私 立 東 海 大 學 資 訊 工 程 研 究 所

碩 士 論 文

指導教授：呂芳懌 博士

使用 Diffie-Hellman PKDS 提昇 IEEE802.16e 認證之安全層級

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

研究生：邱兆弘

中華民國 九十九 年 六 月

東海大學碩士學位論文考試審定書

東海大學資訊工程學系 研究所

研究生 邱 兆 弘 所提之論文

使用 Diffie-Hellman PKDS 提昇 IEEE802.16e 認證之安全層級

經本委員會審查，符合碩士學位論文標準。

學位考試委員會
召 集 人 _____ 簽章

委 員 楊朝棟

_____

_____

_____

指 導 教 授 _____ 簽章

中華民國 99 年 7 月 9 日

中文摘要

近年來，Worldwide Interoperability for Microwave Access(簡稱 WiMAX)提供了低成本、高效率以及高頻寬的網路服務。但是因為 WiMAX 是透過無線電電波來傳輸資料，因此 WiMAX 的資料安全問題與 WiFi 是相同的。為了解決這個問題，IEEE802.16std 發展時便提出了 Privacy Key Management (簡稱 PKM)單向的認證流程。也因為是採用單向認證，導致 SS 可能會連接到駭客偽造的 BS，使得 SS 暴露在可能被攻擊的風險下。因此採用雙向認證便可以解決這個問題。在本論文中，我們在 Diffie-Hellman 認證機制的基礎上發展出認證金鑰管理機制(簡稱 DiHam)、利用 One-Way Function 以及使用分享公開金鑰來提昇 PKMv1 的安全層級。也使用雙向認證機制來讓 BS 以及 SS 可以確認通訊對象的合法性。這個機制我們稱為 PKM-DiHam(簡稱 P-DiHam)。P-DiHam 雖然提高了安全性，但是基於效能的考量，我們又提出了 Advanced P-DiHam(簡稱 AP-DiHam)機制。AP-DiHam 的安全性是在 P-DiHam 的基礎上繼續發展之外，也使用了更有效率的 Data Carrier Function 來保護夾帶的資料，更利用二維串流加密技術來對傳輸的資料進行加密以及解密。藉此達到高安全、高效率的認證機制。

關鍵字：Diffie-Hellman 公開分享金鑰，共同密鑰，PKMv1，WiMAX 安全機制，IEEE802.16e 資料安全。

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. A two-way authentication can avoid this problem. Therefore, in this article, 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 the Initialization Vector between SS and BS, the two stations can mutually authenticate each other. Messages including those conveying user data and authentication parameters can be then more safely delivered.

Keywords: Diffie-Hellman PKDS, Common secret key, PKMv1, WiMAX security, IEEE802.16e data security.

致謝

首先，我想先感謝我的指導教授 呂芳懌博士，身為在職專班的研究生，白天要上班晚上要上課，長期下來對生理與心理可說是相當疲憊，幾次工作繁忙之際曾起放棄學業的念頭時，就會想到老師說過：『考上研究所不難，難的是要堅持下去完成學業』，真是心有同感。另外，呂老師體諒我們這些在職專班的學生工作繁忙，老師放棄自己的休息時間將輔導論文的時間安排在晚上或是假日。因此呂老師在生活上的關懷以及課業上的敦促是我堅持下去並完成學業的主要推手。呂老師在研究上的堅持、認真與仔細，對未來在面對工作與挑戰時也是我值得學習的榜樣。

接下來我要感謝 黃宜豐老師，黃老師豐富的知識以及源源不絕的點子，常常讓身為學生的我有驚豔的感覺。我們有時會跟不上老師的快速思緒，但遇到問題時黃老師總會停下來耐心的重複講解。黃老師對研究的熱情也是鼓舞我在論文寫作上的助力之一，黃老師常說：『這份 Paper 是我們整個團隊合力的結果』，其實若是沒有黃老師指引一條明路，我也無法順利發展後續的研究與論文。

在看似漫長實則短暫的兩年研究所求學生涯中，呂老師與黃老師兩位老師亦師亦友，對兩位老師已不是短短的『謝謝』兩個字就能完整表達我心中的感謝之意。

同時也要感謝林熙禎老師、楊伏夷老師、連志誠老師與楊朝棟老師在百忙之中擔任我的口試委員，並在口試中對論文提供眾多的寶貴經驗以及建議。

另外我也要感謝一路陪伴上來的同學們，可忠、厚泯、信佑與龍騰，沒有你們在課業上互相提醒與幫忙，我也無法順利的完成學業。回想當時在麥當勞討論課業彷彿是沒多久以前的事，畢業後要在聚首也就沒有那麼容易了。

最後我要感謝我的家人，尤其是我的太太季芬，謝謝妳當我最穩固的靠山，讓我沒有後顧之憂。當論文與時間賽跑時，我常常需要到學校討論與進行實驗，妳一個人照顧調皮又搗蛋的澔澔一定很辛苦吧。這兩年來我的時間大多分配給工作與學校，留給家人的都是短暫片段的時間，對於妳與孩子的體諒讓我的心中滿懷感恩。

Contents

List of Figures

List of Tables

Chapter 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 [1], due to using radio signals to transmit data, it is now facing network security issues. In fact, its security is fragile as that of Wi-Fi. So, the IEEE802.16 standard [2] was drawn up a security mechanism called Privacy Key Management version 1 (PKMv1) which mainly manages keys and defines particular confidential and unidirectional authentication for later message delivery. The IEEE802.16e [3][4] owing to performing mobile authentication has been practiced in the way of 802.16 key management (i.e., PKMv1) and set up PKMv2. In PKMv1, the authentication between SS and BS is not implemented in two way, so an SS has the possibility to connect to a fake BS.

Rahman and Kowsar [5] 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. [6] 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 [7], the authors assumed

that there was a one-way function *H (x)*, that generates an identifying code. However, the authors did not describe how the identifying code is generated. So, it is hard for us to evaluate the system security level.

In this article, 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) to provide two-way authentication. We further integrate the PKMv1 and the P-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 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 work is published in [8]. But because the P-DiHam to perform calculations PKDS require time-consuming. Therefore, we proposed the Advanced P-DiHam (AP-DiHam for short) solution, the AP-DiHam improve the poor efficiency of P-DiHam shortcomings, while maintaining a high-security benefits.

With the AP-DiHam, a wireless communication system has the following characteristics:

1. It has a mutual authentication mechanism between SS and BS. Each authentication packet delivery is proceeded by identity certification.
2. All AKs, TEKs, and NTEKs are individually and independently generated by SS and BS,

without delivering them through a wireless channel.

3. All the parameters transmitted through wireless channels are used only once.
4. Every transmitted packet has an operation code (OP_Code for short) to clearly point out the function of the packet. This can simplify and shorten packet recognition process.
5. The fundamental security keys of the system are built during the first cycle of communication between SS and BS, that is, during the steps of authentication request from SS to BS and authentication replay and key relay from BS to SS.
6. Plaintexts are encrypted by a two-dimensional stream cipher technique such that the wireless signal propagation from the sender to the receiver can be more securely performance than before.

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 and high efficiency authentication key exchange mechanism with which SS and BS can more safely 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 article is organized as follows. Chapter 2 and 3 describes background, related work and PKMv1 of this study. Chapter 4 and 5 introduces the new method and how it provides a

more secure and convenient environment than what PKMv1 can. Simulation and discussions are presented in chapter 6. Chapter 7 concludes this article and outlines areas of future research.

Chapter 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 [1], 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 with the its' MAC address as the source MAC address to request ranging parameters. BS based on the MAC address assigns a channel ID (CID for short), 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 [9] 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 [10] 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 in a network, two parties, e.g., A and B, as shown in Figure 1, 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 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= $g^{X_a X_b} \bmod P$.

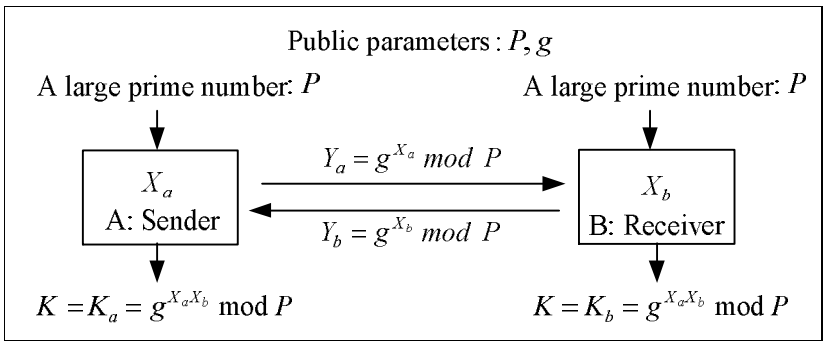

Figure 1. The generation of a Common secrete key by the DH-PKDS

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 [7].

2.3 Data Carriers

A key encrypted by a sender, transmitted through a wireless channel and then decrypted by a receiver is very usual way to transmit the key in a wireless system. All invertible functions, such as exclusive-or function and binary adder function employed to encrypt keys and data, are called data carriers. The necessary condition that a data carrier can securely carry a key is that at

least one connection data for authentication (CDA for short) currently exists on both sides. After exchanging keys, the sender and the receiver can then authenticate each other.

2.4 Identity certification key

Assume that keys X, Y, and Z are CDAs between the sender and receiver. Their certification key, denoted by $Certfun(X,Y,Z)$, in this study, is defined as $Certfun(X,Y,Z) = (X \oplus Y) + Z$ which brings forth the security in that the sender and the receiver can verify the received *Certfun(X,Y,Z)*, but the hackers have no way to acquire the three keys and *Certfun(X,Y,Z)*. Hence, it is a novel method that can efficiently resist forgery attacks.

2.5 Mutual authentication mechanism

The mutual authentication process performed between a sender and a receiver by employing keys *X*, *Y*, and *Z* is as follows:

For each message *M* sent by the sender through wireless channels, at least one of the there keys, $X+Y$, $X \oplus Y$ and $X \oplus Y+Z$, should be involved as one of attributes.

The receiver on receiving *M* checks to see whether the key *Ks* received is equal to the key *Kr* calculated by the receiver itself or not, where $Kr = X+Y$, $X \oplus Y$, or $X \oplus Y+Z$. If yes, then *M* is legal. Otherwise, the receiver discards *M*.

2.6 Pseudo random number generator

A pseudo-random number generator [12] (PRNG for short) is a deterministic polynomial-time algorithm satisfying two conditions, expandability and Pseudo-randomness. The input that triggers

the generator to generate random number sequence is called the seed. The expandability requires the algorithm to output a *l*-bit string by inputting an n-bit seed, where $l>n$.

A pseudo-random sequence is cryptographically secure if it has two properties. The first is that it looks random, and should pass all given statistical tests of randomness. The second is that it is unpredictable. It is not easy to design a highly secure and fast PRNG with a long recycle period [13]. There are famous PRNGs, such as the QR-generator [14], the Yarrow-160 generator [15], and the Grey-PRNG [16]. Figure 2 shows the working model of a PRNG in which the seed may be either a bit string or a set of random numbers, and the output is an random number sequence, $\psi_0,\psi_1,\cdots,\psi_{q-1}$, called pseudo random number sequence (PRNS for short), where q is the length of a PRNS.

Figure 2. Block diagram of PRNG.

2.7 Two-dimensional stream cipher technique

The stream cipher, as widely used in wireless network [17], is a symmetric key cipher method since the sender and the receiver generate the same cipher stream so that the ciphertext can be accurately decrypted, and plaintext units, e.g. bytes, are sequentially encrypted one at a time by using a PRNS, typically by an exclusive-or (xor) or add operation. That is plaintext units are encrypted by different elements of the cipher stream. Figure 3 shows the block diagram of a stream cipher system.

