]. The DH-PKDS can also be applied to maintain and improve hand-off security. The authors of [Kleinjung, 10] had estimated that within the coming 5 to 10 years, 1024bit RSA encryption system will be cracked. Once the hackers in the *AK* life cycle break the RSA algorithm, they can generate *KEK* to decrypt the *TEK*. Therefore, the effective lifetime of an AK is an important research issue. Those constitute our future research.

References

[Aboud, 04] Aboud, S.: Baghdad Method for Calculating Multiplicative Inverse, In Proc. Int. Conf. on Information Technology, 2004, 816-819.

[Aboud, 08] Aboud, S., AL-Fayoumi, M. A., AL-Fayoumi, M., Jabbar, H.S.: An Efficient RSA Public Key Encryption Scheme, In Proc. Int. Conf. on Information Technology: New Generations, 2008, 127-130.

[Arkko, 04] Arkko, J., Carrara, E., Lindholm, F. Naslund, M., Norrman, K.: Multimedia Internet KEYing (MIKEY), Network Working Group, RFC 3830, August 2004.

[Barbeau, 05] Barbeau, M,: WiMax/802.16 threat analysis, In Proc. ACM Int. Workshop on Quality of Service & Security in Wireless and Mobile Networks, 2005, 8–15.

[Bhattacharya, 05] Bhattacharya, P., Debbabi, M., Otrok, H.: Improving the Diffie-Hellman Secure Key Exchange, In Proc. Int. Conf. on Wireless Networks, Communications and Mobile Computing, 2005, 193-197.

[Bien, 10] Bien, V.Q. Prasad, R.V., Niemegeers, I: Handoff in Radio over Fiber Indoor Networks at 60 GHz, Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, vol. 1, no. 2/3, 2010, 71-82.

[DES] http://en.wikipedia.org/wiki/Data_Encryption_Standard

[Deininger, 07] Deininger, A., Kiyomoto, S., Kurihara, J., Tanaka, T.: Security Vulnerabilities and Solutions in Mobile WiMAX, International Journal of Computer Science and Network Security, vol. 7 no. 11, November 2007, 7-15.

[Diffie, 76] Diffie, W., Hellman, M.E.: New Directions in Cryptography, IEEE Transactions on Information Theory, vol. 22, June, 1976, 638-654.

[Elgamal, 85] Elgamal, T.: A Public Key Cryptosystem and Signature Scheme Based on Discrete Logarithms, IEEE Transactions on Information Theory, vol. 31, July, 1985, 469-472.

[Kleinjung, 10] Kleinjung, T., Aoki, K., Franke, J., Lenstra, A.K., Thomé, E., Bos, J.W. Gaudry,

P., Kruppa, A., Montgomery, P.L., Osvik, D.A., Riele, H.T., Timofeev, A., Zimmermann, P.: Factorization of a 768-bit RSA modulus; February 18 , 2010 http://arstechnica.com/security/news/2010/01/768-bit-rsa-cracked-1024-bit-safe-for-now.ars

[Han, 08] Han, T., Zhang, N., Liu, K., Tang, B., Liu, Y.: Analysis of Mobile WiMAX Security: Vulnerabilities and Solutions, In Proc. Int. Conf. on Mobile, Ad Hoc and Sensor Systems, 2008, 828-833.

[IEEE, 04] IEEE Standard for Local and metropolitan area networks Part 16: Air Interface for Fixed Broadband Wireless Access Systems, IEEE Std 802.16-2004, http://www.ieee802.org/16/, 2004.

[IEEE, 05] IEEE Std 802.16e-2005, http://ieee802.org/16/published.html, 2005.

[IEEE, 06] IEEE Standard for Local and metropolitan area networks Part 16:Air Interface for Fixed and Mobile Broadband Wireless Access Systems Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands and Corrigendum 1, IEEE Std 802.16e-2005 and IEEE Std 802.16-2004 / Cor 1-2005, 2006.

[Johnston, 04] Johnston, D., Walker, J.: Overview of IEEE 802.16 security, IEEE Security &Privacy Magazine, vol. 2, no. 3, May–June 2004, 40–48.

[Johnston, 04] Johnston, D.: Mutual Authorization for PKMv2, IEEE 802.16 Broadband Wireless Access Working Group http://ieee802.org/16

[Lee, 10] Lee, K., Yeuk, H, Choi, Y, Pho, S, You I, Yim, K,: Safe Authentication Protocol for Secure USB Memories, Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, vol. 1, no. 1, 2010, 46-55.

[Leu, 10a] Leu, F.Y., Huang, Y.F., Chiu, C.H.: Improving Security Levels of IEEE802.16e Authentication by Involving Diffie-Hellman PKDS, In Proc. Int. Workshop on Intelligent, Mobile and Internet Services in Ubiquitous Computing, 2010, 391-397.

[Leu, 10b] Leu, F.Y., Yang C.T., Jiang, F.C.: Improving Reliability of a Heterogeneous Grid-based Intrusion Detection Platform using Levels of Redundancies, Future Generation Computer System, no. 26, 2010, 554-568.

[Neves, 08] Neves, P., Fontes, F., Monteiro, J., Sargento, S., Bohnert, T.M.: Quality of Service Differentiation Support In WiMAX Networks, In Proc. Int. Conf. on Telecommunications, 2008, 411-415.

[NS-2, 09] NS2 Notebook: Multi-channel Multi-interface Simulation in NS2 (2.29), http://www.cse.msu.edu/~wangbo1/ns2/nshowto8.html, February 2009.

[Rahman, 09] Rahman, M.S., Kowsar, S.: WiMAX Security Analysis and Enhancement, In Proc. Int. Conf. on Computer and Information Technology, 2009, 679-684.

[RSA] http://en.wikipedia.org/wiki/RSA

[Shi, 06] Shi, D., Tang, C.: A Fast Handoff Scheme Based on Local Authentication In Mobile Network, In Proc. Int. Conf. on ITS Telecommunications, 2006, 1025-1028.

[Sun, 07] Sun, H.M., Chen, S.M., Sou, E.J.: WiMax Security Problem Study, Communication Security Forum, June, 2007 (in Chinese).

[Scarfone, 09] Scarfone, K., Tibbs, C., Sexton, M.: Guide to Security for WiMAX Technologies (Draft), National Institute of Standards and Technology, Sept. 2009. http://csrc.nist.gov/publications/draft/800-127/draft-sp800-127.pdf

[WiMAX-Forum] http://www.wimaxforum.org/