東海大學資訊工程研究所

碩士論文

指導教授：呂芳懌 博士

在無線網路環境中使用資料連結核心建立安全的通訊

A Secure Communication over Wireless Environments by Using a Data Connection Core

研究生：尉可忠

中華民國 102 年 1 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 尉 可 忠 所提之論文

在無線網路環境中使用資料連結核心建立安全的通訊

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 羅崇哲 簽章

委 員 陳金鈴

王七二教

[illegible]

指 導 教 授 [illegible] 簽章

中華民國 102 年 1 月 16 日

Table of Contents

中文摘要

近年來，無線網路如 Wi-Fi 及 3G 在世界上已被廣泛的裝備及普遍的使用。人們在現代的無線網路以手持智慧型裝置可以方便地存取網際網路。然而，當人們正享受的使用無線系統時，網路安全是一個關鍵的挑戰，因為無線的訊號，無論是否有被加密，都可能被駭客惡意的截取。在分析及解密後，駭客非法的取得或偷竊訊號內之重要訊息，如信用卡卡號或帳號密碼。目前的網路安全機制是利用 SSL 及 IPsec 來保護這些傳遞中的訊息。然而，這兩種安全協定在金鑰交換和加解密的步驟中仍有其缺點。本研究中，我們提出一種安全的通訊系統 Wireless Security System with Data Connection Core(簡稱 WiSDC)，其中有兩種安全構想，包含對稱金鑰交換程序及二維串流加密機制。前者在 Data Connection Core(簡稱 DCC)中採用隨機變數及關聯金鑰以產生內部金鑰，藉以加強金鑰交換程序的安全等級。在此，DCC 是一組在嚴謹的無線系統中使用者登錄相關資料後所創造的隨機亂數，這些隨機亂數在無線系統中只有使用者及 AAA 伺服器知道。後者，以二維串流加密機制為例，則引用兩種數學運算，包括互斥運算及二進制加法，並使用兩個隨機亂數來加密明文以便有效的保護明文。WiSDC 同時採用隨機亂數產生器以反饋的方式產生更複雜的金鑰及加密密文。實驗的結果顯示 WiSDC 對於無線環境中可以有效地保護傳輸訊息。分析的結果指出 WiSDC 比 SSL 及 IPsec 有更高的安全等級及執行效率。

關鍵字 ：Wi-Fi，3G，Data Connection Core，內部金鑰，隨機亂數產生器，二維串流加密

Abstract

Recently, wireless networks, such as Wi-Fi and 3G, have been widely equipped and popularly used in the world. People holding smart devices can conveniently access the Internet services through modern wireless networks. However, when people are enjoying using wireless systems, network security has been a crucial challenge because wireless messages, encrypted or unencrypted, may be maliciously intercepted by hackers. After analyzing and/or decrypting the messages, hackers can illegally capture or steal important information, such as credit card numbers or usernames/passwords, carried in the messages. Currently, SSL and IPsec are utilized to protect the delivery of these types of information. However, each of the two security protocols has its own drawbacks both in their key exchange and message encryption/decryption processes. To solve these drawbacks, in this paper, we propose a secure communication system, named the Wireless Security System with Data Connection Core (WiSDC for short), which consists of two security schemes, including a symmetric key exchange process and a two-dimensional stream cipher mechanism. The former employs random numbers and the connection keys contained in the Data Connection Core (the DCC for short) to generate internal keys, through which the security level of the key exchange process can be enhanced. Here, the DCC is a set of random numbers created when the underlying user registers himself/herself with the wireless system being considered, and the random numbers are only known to the user and AAA server of the wireless system. The latter, i.e., the two-dimensional stream cipher mechanism, invokes two operators, including exclusive-or $\oplus$ and binary adder $+_2$ operators, and two Pseudo Random Number Sequences (PRNSs) to encrypt plaintext so as to well protect the plaintext. The WiSDC also adopts a pseudo random number generator, which feeds back keys generated in current stage as a part of the inputs of the next stage, to produce more complicated keys for data encryption. Experimental results show that the WiSDC can effectively protect transmitted messages for wireless environments. The analytical results indicate that the WiSDC has higher security level and execution efficiency than those of the SSL and IPsec.