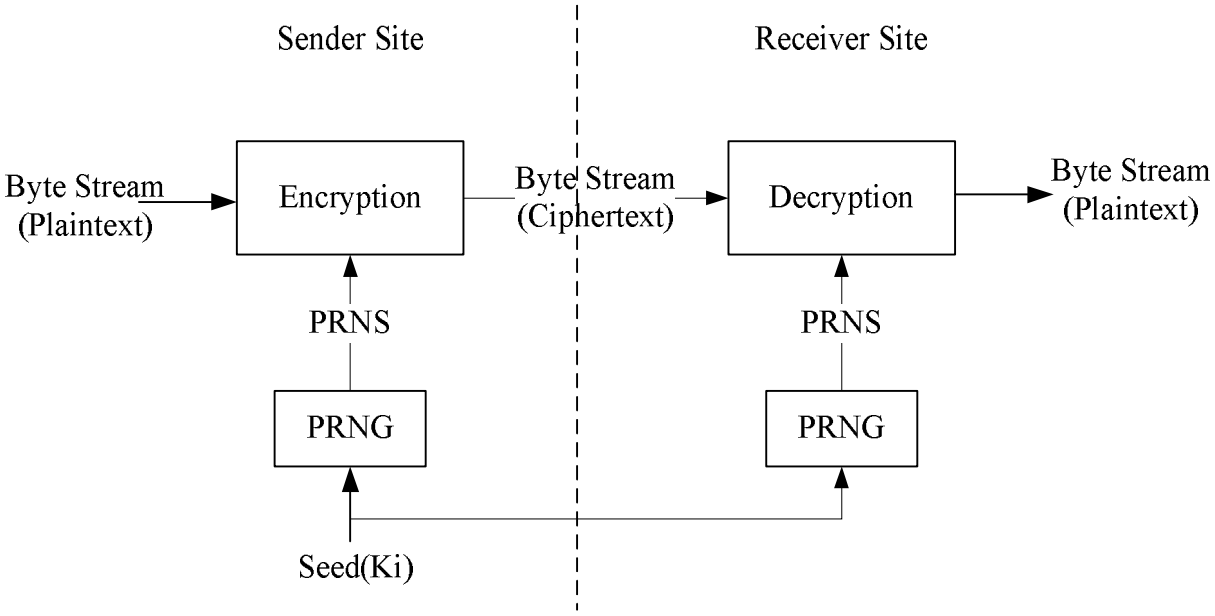

Figure 3. Block_diagram of a stream cipher system.

A stream cipher with which plaintext is encrypted with two different PRNSs by two different operations is called a two-dimensional stream cipher which has higher security than a stream cipher technique has [18].

2.8 Related Work

Rahman and Kowsar [5] 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 ISSI 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 safe since both R_{BS} and R_{SS} are not delivered through wireless channels, and it employed a cryptographic function and an ISSI to prevent the man-in-the-middle attack. However, once the key is solved by hackers, this method will lose its protection capability.

Han et al. [6] 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. The security process is as follows.

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, and cascades H(TSSI) , R_{BS} and its public key PK_{SS} to generate the response, $H(H(TSSI)||\ R_{BS}\ ||\ PK_{SS})$. SS 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 the 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 established.

Basically, Han et al's approach is more strict than that of [6]. But Han et al. as stated above did

not specify how to generate TSSI from ISSI, and how SS and BS know each other's ISSI.

Chapter 3 PKMv1 Model

In PKMv1, two important keys, authentication key (AK for short) and traffic encryption key (TEK for short) both generated by BS by invoking the RSA algorithm, are involved in PKMv1's authentication process. *AK* is produced for authorization, and *TEK* is generated to encrypt transferred data.

3.1 PKMv1 Process

Figure 4 shows the PKMv1 process in which five steps are performed before *TEK* can be used to encrypt data.

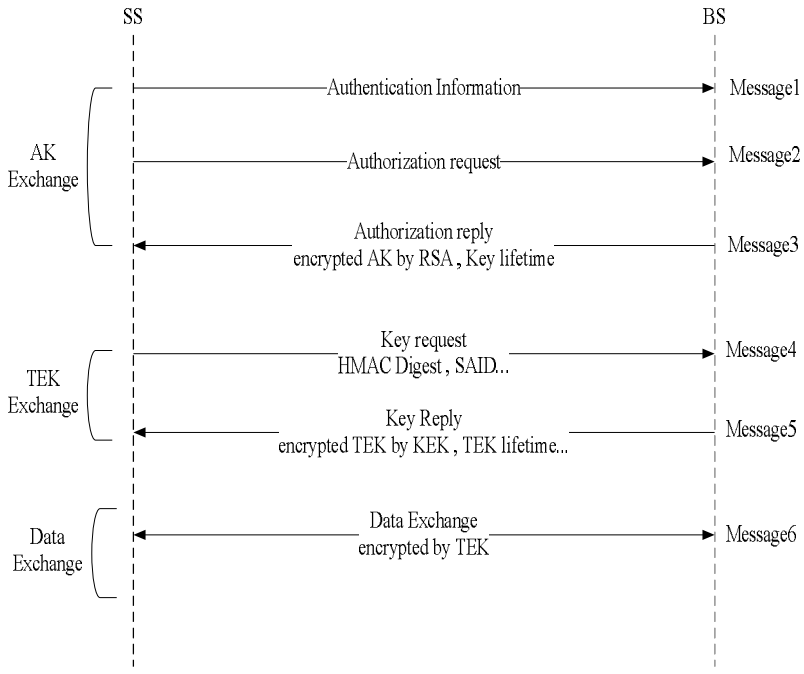

Figure 4. PKMv1 process in PMP mode [11]

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 [2] ignore this message. Figure 5 shows the message format. The PKM code, one byte in length, is used to identify the type of the PKM message. The code values are defined in Table 1. When a message is received with an invalid code, it will be discarded. The *Cert(SS Manufacturer)* attribute contains an X.509 CA certificate identifying the CA or the external authority that issued the certificate to the manufacturer .

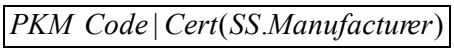

Figure 5. An authentication-information message (message 1)

Table 1. PKM message codes

PKM code	PKM message type	MAC management message
0-2	Reserved	--
3	SA Add	PKM-RSP
4	Authentication Request	PKM-REQ
5	Authentication 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	--

After sending an authentication-information message to BS, SS immediately delivers an authorization-request message to BS (i.e., message 2). Figure 6 shows the message format in which the *Cert(SS)* contains an X.509 SS certificate issued by SS's manufacturer. The certificate is a public key that binds SS's identifying information to its RSA public key in a verifiable manner [1]. In fact, the X.509 certificate is digitally signed by SS's manufacturer, and that signature can be verified by a BS that knows the manufacturer's public key. The manufacturer's public key is placed in an X.509 certification authority certificate, which in turn is signed by a higher-level CA. The *Security-Capabilities* is a compound attribute conveying the requesting SS's security capabilities, including the data encryption and authentication algorithms, that SS supports. An SAID attribute contains a Privacy SAID, which as the Basic CID assigned to SS during the initial ranging phase, is SS's Basic CID.

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

Figure 6. An authentication-request message (message 2)

Step2: BS on receiving the authorization-request message 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 7 also contains the *AK*'s lifetime, a 4bits *AK* sequence number used to identify the chosen *AK* from other ones, 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 | *AK* | *AK – lifetime* | *AK – sequence – Number* | *SA – Descriptor*

Figure 7. An authentication-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 8 contains a 160 bits *HMAC-Digest* derived from the *AK* for downlink authentication.

PKM Code| *AK – Sequence– Number*| *SAID*| *HMAC– Digest*

Figure 8. 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, 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 contains a old *TEK* and a new *TEK*. When the old expires, the new one will be used. Both are encrypted by the *KEK*. Figure 9 shows the message.

PKM Code | *AK – Sequence – Number* | *SAID* | *old TEK parameters* | *new TEK parameters* | *HMAC – Digest*

Figure 9. A key-reply message (message 5)

Step5: SS on receiving message 5 checks to see whether or not BS is legal by validating the received *HMAC-Digest* value. After the checking, SS decrypts the value by using *KEK*. With the

value, *TEK*s can be recovered. Data messages are then encrypted by *TEK* before their delivery between SS and BS.

3.2 IEEE802.16 Encryption

IEEE802.16 uses data encryption and decryption to achieve data privacy. The algorithms used can be classified into two types, symmetrical and asymmetric.

A) Symmetric Cryptography: This kind of algorithms uses the same key to encrypt and decrypt data. The most representative one is Data Encryption Standard (DES for short) [19], which is a block cipher (a form of shared secret encryption) that uses a 56-bit key. However, DES has been broken in 22 hours and 15 minutes. The algorithm is believed to be practically secure in the form of Triple DES, although several theoretical attacks exist. In recent years, the cipher has been superseded by the Advanced Encryption Standard (AES) [19].

B) Asymmetric Cryptography: This kind of algorithms uses different keys to encrypt and decrypt data, known as a public key cryptosystem. The most representative one is the Rivest Shamir Adleman (RSA for short) algorithm [20], which is used to encrypt SS's public key in an authentication reply message.

To break the RSA algorithm is the problem of solving an integer factorization problem [20], which is very difficult. In other words, RSA algorithm is reliable. Today, only a short RSA key can be cracked. Up to the year of 2008, the world does not have any effective method to attack RSA algorithm. When the key is long enough, the RSA encrypted information in fact can not be easily

cracked in. The RSA scheme is as follows [21]:

(1) Key generation algorithm

The algorithm to generate the keys for an entity, e.g., entity A , is as follows.

1). randomly and secretly chooses two large prime numbers p and q.

2). computes the modulus $n = p \times q$.

3). computes $\theta(n) = (p-1)(q-1)$

4). selects random integer e, $1 < e < n$ where $gcd(e, \theta(n)) = 1$

5). uses Baghdad method [22] to compute the unique decrypted key d, $1 < d < \theta(n)$ where

$e \times d \equiv 1 \bmod \theta(n)$

6). determines entity A's public and private keys. The pair $(d, \theta(n))$ is the private key, whereas the pair (n, e) is the public key.

(2) Public key encryption algorithm

Algorithm 1 lists the steps used by entity B to encrypt plaintext (i.e., message) m for entity A.

Algorithm 2 shows the steps used by entity A to decrypt m.

Algorithm 1 Encryption: performed by entity B.

Input: entity A's public key (n, e) , and a plaintext m.

Output: the encrypted message c.

1). Represent m as an integer in the interval [0...n-1]

2). Compute $c = m^e \bmod n$

Algorithm 2 Decryption: To recover m from c by Entity A.

Input: c from entity B

Output: m

1). Recover m by the $c^d \bmod n$

3.3 IEEE802.16 Security Analyses

In PKMv1, only BS authenticates SS, and the authentication as shown in Figure 10 does not testify the completeness of messages. SS may possibly connect to a fake BS. To avoid this spoofed attacks [23], interrogation messages should be added.

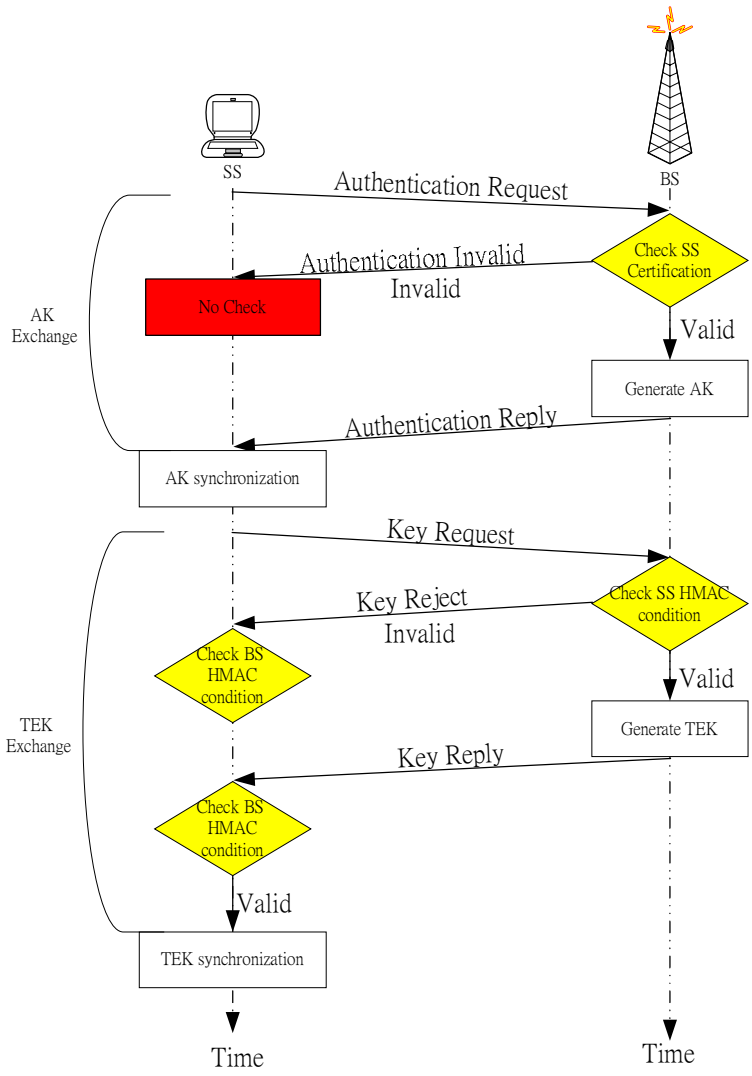

Figure 10. PKM process analyses

As a random number, an *AK*'s random number generator should be trustable. Otherwise, once the algorithm is known to the hackers, the hackers can then accomplish all authentication steps or establish a fake *TEK* to steal user's secret data. 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 the lifetime, up to $3360(=\frac{70*24}{0.5})$ *TEK*s can be gathered and used to decode messages. WiMAX employs the RSA algorithm to protect *AK*. However, the RSA-768 algorithm has been cracked [24]. In fact, the RSA Factoring Algorithm is rather complex. It is difficult to break the RSA-1024 [24].

Chapter 4 The Proposed Approach

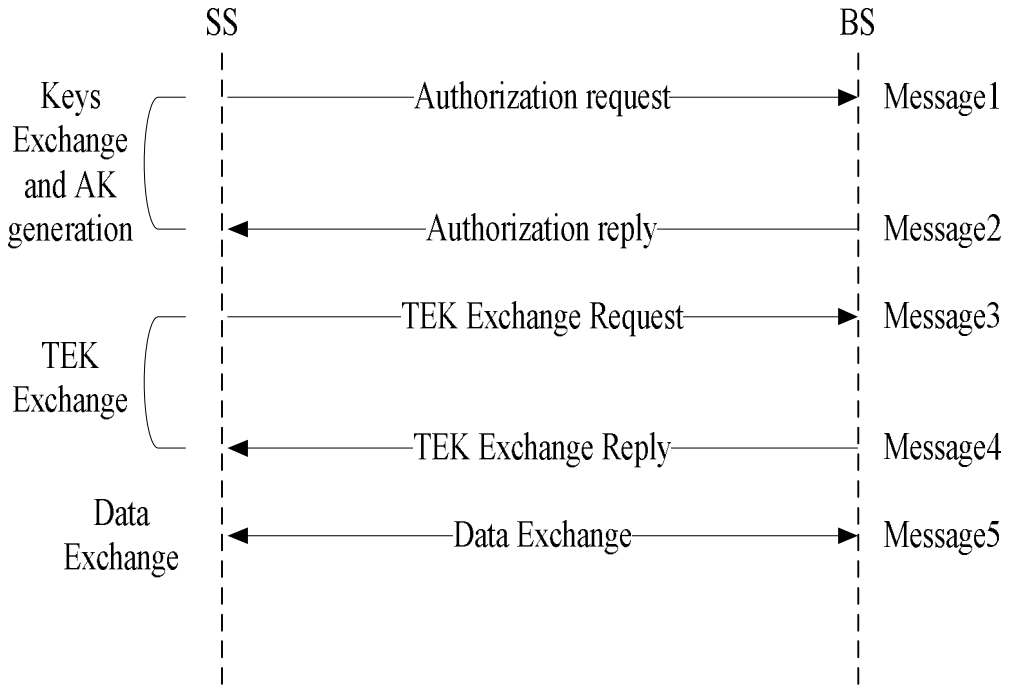

Figure 11. P-DiHam process

In this study, the *TEK* exchange method is different from the original PKMv1 standard in that in the PKMv1 *TEK* is encrypted 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 11 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.[25]

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

Mutual authentication function: $Certfun(x, y) = g^{x+y} \bmod P$

Data carriers function: $EXOR(x, y) = x \oplus y$

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

Table 2. Operation Codes and their descriptions

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

4.2 Applying PKDS to WiMAX Authentication

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

Step1：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 authentication-request message (i.e., message 1 shown in Figure 11) to BS. Figure 12 illustrates the format in which OP_Code =1.

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

Figure 12. An authentication-request message (message 1 from SS to BS) with *OP_Code*=1.

The *Cert(SS Manufacture)* and the *Cert(SS)* contain X.509 digital certificates as the facility certificates to identify 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 requirement between SS and BS. If only one public key were sent to BS, which of course generates

only one common key. The security level of the system would be lower.

Step2：BS on receiving authentication-request message validates whether SS is a legitimate user of the system by checking user certificate *Cert(SS)*. If not, BS sends an authentication-invalid message (see Figure 13a), i.e., OP_Code=3, to SS. Otherwise, BS produces three private keys *RB1*, *RB2* and *RB3*, with which BS generates three common secret keys *CSK1*, *CSK2* and *CSK3* where $CSKi = P_{RSi}^{RBi} \bmod P = g^{RSi*RBi} \bmod P$ ，$1 \le i \le 3$. BS then yields three public keys P_{RB1}, P_{RB2} and P_{RB3} where $P_{RBi} = g^{RBi} \bmod P, 1 \le i \le 3$. BS further invokes its random number generator to produce three *pre_AK*s (i.e., *pre_AK*1, *pre_AK*2 and *pre_AK*3), produces five *AK*s, *AK*1 ~ *AK*5, by using HMAC-SHA algorithm [23] where

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

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

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

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

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 $EXOR(CSKi, pre\_AKi), i = 1,2,3$, for further authentication, and at last it passes an authentication-reply message (i.e., message 2), of which the format is shown in Figure 13b 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*. With *Certfun(CSK1, CSK2)* is

for SS to authenticate BS, i.e., SS authenticates BS twice (the facility certification and user authentication). *XOR*(*CKSj*, *pre_AKj*), 1≤j≤3, carry the security parameters *pre_AK1*, *pre_AK2* and *pre_AK3* to BS.

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 level. Five *AK*s are generated to support the production of the following 75 *TEK*s.

$$\boxed{OP\_Code \mid Cert(SS) \mid P_{RB1} \mid P_{RB2} \mid Certfun(CSK1, CSK2)}$$

(a) An authentication-invalid message with OP_Code=3.

$$\boxed{\begin{array}{l} OP\_Code \mid Cert(SS) \mid P_{RB1} \mid P_{RB2} \mid P_{RB3} \mid Certfun(CSK1, CSK2) \mid \\ EXOR(CSK2, pre\_AK2) \mid EXOR(CSK3, pre\_AK3) \end{array}}$$

(b) An authentication-reply message with OP_Code=2.

Figure 13. An authentication-invalid message with *OP_Code*=3 (message 3 from BS to SS) and an authentication-reply message (message 2 from BS to SS) with *OP_Code*=2.

Step3：SS on receipt of the message first checks to see whether the message is a legitimate or not by authenticating the certificate *Cert(SS)*. If not, it discards the message. If yes, it further checks to see whether BS authentication code *Certfun(CSK1,CSK2)* is correct or not. If yes, that means BS is legal. Then, SS retrieves P_{RB1}, P_{RB2} and P_{RB3} from the message to generate its common secret keys, $CSKi = P_{RBi}^{RSi} \bmod P = g^{RBi*RSi} \bmod P, 1 \le i \le 3$, and obtains *pre-AKi* by invoking encryption functions $EXOR(CSKi, pre\_AKi), i = 1,2,3$. Finally, SS generates $AK1 \sim AK5$ by using

the same HMAC-SHA algorithm.

SS sends a *TEK*-exchange-request message (i.e., message 3) 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 for level 1, of which the format is shown in Figure 14 in which security-capabilities = 1, telling BS that SS requests a level-1 security. The role of *Certfun(CSK1, pre_AK1)* is mentioned above, to request BS generate a *TEK*.

$$\boxed{OP\_Code \mid Certfun(CSK1, pre\_AK1) \mid Security-capabilities}$$

Figure 14. 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 $EXOR(CSK2, pre\_TEK)$ and sends a level-2 *TEK*-exchange-request message (i.e., level-2 message 3) to BS. The message format is shown in Figure 15 in which *OP_Code*=4 and *Security-capabilities* = 2, indicating that SS requests a Level-2 security. The roles of *Certfun*() and *EXOR*() are individually mentioned above.

$$\boxed{OP\_Code \mid Certfun(CSK1, pre\_AK1) \mid EXOR(CSK2, pre\_TEK) \mid Security-capabilities}$$

Figure 15. 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**：There are two phases:

Phase1：SS produces 15 pre_TEKs and 75 *TEKs* where

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

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

Phase2：it sends the level-3 *TEK*-exchange-request message in which *security-capabilities* = 3 to BS. The message format is illustrated in Figure 16.

$$\boxed{OP\_Code \mid Certfun(CSK1, pre\_AK1) \mid Security-capabilities}$$

Figure 16. 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.

Step4：BS on receiving the *TEK*-exchange-request message checks the authentication code $Certfun(CSK1, pre\_AK1)$ contained in the message to see whether SS is legal or not. If yes, BS checks the security-capabilities and generates the corresponding *TEK* 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 *EXOR(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 17. *Certfun*(*CSK2, pre_AK2*) is the third authentication key (besides *Cert(SS)* and *Certfun(CSK1*,

pre_AK1)) between SS and BS. The two *EXOR()*s are used to carry *old_TEK* and *new_TEK* to SS.

$$OP\_Code \mid Certfun(CSK2, pre\_AK2) \mid EXOR(CSK2, old\_TEK) \mid EXOR(CSK3, new\_TEK) \mid TEK\ lifetime$$

Figure 17. 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**：BS on receipt of the level-2 *TEK*-exchange-request message decrypts and recovers the *pre_TEK* and generates the five *TEK*s also by using the formulas $TEKi = g^{AKi + pre\_TEK} \bmod P,\ 1 \le i \le 5$. Next, it further chooses one of the *TEK*s and sends the level-2 *TEK*-exchange-reply message with *OP_Code*=5 to SS. The message format is shown in Figure 18 in which the *TEK* sequence number (a number between 1~5) is employed to tell SS which *TEK* is chosen.

$$OP\_Code \mid Certfun(CSK2, pre\_AK2) \mid TEK\ seq\ num \mid TEK\ lifetime$$

Figure 18. 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**：BS on receiving the level-3 *TEK*-exchange-reply message calculates the *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 then sends a level-3 *TEK*-exchange-reply message which contains the *TEK* sequence number (a number between 1~75) to SS. The message format is

shown in Figure 19.

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

Figure 19. A level-3 TEK-exchange-reply message (i.e., level-3 message 4 from BS to SS) with *OP_Code*=5.

Step5：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. Another, because the *TEK* is a random number, so the quantities are 2^n.

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*. Another because the pre_TEK is a random number, so the TEK quantities

are 2^n.

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.

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

Item / TEK	Perform.	Hardware	Security	TEK Qty
Level 1	High	Low	Low	2^n
Level 2	Middle	Middle	Middle	2^n
Level 3	Low	High	High	75

4.4 P-DiHam Security Analyses

The biggest 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 to receive more information. Therefore, the security of wireless transmission is very important.

P-DiHam uses *EXOR* (*X, Y*) to carry and protect the transmitted data, and *Certfun* (*X, Y*) to provide authentication. When hackers intercept *EXOR*(*X, Y*), the probability that they can solve *Y* from *EXOR*(*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 X and Y on one trial from illegally received *EXOR (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 $EXOR(X,Y) = z_{n-1}...z_2z_1z_0$ 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 (x_i, y_i) pair on one trial is $\frac{1}{2}$, and then the probability to correctly recover original (*X, 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., $RS1 \sim RS3$ and $RB1 \sim RB3$. Once either set is successfully recovered, *CSK1*, *CSK2*, *CSK3* and *AK* can be then produced. However, P-DiHam security solution is based on the difficulty of solving discrete logarithm problem. The time complexity of solving the problem currently known is $O(\exp\sqrt{cm \ln m})$ [10][31], where *c*=0.69 and *m* is the length of public

key. If the length of the public key is 256bit, the time required is 3.9093×10^{13} s (=1.2 million years). In other words, the P-DiHam is a secure and safe system.

Chapter 5 Advanced P-DiHam

In the following, we would like to propose an enhanced version of the P-DiHam, called Advanced P-DiHam (AP-DiHam for short). The DH-PKDS spends a lot of time to calculate keys. The longer the *AK*, the more time it consumes for the calculation. Despite of high security, the time spent by the P-DiHam is highly related to the security levels. Both Level-3 *TEK*-exchange-request and the Level-3 *TEK*-exchange-reply steps invoke the DH-PKDS algorithm 75 times. That is why AP-DiHam is proposed to shorten the computation time, and further improve its security and efficiency by using two-dimensional stream cipher to encrypt and decrypt delivered messages.

5.1 New Parameters, Functions and OP_Codes

The new parameters defined for the proposed method are as follows.

Pubkey(SS): The SS's public key.

NTEKi, i=1,2,3: New traffic encryption keys.

The new functions defined and used include

$DH(P, g, X) = g^X \bmod P$

$Certfun(X, Y, Z) = (X \oplus Y) + Z$, a identity certification key.

$ADR(X, Y) = X + Y$, a binary adder but ignoring the carry of the greatest significant bit.

$RHSEXOR(X, Y) = RHS(X) \oplus Y$, where RHS(X) and Y are of the same size by truncating X's most significant bits.

The new OP_Codes employed are described in Table 4.

Table 4. New Operation codes and their descriptions

OP_code	Description
1	Authentication request
2	Authentication reply and key reply
3	Authentication failure
4	Data transmission request
5	Data transmission reply
6	Data transmission failure
7	Data transmission
0,8-15	Reserved

5.2 The method of advanced P-DiHam

The operation flow of the proposed method, which consists of 10 steps, is shown in Figure 20 in which Step1 through step4 are the Authentication phase. Step5 through step8 are the Pre_data transmission phase, and Step9 and step10 are Data transmission phase.

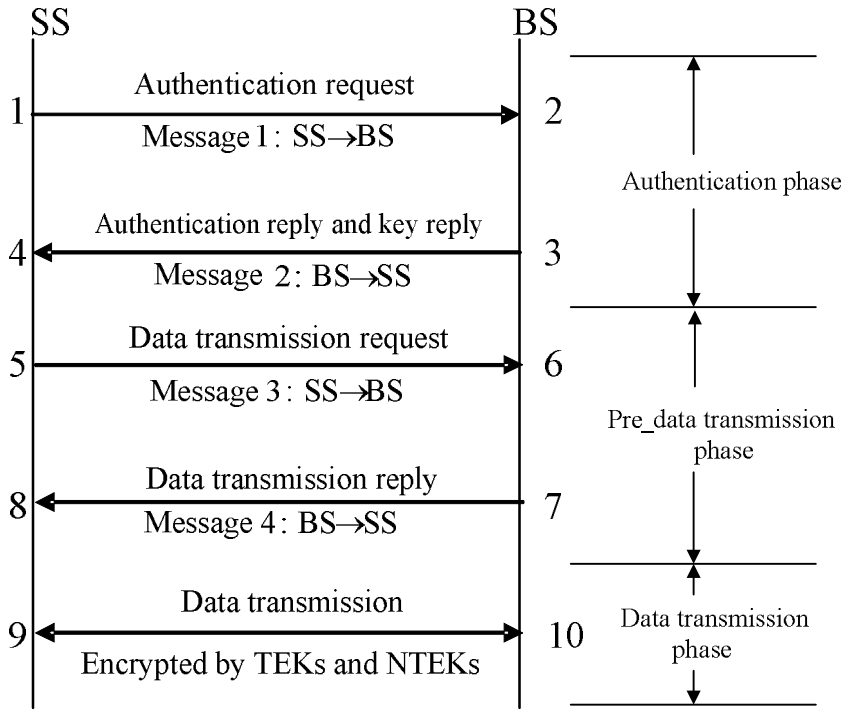

Figure 20. Operation flow of the proposed method.

5.2.1 Authentication phase

This phase is involved to exchange keys for authentication.

Step 1：SS first self-produces three random numbers, *RS1*, *RS2*, and *RS3*, as its private keys. It further generates three public keys P_{RS1}, P_{RS2} and P_{RS3} where $P_{RSi} = g^{RSi} \bmod P, 1 \le i \le 3$, and sends an authentication-request message (i.e., message 1) with OP_code = 1 to BS. Figure 21 shows format of the message.

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

Figure 21. Format of the authentication request message (message1)

Step 2：BS on receiving the message retrieves the PubKey(SS) from *Cert(SS)* and randomly selects three random numbers *RB1, RB2,* and *RB3* from its pre-produced internal random number table as

its private keys, and retrieves the three corresponding public keys P_{RB1}, P_{RB2} and P_{RB3} also from the table where $P_{RBi} = g^{RBi} \bmod P, 1 \le i \le 3$. After that, it generates the three common secret keys *CSK1*, *CSK2* and *CSK3* where $CSKi = P_{RSi}^{RBi} \bmod P, 1 \le i \le 3$, and the identity certification key *Cerfun(PubKey(SS),CSK1,CSK2)*. The PubKey(SS) and the three common secrete keys CSK1, CSK2, and CSK3 are employed as the CDA between SS and BS.

Step 3：BS further validates whether SS is legitimate or not by checking PubKey(SS). If not, BS generates the identity certification key $Cerfun(CSK1, CSK2, CSK3)$ and writes the error messages into a fault list FA_List, and sends an authentication-failure message (i.e. message 2a) with OP_code = 3, to SS. Figure 22, shows format of the message.

$$OP\_code \mid P_{RB1} \mid P_{RB2} \mid P_{RB3} \mid Certfun(CSK1, CSK2, CSK3) \mid FA\_List$$

Figure 22. Format of the authentication-failure message (message 2a).

If yes, BS selects another three random numbers as *pre_AK*1, *pre_AK*2, and *pre_AK*3 from its internal random number table, and sends an authentication-reply-and-key-reply message (i.e., message 2b) with OP_code = 2, of which the format is shown in Figure 23, to SS.

$$\begin{aligned} &OP\_code \mid PubKey(SS) \mid P_{RB1} \mid P_{RB2} \mid P_{RB3} \mid Certfun(CSK1, CSK2, PubKey(SS)) \mid \\ &ADR(CSK1, pre\_AK1) \mid ADR(CSK2, pre\_AK2) \mid ADR(CSK3, pre\_AK3) \end{aligned}$$

Figure 23. Format of the authentication-reply-and-key-reply message (message 2b)

Meanwhile, BS produces

(1) six *AK*s, i.e., AK1 ~ AK6, by using HMAC-SHA algorithm where

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

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

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

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

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

$$AK6 = HMAC - SHA(CSK1, CSK3 \mid CSK2 \mid pre\_AK3 \mid pre\_AK1 \mid SS\_MAC\_Addr),$$

(2) 243 traffic encryption keys, TEKj, 1≤ j ≤ 243, where

$TEK_{(i-1)\times 81+(j-1)\times 9+k} = (AK_i \oplus TAK_j) + TCK_k, 4 \le i \le 6, 1 \le j, k \le 9$ in which

$TAK_{(i-1)\times 3+j} = AKi + pre\_AKj, 1 \le i, j \le 3$, and
$TCK_{(i-1)\times 3+j} = CSKi + pre\_AKj, 1 \le i, j \le 3.$

(3) 243 new traffic encryption keys, NTEKj, 1≤ j ≤ 243, where

$NTEK_{(i-1)\times 81+(j-1)\times 9+k} = (AK_i \oplus NTAK_j) + (NTCK_k \oplus AK6), 1 \le i \le 3, 1 \le j, k \le 9.$ in which

$NTAK_i = AK4 \oplus TAK_i, 1 \le i \le 9$, and
$NTCK_j = AK5 \oplus TCK_j, 1 \le j \le 9.$

Step 4：SS on receiving of the message checks the OP_code to see.

(1) If OP_code = 3, SS first self-produces the three common secret keys $CSKi = P_{RBi}^{RSi} \bmod P$, $1 \le i \le 3$, and authenticates BS by checking to see whether *Cerfun(CSK1,CSK2,CSK3)* received is the same as the value calculated by using the three self-produced parameters *CSK1, CSK2* and, *CSK3.*

If not indicating this is a fake message, it discards the message and waits for an authentication-reply-and-key-reply message from a valid BS. If yes, it retrieves the error messages contained in FA_List, shows the error message to the user and then terminates the authentication

connection request. The user may resubmit another request later.

(2) If OP_code = 2 (i.e., an authentication-reply-and-key-reply message), SS first retrieves three public keys P_{RB1}, P_{RB2} , and P_{RB3} from the message, and calculates *CSKi*, $CSKi = P_{RBi}^{RSi} \bmod P, 1 \le i \le 3$, and the identity certification key *Cerfun(PubKey(SS),CSK1,CSK2)*. SS authenticates BS by comparing the retrieved *Cerfun(PubKey(SS),CSK1,CSK2)* and the calculated one. If they are not equal, SS discards the fake message and waits for an authentication-reply-and-key-reply message issued by a valid BS. Otherwise, it recovers *pre-AKi* by invoking reverse function of the data carriers $ADR(CSKi, pre\_AKi),\ i = 1,2,3$.

5.2.2 Pre_data transmission phase

This phase is involved to make sure whether or not both of the sender and the receiver can communicate with each other through BS.

Step 5：SS produces six *AK*s, i.e., *AK*1 ~ *AK6*, by using the same HMAC-SHA algorithm shown above, and generates 243 traffic encryption keys, TEKj, 1≤ j ≤ 243, and 243 new traffic encryption keys, NTEKj, 1≤ j ≤ 243, by using the same functions defined above. SS further sends a data-transmission-request message (i.e., message 3) with OP_code = 4 to BS. Figure 24 shows the message format.

$$\boxed{OP\_Code \,|\, Certfun(AK1, AK2, AK3)}$$

Figure 24. Format of the data transmission request message (message 3).

Step 6：BS on receipt of the message authenticates the message by comparing

Cerfun(AK1,AK2,AK3) calculated and the one retrieved from the message. If they are not equal, BS discards the fake message and waits for a data-transmission-request message issued by the valid SS. Otherwise, it proceeds to the next step.

Step 7：If the correspondent node (CN for short) of SS is now on line and can be contacted by BS, BS sends a data-transmission-reply message (i.e., message 4a) with OP_code = 5 to SS. Figure 25 show the message format.

$$\boxed{OP\_Code \,|\, Certfun(AK4, AK5, AK6)}$$

Figure 25. Format of the data transmission reply message (message 4).

If the CN is now off line, BS sends an Transmission-request-failure message (i.e. message 4b) with OP_code = 6 to SS. The message format reuses the one shown in Figure 25.

Step 8：The SS on receipt of the message authenticates the message with the same process mentioned in step 6 with *Cerfun(AK1,AK2,AK3) replaced by Cerfun(AK4,AK5,AK6)*. If the authentication fails, SS discards the fake message and waits for a message issued by the valid BS. Otherwise, SS terminates the pre_data transmission phase, and proceeds to the next step.

5.2.3 Data transmission phase

This phase is involved to transmit data messages.

Step 9：If the plaintext of q bits in length can be partitioned into n l-bit segments, e.g., $plaintext_0$ ~ $plaintext_{n-1}$, i.e., $Pla\,\mathrm{int}\,ext = pla\,\mathrm{int}\,ext_0 + pla\,\mathrm{int}\,ext_1 + ... + pla\,\mathrm{int}\,ext_{n-1},\ n \geq 0$ where $n = \left\lceil \frac{q}{l} \right\rceil$.

The encryption process is $ciphertext_i = (pla\,\text{int}\,ext_i \oplus NTEK_j) + TEK_j, \quad 0 \le i \le n-1, \quad j = (i+m) \bmod 243, 0 \le \text{m} \le 242$ and

$Ciphertext = ciphertext_0 + ciphertext_1 + ... + ciphertext_{n-1}, n \ge 0$ SS sends the ciphertext, as a data message, to the BS. Figure 26 shows the message format.

$$\boxed{OP\_Code \mid RHSEXOR(AK6,m) \mid EXOR(TEKm, NTEKm) \mid Ciphertext}$$

Figure 26. Format of a data message from SS to BS

Step 10：BS authenticates the message by comparing the self-calculated value of the traffic certification key *EXOR(TEKm,NTEKm)*, and the value retrieved from the message. If they are not equal, BS discards the fake message and waits for the message issued by the valid SS. Otherwise, it decrypts the ciphertext with the following process.

$$pla\,\text{int}\,exti = \begin{cases} (ciphertext_i - TEK_j) \oplus NTEK_j, & if \;\; ciphertext_i \ge TEK_j \\ (ciphertext_i + \overline{TEK_j} + 1) \oplus NTEK_j, & if \;\; ciphertext_i < TEK_j \\ \text{and } 0 \le i \le n-1, \;\; j = (i+m) \;\bmod\; 243 \end{cases}$$ and

$Plaintext = plaintext_0 + plaintext_1 + ... + plaintext_{n-1}, \; n \ge 0$

5.3 AP-DiHam Security Analysis

In the AP-DiHam, *EXOR(X, Y)* and *ADR(X, Y)* as data carriers carry key *Y* from a sender to a receiver under the assumption that the *CDA*, i.e., key *X* is owned by both side beforehand. The probability that hackers can guess the correct *X* and *Y* pair on one trial is $\frac{1}{2^n}$, where $|x| = |y| = n$.

Theorem 2:

Assume that keys *X* and *Y* are both *n* bits in length. The probability *p* of recovering (*X,Y*) pair

values from illegally received *ADR(X,Y)* is $\frac{1}{2}\prod_{i=1}^{n-1}\frac{2^i-1}{2^{i+2}} \le p \le \frac{1}{2}\prod_{i=1}^{n-1}\frac{2^i+1}{2^{i+2}}$ and when $n >> 1$, $p \to \frac{2}{4^n}$.

Proof:

Proof is by mathematical induction. Let $X = x_{n-1}...x_2x_1x_0$, $Y = y_{n-1}...y_2y_1y_0$, and $ADR(X,Y) = z_{n-1}...z_2z_1z_0$ where x_i, y_i, z_i, $0 \le i \le n-1$, are binary digits. Let $C = c_{n-1}...c_2c_1c_0$ be the carry of $X + Y$, then when for $i = 0$, we have $z_0 = x_0 + y_0$.

If $z_1 = 0$, (x_0, y_0) may be either (0,0) or $(1,1)^c$ where $(1,1)^c$ represents that $x_0 + y_0$ yields a carry .

If $z_1 = 1$, (x_0, y_0) may be either (0,1) or (1,0) which yields no carry. Hence, given z_0 the probability of giving (x_0, y_0) the correct value on one trial is $\frac{1}{2}$. The probability of $c_0 = 1$ is $\frac{1}{4}$ since only $(1,1)^c$ generates a carry, and that of $c_0 = 0$ is $\frac{3}{4}$.

When $i = 1$, we have $z_1 = x_1 + y_1 + c_0$, and

if $z_1 = 0$, there are two cases.

(a1) When $c_0 = 0$, (x_1, y_1) maybe $(0,0)$ or $(1,1)^c$. The probability of giving (x_1, y_1) the correct value on one trial is $\frac{3}{4}\times\frac{1}{2} = \frac{3}{8}\left(= \frac{2^1+1}{2^{1+2}}\right)$ when $\frac{1}{2}$ is the probability of choosing $(0,0)$ or $(1,1)^c$.

(a2) When $c_0 = 1$, (x_1, y_1) may be $(0,1)^c$ or $(1,0)^c$. The probability of giving the correct value to (x_1, y_1) on one trial is $\frac{1}{4}\times\frac{1}{2} = \frac{1}{8}\left(= \frac{2^1-1}{2^{1+2}}\right)$ where $\frac{1}{2}$ is the probability of choosing $(0,1)^c$ or $(1,0)^c$.

If $z_1 = 1$, there are two cases. From i=1's viewpoint,

(b1) when $c_1 = 0$, (x_1, y_1) may be (0,1) or (1,0), the probability of giving the correct value to (x_1, y_1) on one trial is $\frac{3}{4} \times \frac{1}{2} = \frac{3}{8} \left(= \frac{2^1 + 1}{2^{1+2}} \right)$.

(b2) When $c_1 = 1$, (x_1, y_1) may be (0,0) or $(1,1)^c$. The probability of giving the correct value to (x_1, y_1) on one trial is $\frac{1}{4} \times \frac{1}{2} = \frac{1}{8} \left(= \frac{2^1 - 1}{2^{1+2}} \right)$.

Now, we can conclude that when $z_1 = 0, c_0 = 0,$ at $(0,0), (1,1)^c$ or $z_1 = 1, c_0 = 0,$ at (0,1), (1,0), the probability of giving the correct value to (x_1, y_1) on one trial is $\frac{3}{8} \left(= \frac{2^1 + 1}{2^{1+2}} \right)$. When $z_1 = 0, c_0 = 1,$ at $(0,1)^c$, $(1,0)^c$ or $z_1 = 1, c_0 = 1,$ at $(0,0), (1,1)^c$, the probability is $\frac{1}{8} \left(= \frac{2^1 - 1}{2^{1+2}} \right)$.

From the conclusion, we can derive the fact that there are a total of eight combinations, and the probability of $c_1 = 0$ is $\frac{1}{2} \left[\frac{3}{8} \times 3 + \frac{1}{8} \right] = \frac{5}{8} \left(= \frac{2^{1+1} + 1}{2^{1+2}} \right)$ since three combinations that generate $c_1 = 0$ are with the probability of $\frac{3}{8}$, and one combination that generates $c_1 = 0$ is with the probability of $\frac{1}{8}$. Similarly, the probability of $c_1 = 1$ is $\frac{1}{2} \left[\frac{1}{8} \times 3 + \frac{3}{8} \right] = \frac{3}{8} \left(= \frac{2^{1+1} - 1}{2^{1+2}} \right)$ since three combinations are with the probability of $\frac{1}{8}$ and one with $\frac{3}{8}$.

When $n = i, i \geq 1$, we assume that the probability of giving the correct value to (x_i, y_i) on one trial is $\frac{2^i + 1}{2^{i+2}}$ if $c_i = 0$, or $\frac{2^i - 1}{2^{i+2}}$ if $c_i = 1$, and the probability of $c_i = 0$ is $\frac{2^{i+1} + 1}{2^{i+2}}$ and that of $c_i = 1$ is $\frac{2^{i+1} - 1}{2^{i+2}}$. Then,

for $n = i + 1$, we have $z_{i+1} = x_{i+1} + y_{i+1} + c_i$.

If $z_{i+1} = 0$, there are two cases.

(c1) When $c_i = 0, (x_{i+1}, y_{i+1})$ may be $(0,0)$ or $(1,1)^c$. The probability of giving the correct value to (x_{i+1}, y_{i+1}) on one trial is $\frac{1}{2} \times \frac{2^{i+1}+1}{2^{i+2}} = \frac{2^{i+1}+1}{2^{(i+1)+2}}$ where $\frac{1}{2}$ as stated above is the probability of choosing $(0,0)$ or $(1,1)^c$, and $\frac{2^{i+1}+1}{2^{i+2}}$ is the probability of $c_i = 0$.

(c2) When $c_i = 1, (x_{i+1}, y_{i+1})$ may be $(0,1)^c$ or $(1,0)^c$. The probability of giving the correct value to (x_{i+1}, y_{i+1}) on one trial is $\frac{1}{2} \times \frac{2^{i+1}-1}{2^{i+2}} = \frac{2^{i+1}-1}{2^{(i+1)+2}}$.

If $z_{i+1} = 1$, there are two cases.

(d1) When $c_i = 0, (x_{i+1}, y_{i+1})$ may be $(0,1)$ or $(1,0)$. The probability of giving the correct value to (x_{i+1}, y_{i+1}) on one trial is $\frac{1}{2} \times \frac{2^{i+1}+1}{2^{i+2}} = \frac{2^{i+1}+1}{2^{(i+1)+2}}$.

(d2) When $c_i = 1$, (x_{i+1}, y_{i+1}) may be $(0,0)$ or $(1,1)^c$. The probability of giving the correct value to (x_{i+1}, y_{i+1}) on one trial is $\frac{1}{2} \times \frac{2^{i+1}-1}{2^{i+2}} = \frac{2^{i+1}-1}{2^{(i+1)+2}}$, and the probability of $c_{i+1} = 0$ is $\frac{1}{2}\left[\frac{2^{i+1}+1}{2^{(i+1)+2}} \times 3 + \frac{2^{i+1}-1}{2^{(i+1)+2}}\right] = \frac{2^{(i+1)+1}+1}{2^{(i+1)+2}}$, the probability of $c_{i+1} = 1$ is $\frac{1}{2}\left[\frac{2^{i+1}+1}{2^{(i+1)+2}} + \frac{2^{i+1}-1}{2^{(i+1)+2}} \times 3\right] = \frac{2^{(i+1)+1}-1}{2^{(i+1)+2}}$. Then by the mathematical induction, the probability p of giving the correct value to (x_i, y_i) from received z_i is,

$$p = \begin{cases} \frac{1}{2}, & \text{for } i = 0 \\ \frac{2^i+1}{2^{i+2}}, & \text{for } i \geq 1 \text{ and } c_{i-1} = 0 \\ \frac{2^i-1}{2^{i+2}}, & \text{for } i \geq 1 \text{ and } c_{i-1} = 1 \end{cases}$$

and the corresponding probability of recovering (x,y) $\frac{2^{i+1}+1}{2^{i+2}}$ and that of $c_{i-1}=1$ is $\frac{2^{i+1}-1}{2^{i+2}}$, for $i \geq 0$.

Now for each bit, there are two ways to choose (x_i, y_i) on receiving z_i, $i \geq 1$ and the probability of giving the correct value to (x_i, y_i) on one trial is $\frac{2^i+1}{2^{i+2}}$ when $c_{i-1}=0$ or $\frac{2^i-1}{2^{i+2}}$ when $c_{i-1}=1$. The possibility of choosing (x_i, y_i) when $c_{i-1}=0$ for all i, $1 \leq i \leq n-1$, the probability of giving correct value to (X,Y) on one trial is $\frac{1}{2}\prod_{i=1}^{n-1}\frac{2^i+1}{2^{i+2}}$, which is the maximum probability of giving correct value to (X,Y). The possibility of choosing (x_i, y_i) when $c_{i-1}=1$ for all i, $1 \leq i \leq n-1$, the probability of giving correct value to (X,Y) on one trial is $\frac{1}{2}\prod_{i=1}^{n-1}\frac{2^i-1}{2^{i+2}}$, which is the minimum probability of giving correct value to (X,Y). Then the probability p of giving correct value to (X,Y) from received Z on one trial is $\frac{1}{2}\prod_{i=1}^{n-1}\frac{2^i-1}{2^{i+2}} \leq p \leq \frac{1}{2}\prod_{i=1}^{n-1}\frac{2^i+1}{2^{i+2}}$. When $n >> 1$, the probability of giving the correct value to (x_i, y_i) on one trial from received z_i approaches $\frac{1}{4}$ for $i > 1$ and hence $p \to \frac{1}{2} \times \frac{1}{4^{n-1}} = \frac{2}{4^n}$. QED#

Assume that key X is the CDA between the sender and the receiver. The receiver can obtain Y by the following decoding process.

$$Y = \begin{cases} ADR(X,Y) - X, & \text{if } ADR(X,Y) \geq X \\ ADR(X,Y) + \overline{X} + 1, & \text{if } ADR(X,Y) < X \end{cases}.$$

But hackers can solve (X,Y) from the illegally received *ADR(X,Y)* with the probability of $\frac{2}{4^n}$ on one trial. When $n >> 1$, the probability is extremely smaller than $\frac{1}{2^n}$ which, by theorem 1, is the probability of giving the correct value to key Y given *EXOR(X,Y),* implying security level of using *ADR(X,Y)* is significantly higher than that of using *EXOR(X,Y)*.

When the identity certification key *Certfun(X,Y,Z)* is known to hackers, the probability of giving (*X*, *Y*, *Z*) the correct value on one trial , denoted by *p*, is $p \to \frac{2}{8^n}$, when n >> 1.

Let us consider the security of *Certfun(X, Y, Z)*. Where *X*, *Y*, and *Z* are binary numbers of *n* bits. The probability of giving the correct value to (*X*, *Y*, *Z*) on one trial given an illegally received *Certfun(X, Y, Z)* is $\frac{2}{8^n}$ when *n*>>1.

Theorem 3:

The probability *p* of trying to give the correct value to (*X*,*Y*,*Z*) on one trial when the identity certification key *Certfun(X,Y,Z)* is known is $\frac{1}{2^{n+1}}\prod_{i=1}^{n-1}\frac{2^i-1}{2^{i+2}} \le p \le \frac{1}{2^{n+1}}\prod_{i=1}^{n-1}\frac{2^i+1}{2^{i+2}}$ and when n >> 1, $p=\frac{2}{8^n}$.

Proof:

Assume that *Certfun(X,Y,Z)* $=(X \oplus Y)+Z$. When *Certfun(X,Y,Z)* is known to hackers, the probability p_1 of giving the correct value to $((X \oplus Y), Z)$ on one trial, by theorem 2, is between $\frac{1}{2}\prod_{i=1}^{n-1}\frac{2^i-1}{2^{i+2}}$ and $\frac{1}{2}\prod_{i=1}^{n-1}\frac{2^i+1}{2^{i+2}}$. The probability p_2 of giving the correct value to (*X*,*Y*) on one trial when the value of $(X \oplus Y)$ is known, by theorem 1, is $p_2=\frac{1}{2^n}$. Then, the probability *p* of giving correct value to (*X*, *Y*, *Z*) from the received *Certfun(X,Y,Z)* on one trial is $p = p_1 p_2$, i.e. $\frac{1}{2^{n+1}}\prod_{i=1}^{n-1}\frac{2^i-1}{2^{i+2}} \le p \le \frac{1}{2^{n+1}}\prod_{i=1}^{n-1}\frac{2^i+1}{2^{i+2}}$, and when *n*>>1, $p \to \frac{1}{2^{n+1}} \times \frac{1}{4^{n-1}} = \frac{2}{8^n}$. QED#

Cerfun(X,Y,Z) conveyed on messages 3 and 4 is used by the sender and the receiver to authenticate each other. But, hackers have no any information about *X*, *Y*, and *Z*. The probability that the hackers can guess the right *Cerfun(X,Y,Z)* value by choosing 0 or 1 for each bit of

Cerfun(X,Y,Z) is $\frac{1}{2^n}$, which is larger than $\frac{2}{8^n}$. It other words, *Cerfun(X, Y, Z)* is secure enough to protect a system from any forgery attacks. Further, the only way that can effectively break through our communication system is to solve private keys $RS1, RS2,$ and $RS3$ from public keys $P_{RS1}, P_{RS2},$ and P_{RS3} or $RB1, RB2,$ and $RB3$ from $P_{RB1}, P_{RB2},$ and P_{RB3}. But it is a discrete logarithm over finite field problem. Let *m* be the length of the strong prime *p*. Time complexity of the algorithm is $O(\exp\sqrt{c\,m\ln m})$ [10][31] which is a huge number, about 1.3 million years, showing that up to present there is no efficient way to solve the problems, where the best estimate for *c* = 0.69 [10]. Hence the security level of our proposed method is definitively high.

We further consider the security of ciphertext which is m*n bits in length. The probability to recover plaintext from illegally received ciphertext is $(\frac{2}{8^n})^m$ as n>>1.

Theorem 4:

Let plaintext Q be a string of m characters where $Q = q_{m-1}...q_2q_1q_0$ and each character q_i, $0 \le \mathrm{i} \le \mathrm{m}\text{-}1$, is n-bits in length. Let PRNS1 and PRNS2 be two pseudo random number sequences where PRNS1 is $r_0r_1, r_2,..., r_n, r_{n+1},...$, PRNS2 is $s_0s_1, s_2,..., s_n, s_{n+1},...$, and each r_j, s_j, $j \ge 0$ is a n-bit binary number. The ciphertext $C = c_{m-1}...c_2c_1c_0$ is obtained by encrypting plaintext Q with PRNS1 and PRNS2 as $c_j = (q_j \oplus r_j) + s_j$, $0 \le j \le m-1$. Then, the probability $\boldsymbol{p}$ to get $(q_{m-1}...q_2q_1q_0, r_{m-1}...r_2r_1r_0, s_{m-1}...s_2s_1s_0)$ from illegally received ciphertext $c_{m-1}...c_2c_1c_0$ is $\left(\frac{1}{2^{n+1}}\prod_{i=1}^{n-1}\frac{2^{i-1}-1}{2^{i+1}}\right)^m \le p \le \left(\frac{1}{2^{n+1}}\prod_{i=1}^{n-1}\frac{2^{i-1}+1}{2^{i+1}}\right)^m$ and $p \to \left(\frac{2}{8^n}\right)^{\mathrm{m}}$ as $n >> 1$.

Proof:

$\because c_j = (q_j \oplus r_j) + s_j,\ 0 \le j \le m-1$, then by theorem 3, the probability pr_j of getting (q_j, r_j, s_j) from the illegally received c_j is $\frac{1}{2^{n+1}} \prod_{i=1}^{n-1} \frac{2^{i-1}-1}{2^{i+1}} \le pr_j \le \frac{1}{2^{n+1}} \prod_{i=1}^{n-1} \frac{2^{i-1}+1}{2^{i+1}}$.

Since each triple (q_j, r_j, s_j) is independent with other triples, i.e., $pr_1 = pr_2 = ... = pr_m$, the probability p of getting $(q_{m-1}...q_2q_1q_0, r_{m-1}...r_2r_1r_0, s_{m-1}...s_2s_1s_0)$ from illegally received ciphertext $c_{m-1}...c_2c_1c_0$ is $\left(\frac{1}{2^{n+1}} \prod_{i=1}^{n-1} \frac{2^{i-1}-1}{2^{i+1}}\right)^m \le p \le \left(\frac{1}{2^{n+1}} \prod_{i=1}^{n-1} \frac{2^{i-1}+1}{2^{i+1}}\right)^m$. Also, $pr_j \to \frac{2}{8^n}$ as $n >> 1$, for each j, $0 \le j \le m-1$. Then $p \to \left(\frac{2}{8^n}\right)^m$ as $n >> 1$. QED#

Since PRNS1 and PRNS2 are the CDA between a sender and a receiver, the receiver on receiving a ciphertext can decode the ciphertext easily. The probability that the hackers can successfully crack the triple (Plaintext, PRNS1, PRNS2) without any information about PRNS1 and PRNS2 is $p \to \left(\frac{2}{8^n}\right)^m$ which is the probability of individually obtaining the plaintext and is extremely smaller than $\left(\frac{1}{2^n}\right)^m$ as $n >> 1$. In other words, it is a novel way to resist the attacks launched by hackers. Also, the probability that the hackers can acquire plaintext, PRNS1, and PRNS2 from illegally received ciphertext is $\left(\frac{2}{8^n}\right)^m$ which is almost equal to $\left(\frac{1}{8^n}\right)^m$ as $n >> 1$. The latter is the probability that the hackers correctly guess all plaintext, PRNS1 and PRNS2, i.e., plaintext, PRNS1, and PRNS2 are all well protected.

Chapter 6 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 [27][28]. The specifications of the simulation hardware are listed in Table 5. In this study, four experiments were performed. In the first, we measured P-DiHam timings required to process an authentication-request message and an authentication-reply message. In the second, we measured P-DiHam timings required to process level-1 to level-3 *TEK*-exchange-request messages. Both experiments were given different lengths of *TEK*s, including 256, 512, 768 and 1024 bits. In the third, we measured AP-DiHam timings required to process an authentication-request message and an authentication-reply and key-reply message. The fourth evaluated the timings required to encrypt and decrypt files of different sizes (include 1K, 512K, 1M and 10M). Both experiments were given different lengths of *AK*s, including 256, 512, 768 and 1024 bits. Each experiment was performed 50 times.

Table 5. Specifications of the simulations

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

6.1 Results of the P-DiHam authentication phase

The results of the first experiment are shown in Figure 27 and Figure 28. Figure 27 depicts the

fact that longer key lengths cause SS to spend a longer time to process an authentication-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 28, the time BS requires to generate an authentication-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 of that between 512-bit *AK*s and 256-bit *AK*s.

Further, on receiving an authentication-reply message, SS needs to certify BS by decrypting parameters conveyed on message so as to generate the *AK*s. So, the costs shown in Figure 28 are respectively longer than those plotted in Figure 11.

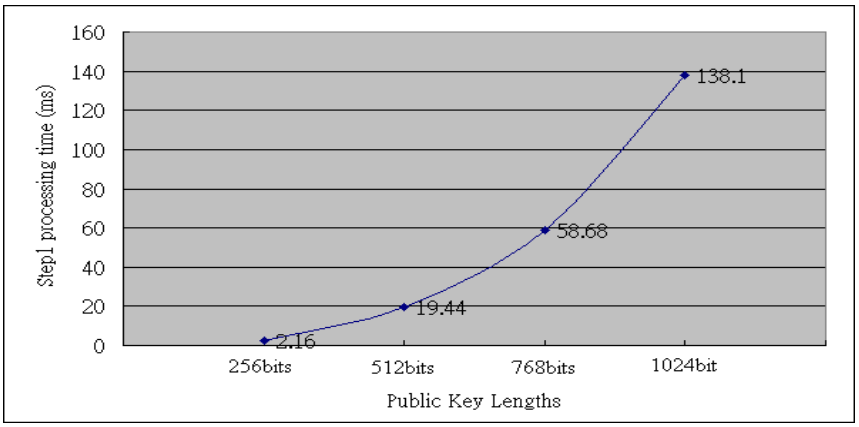

Figure 27. The times consumed by SS to generate an authentication-request message (i.e., step 1).

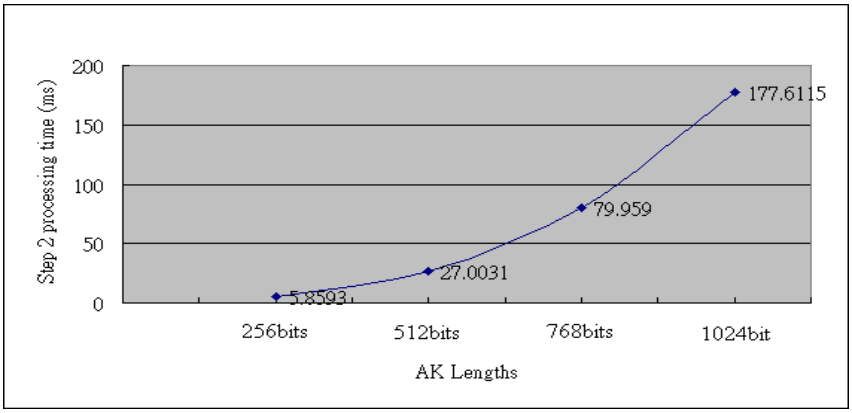

Figure 28. The times consumed by BS from the time point when it receives an authentication-request message to the time point when it generates an authentication-reply message (i.e., step 2).

6.2 Results of the P-DiHam TEK exchange phase

Figure 29-Figure 31 illustrate how *TEK* security levels and lengths of *TEK*s affect the processing time of the *TEK*-exchange-request and *TEK*-exchange-reply messages given three levels of *TEK*s. With level-1 *TEK*, SS only produces an authentication code without producing the *TEK*, so the trend of the processing times of level-1 *TEK* on different lengths of *TEK*s as shown in Figure 29 is similar to those of processing an authentication-request message and an authentication-reply message shown in Figure 27 and Figure 28, respectively, since the main tasks of the three message processes are identifying BS or SS. With 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 level-1 *TEK*. If Level-3 *TEK* is in use, SS generates 15 *pre_TEK*s and 75 *TEK*s. That is why the costs of level-3 *TEK* are very much higher than those of level-2. This meets our description above.

Figure 30 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 level-3 *TEK* to encrypt data messages is the most time-consuming process because BS also needs to compute 15 *pre_TEK*s and 75 *TEK*s. The costs required are very much higher than those of using level-1 and level-2, particularly when *TEK* is longer.

Figure 31 illustrates the fact that SS on receiving a *TEK*-exchange-reply message, no matter level-1, level-2 or level-3 *TEK* is in use spends most of its time to identify the legality of BS. So, the costs of the three levels of *TEK*s are similar.

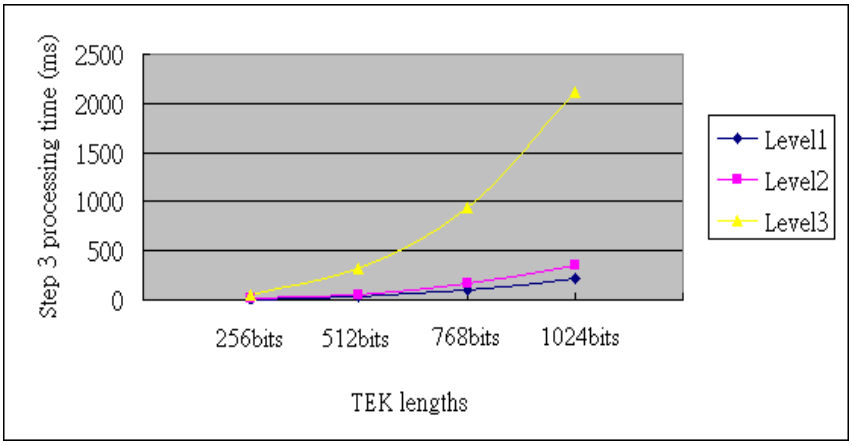

Figure 29. The times consumed by SS from the time point when it receives an authentication-reply message to the time point when it sends out a *TEK*-exchange-request message (i.e., step 3).

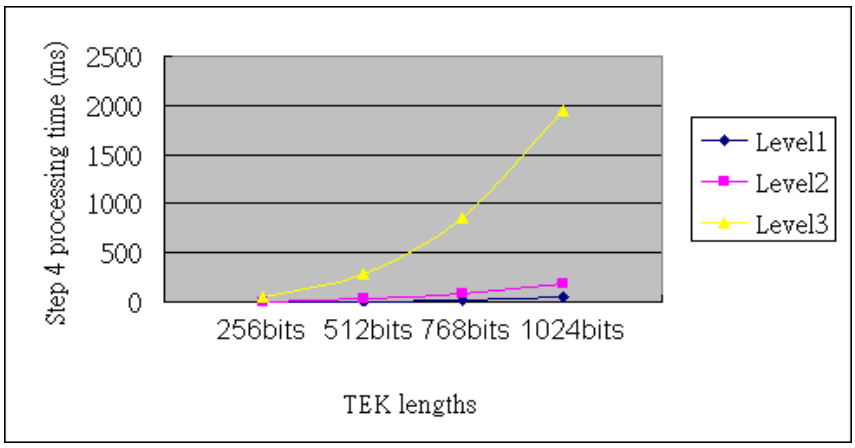

Figure 30. The times consumed by BS from the time point when it receives a *TEK*-exchange-request message to the time point when it generates a *TEK*-exchange-reply message (i.e., step 4).

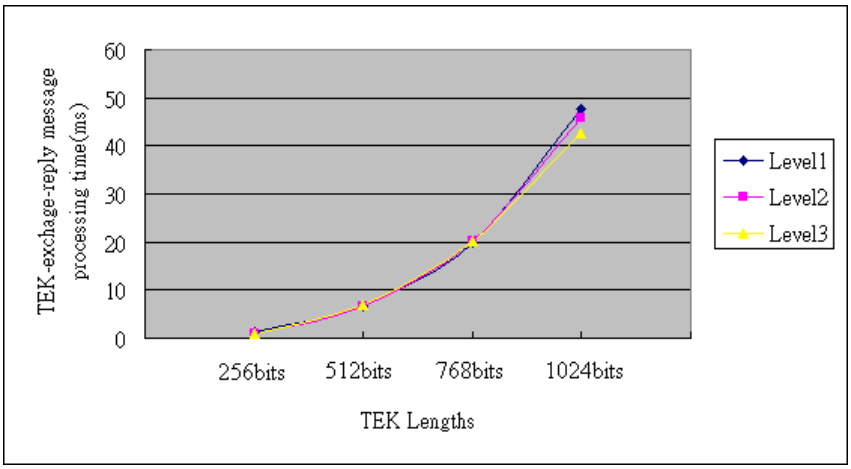

Figure 31. The times required by SS to process a *TEK*-exchange-reply message.

6.3 Results of the AP-DiHam authentication phase and pre_Data transmission phase

The results of the first experiment are shown in Figure 32 and Figure 33. Figure 32 illustrates that the process times required by different steps were functions of lengths of *AK*s. In step 1, we calculated three private keys for SS, i.e., P_{RS1}, P_{RS2} and P_{RS3} by using Diffie-Hellman PKDS algorithm. Hence, the longer the length of P_{RSi}, the more time required by the key. In steps 2 and 3,

BS on receiving an Authentication-request message generates *CSK1*, *CSK2* and *CSK3*, also with Diffie-Hellman algorithm. However, 6 *AK*s, 9 *TAK*s, 9 *TCK*s, 243 *TEK*s, 9 *NTAK*s, 9 *NTCK*s and 243 *NTEK*s are all generated after message 2 transmitted by BS, the consumed time of these keys will not be include. That is why its times required were almost the same as that of step 1. In step 4, SS on receiving an Authentication key and key reply message generated three *CSK*s. Its processing times were similar to those of step 1 since both calculated three discrete logarithm numbers. The total consumed time required from step1 through step4 is about 250ms which shows that AP-DiHam is available.

Figure 33 illustrates the processing times of steps 5~8 in which only step 5 consumed the longest times since SS had to calculate 6 *AK*s, 9 *TAK*s, 9 *TCK*s, 243 *TEK*s, 9 *NTAK*s, 9 *NTCK*s and 243 *NTEK*s. Other steps mainly validated the received message. So the times required were relatively short.

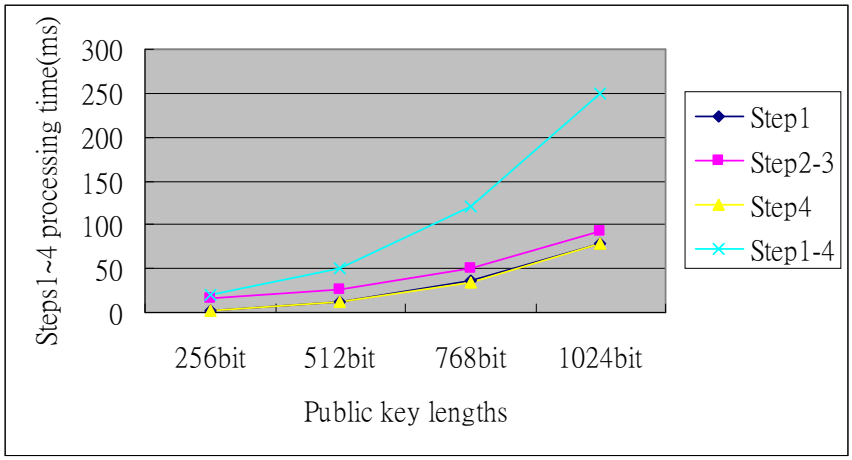

Figure 32. The times consumed by authentication phase. (i.e., Steps 1~4)

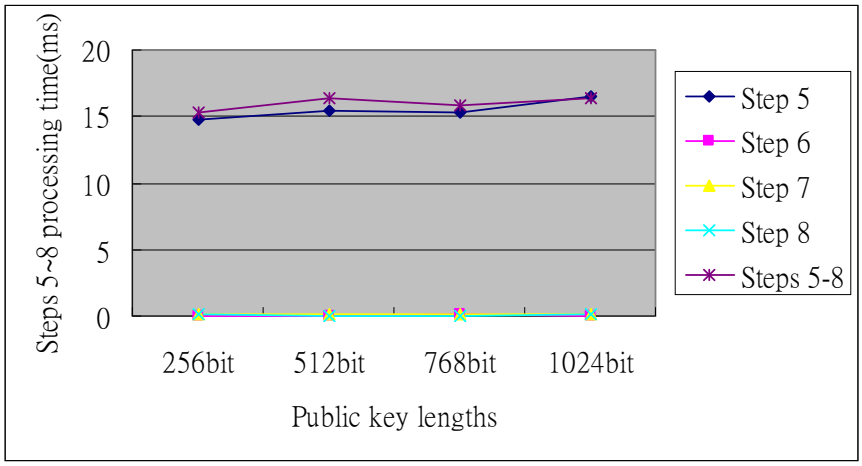

Figure 33. The times consumed by pre_Data transmission phase. (i.e., Steps 5~8)

6.4 Results of the AP-DiHam data transmission phase

Table 6. The times consumed by SS to encrypt a file. (i.e., Step 9)

	1KB	512KB	1MB	10MB
256bit	1.09ms	802.86ms	1563.11ms	15882.62ms
512bit	1.10ms	786.77ms	1598.35ms	16379.30ms
768bit	0.95ms	947.42ms	1902.83ms	19868.80ms
1024bit	0.97ms	1208.45ms	2388.60ms	25508.10ms

Table 7. The times consumed by BS to decrypt a file. (i.e., Step 10)

	1KB	512KB	1MB	10MB
256bit	0.94ms	764.80ms	1562.42ms	15852.47ms
512bit	1.06ms	793.61ms	1558.74ms	16325.97ms
768bit	0.98ms	956.93ms	1883.13ms	19789.98ms
1024bit	1.03ms	1205.81ms	2392.69ms	25148.83ms

This experiment mainly evaluated the efficiency of message encryption and decryption. File of four different sizes, as shown in Table 6 and Table 7, including 1KB, 512KB, 1MB and 10MB, were tested.

The speed of encryption/decryption about 512KB, 1MB, and 10MB data files are almost the

same. Of course, the longer of length of data, the less encryption/decryption speed. However, the encryption/decryption speed of 1 KB is about twice encryption/decryption speed of the other longer size. That is why we suggest to partition a huge message into several smaller messages before it is delivered. Furthermore, the fastest encryption/decryption speed is 1.05 MB/s ($=\frac{1KB}{0.95ms}$) and the slowest encryption/decryption speed is 0.4MB/s ($=\frac{10MB}{25508.1ms}$), they show that the two-dimensional stream cipher has high performance in encryption/decryption data message.

6.5 Comparisons

In the following, we compare the IEEE 802.16e PKMv1 and the AP-DiHam. Both have the following characteristics.

(1) They have no any CDA before SS and BS start their authentication.

(2) Only device certification is required. The user identity certification is not necessary. Any authorized machine that transmits data through wireless channels will be protected by the system.

Security and performance issues of the two system are discussed as follows.

(1) Security

(A) From security viewpoint, the main weakness of IEEE 802.16e PKMv1 is that the protection on *Cert(SS)* is insufficient. Hackers can obtain *PubKey(SS)* from *Cert(SS)* conveyed on message 2. After that, hackers can then acquire *AK* or *pre_AK* from message 3, to act as a legitimate SS. As

*AK*s are known to hackers, the *HMAC()* is less secure, and *oldTEK* and *newTEK* will also be known to hackers, consequently destroying security of the protected system. In the AP-DiHam, hackers not only need to obtain *PubKey(SS)*, but also have to get either set of the private keys *RS1*, *RS2* and *RS3* or *RB1*, *RB2* and *RB3*. It is not easy to obtain private key from public key [10]. If *PubKey(SS)* is know to hackers, the AP-DiHam is still secure system.

(B) IEEE 802.16e PKMv1 suffers forgery BS attacks, even though *PubKey(SS)* is not known to hackers. However, the AP-DiHam can effectively protect against the forgery BS attacks by certificating *Certfun(CSK1, CSK2, PubKey(SS))* transmitted by BS. Mutual authentication which is lacked by IEEE802.16e PKMv1 is fully supported by the AP-DiHam.

(C) In IEEE 802.16e PKMv1, when *TEK*s are transmitted from BS to SS through wireless channels, the encrypted *TEK*s may be caught and decrypted by hackers, causing the following data to be transmitted in a unsafe status. In the AP-DiHam, all the security keys, such as *AK*s, *TEK*s, and *NTEK*s, are generated by SS and BS individually and independently, i.e., the keys do not need to be transmitted through wireless channels from BS to SS. Hence, the AP-DiHam is more secure than IEEE 802.16e PKMv1.

(D) In data transmission phase, data was encrypted by DES-CBC mode in IEEE 802.16e PKMv1, and encrypted by two-dimensional stream cipher in the AP-DiHam. Security levels of both are high. But the *TEK* key and initialization vector (IV for short) of PKMv1 are more easily cracked,

than *TEK*s and *NTEK*s of the AP-DiHam done by hackers. Hence, the ciphertext of IEEE 802.16e PKMv1 can be more easily decrypted into plaintext than that of the AP-DiHam can. The AP-DiHam has higher security than that of the IEEE 802.16e PKMv1 in data transmission phase. From above analysis, we can conclude that the AP-DiHam is more secure than the IEEE 802.16e PKMv1.

(2) Performance

(A) From performance viewport, in the authentication phase, due to the generation of public keys and common secrete keys, the AP-DiHam consumes longer computation time than that of IEEE 802.16e PKMv1 does.

(B) In the data transmission phase, although data encrypted/decrypted by both two-dimensional stream cipher and DES-CBC mode are efficiently performed, it is obviously that DES-CBC mode needs more calculations and conger calculation time than two-dimensional stream cipher does. Therefore, the performance of two-dimensional stream cipher is better than that of the DES-CBC.

A wireless communication system often consumes most of it time to deliver data. Hence, the transmission phase is longer than authentication phase and pre_Data transmission phase. That is why the AP-DiHam has higher performance than that of IEEE 802.16e PKMv1.

Chapter 7 Conclusions and Future Research

If a wireless system provides no any CDA before SS and BS start their communication, BS can not effectively certify whether the user is legitimate or not. Any authorized machine that transmits data through wireless channels should be protected by the system. In PKMv1 authentication phase, SS lacks a security mechanism to authenticate BS. In other words, PKMv1 cannot effectively protect a WiMAX system from forgery BS attacks, and SS and BS should take more computations to set up the security mechanism for their data transmission, consequently reducing the system performance.

In this article, we have developed an authentication mechanism to effectively improve WiMAX facility authentication by employing a system which integrates the DH-PKDS and the PKMv1, and in which a two-way authentication instead of the unidirectional authentication of PKMv1 is used. When the authentication fails, the message will be dropped so as to avoid wireless facilities from being attacked, particularly a Dos/DDoS attack and a man-in-the-middle attack, launched by a fake BS or SS during the network facility authentication phase. We also designed data carriers to protect CDA and used two-dimensional stream cipher to encrypt and decrypt transmitted data.

Basically, this study only modifies the PKMv1 and integrates the modified one with the DH-PKDS as the AP-DiHam. In 802.16e standards, the more secure and well-organized PKMv2 has been released [3]. Our opinion is the AP-DiHam can also be applied to the PKMv2 to further

enhance its security. The CDA is one effective method to improve the security level and performance of the wireless system. How to design a well defined CDA scheme so that the mutual authentication mechanism can be effectively performed. We would also like to derive a reliability model [29] 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 [30]. The AP-DiHam can be also applied to maintain and improve hand-off security. The authors of [24] 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 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

[1] 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.

[2] M. Barbeau, "WiMax/802.16 threat analysis," ACM International Workshop on Quality of Service & Security in Wireless and Mobile Networks, 2005, pp. 8–15.

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

[4] 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.

[5] M. S. Rahman and S. Kowsar, "WiMAX Security Analysis and Enhancement," Proceedings of the International Conference on Computer and Information Technology, 2009, Dhaka, pp.679-684

[6] T. Han, N. Zhang, K. Liu, B. Tang and Y. Liu "Analysis of Mobile WiMAX Security: Vulnerabilities and Solutions," the International Conference on Mobile Ad Hoc and Sensor Systems, 2008, pp.828-833.

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

[8] F.Y. Leu, Y. F. Huang and C. H. Chiu, “Improving security levels of IEEE802.16e authentication by Involving Diffie-Hellman PKDS,” The International Conference on Complex, Intelligent and Software Intensive System, 2010, pp.391-397

[9] W. Diffie and M.E. Hellman, “New Directions in Cryptography,” IEEE Transactions on Information Theory, vol. 22, June, 1976, pp.638-654.

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

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

[12] Y. Zeng, and J.F. Ma, Pseudo-random number generators based on evolutionary algorithm, in: Proc. The 2003 Congress on Evolutionary Computation, Vol 3 (Canberri, Austrialia, 2003) 1662-1668.

[13] P. Hellekalek, Good random number generator are (not so) easy to find, in: J. Mathematics and Computers in Simulation, Vol. 46 (Elsevier, Europe, 1998) 485-505.

[14] S. Brands and R. D. Gill, Cryptography, statistics and pseudo-randomness. Part I, J. Probab. Math. Statist. 15(1995), 101-114.

[15] J. Kelsey, B. Schneier and N. Ferguson, Yarrow-160: notes on the design and analysis of the yarrow cryptographic pseudo-random number generator, in: Sixth Annual Workshop on Selected Areas in Cryptography (Springer-Verlag, UK, 1999)

[16] Yi-Fung Huang, Chu-Hsing Lin and Kun-Li Wen, "A Pseudo-random Number Generator Based on Grey System Theory", Far East Journal of Mathematical Sciences, vol. 35, Issue 1, pp. 1 ~ 17, September 2009

[17] X. Yang, H.H. Chen, X. Du and M. G., ” Stream-based cipher feedback mode in wireless error channel, ” IEEE Transactions on Volume 8, Issue 2, 2009, pp. 622 – 626.

[18] Pal, J.K. Mandal, J.K. Dept, “A random block length based cryptosystem through multiple cascaded permutation-combinations and chaining of blocks” The International Conference on Industrial and Information Systems, 2009, pp.391-397

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

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

[21] S. Aboud, M. A. AL-Fayoumi, M. AL-Fayoumi and H.S. Jabbar,“ An Efficient RSA Public Key Encryption Scheme” the International Conference on Information Technology: New Generations, USA. 2008, pp.127-130.

[22] S. Aboud, “Baghdad Method for Calculating Multiplicative Inverse,” International Conference on Information Technology, Las Vegas, Nevada, USA. 2004, pp.816-819.

[23] D. Johnston and J. Walker, “Overview of IEEE 802.16 security,” IEEE Security &Privacy Magazine, vol. 2, no. 3, May–June 2004, pp. 40–48.

[24] T. Kleinjung, K. Aoki, J. Franke, A.K. Lenstra, E. Thomé, J.W. Bos, P. Gaudry, A. Kruppa, P.L. Montgomery, D.A. Osvik, H.T. Riele, A. Timofeev and P. Zimmermann, “ 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

[25] Gordon. J. "Strong RSA keys", Electronics Letters, June 1984, pp. 514-516

[26] P. Bhattacharya, M. Debbabi and H. Otrok, "Improving the Diffie-Hellman Secure Key Exchange," International Conference on Wireless Networks, Communications and Mobile Computing, 2005, pp.193-197.

[27] P. Neves, F. Fontes, J. Monteiro, S. Sargento and T.M. Bohnert, "Quality of Service Differentiation Support in WiMAX Networks," International Conference on Telecommunications, 2008, pp.411-415.

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

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

[30] D. Shi and C. Tang, "A Fast Handoff Scheme Based on Local Authentication In Mobile Network," International Conference on ITS Telecommunications, 2006, pp.1025-1028.

[31] L. Adleman "A Subexponential Algorithm for the Discrete Logarithm Problem with Applications to Cryptography", the Foundations of Computer Science Symposium, 1979, pp. 55-60.