Keywords：Wi-Fi, 3G, Data Connection Core, Internal Key, Pseudo Random Number Generator, Two-dimensional Stream Cipher

List of Figures

List of Tables

Chapter 1 Introduction

Nowadays, mobile devices have been widely used in modern network environments. People often communicate with others and access Internet services through mobile devices. But wireless communication has its own drawbacks, one of which is that hackers can easily intercept the messages delivered via wireless channels. That is why network security has been one of the important issues in the study of wireless systems.

Today, e-commerce is popular in the world. Many people like to purchase something through networks. However if authenticated users wish to transmit important data, such as credit card numbers or usernames/passwords of a system, a security mechanism that can effectively protect the important information is required. Security Socket Layer (SSL for short) and Internet Protocol Security (IPsec for short) are two security protocols commonly used in the Internet application services.

However, SSL has two disadvantages in protecting the delivery of these types of information. One is that its key exchange process comprises six steps, in which four may expose important keys in the air [1], implying hackers have four opportunities to capture important information of a session. The second is that only one master-secret key is utilized to encrypt exchanged keys. Therefore, the generated ciphertext may be cracked relatively more easily by using brute-force attacks compared to those using multiple keys [2].

The disadvantages of IPsec are similar to those of the SSL. The number of its key exchange steps is six in the main mode, and only one encryption key, called session key [3], is deployed to encrypt transmitted messages.

To solve these two security problems, in this study we propose a symmetric key exchange system, called the Wireless Security System with Data Connection Core (WiSDC for short), which as a mutual authentication mechanism employs the data connection core (the DCC for short) as its security base to preserve the Secrecy, Authenticity, Integrity and Nonrepudiation characteristics of those transmitted messages, where the DCC is a set of random numbers

created when the underlying user registered himself/herself with the wireless system being considered, and the random numbers are only known to the user and AAA home server of the wireless system. The WiSDC also creates and invokes three specific mechanisms to increase the security level of its own. The first is producing internal keys from a part of the parameters, called connection keys, collected in the DCC and a set of other random numbers. We also derive parameters, named communication keys, from the internal keys. The communication keys, rather than the DCC, are transmitted through the air to protect the DCC from being known to hackers. The second is reducing key exchange steps to lower the probability of important information being captured. The number of its key exchange steps is four, implying that the opportunity for key information being stolen by hackers in the wireless environment is lower than when the SSL and IPsec is employed. The third is adopting a two-dimensional stream cipher technique to encrypt/decrypt the transmitted messages so as to promote the WiSDC to a high security and high efficiency system.

A part of this paper has been published [4]. In this version, we extend several new concepts and encryption techniques.

The rest of this thesis is organized as follows. Chapter 2 describes the background and related work of this study. Chapter 3 introduces the WiSDC architecture. The security analysis and simulation of the proposed system are presented and discussed in Chapter 4 and 5, respectively. Section 6 concludes this thesis and outlines our future studies.

Chapter 2 Background and Related Work

This chapter describes the background and related work of this study.

2.1 Background

2.1.1 Stream cipher Encryption

In cryptography, due to generating the same cipher streams between sender and receiver before the ciphertext sent by the sender can be accurately decrypted by the receiver, a stream cipher [5], encrypting a message on a bit-by-bit base, basically is a symmetric key cipher method. Its encryption/decryption speed is faster than that of a traditional encryption mechanism, such as the block cipher [6] (e.g., DES and AES). However, a stream cipher has three important security flaws, including that only the exclusive-or operation is utilized, the same key stream is used throughout the encryption/decryption process of a session, and the random numbers deployed are not very complicated so as unable to effectively protect the transmitted messages. The three flaws may attract certain types of attacks, such as brute-force attacks [7] and hijacking attacks [8]. Consequently encrypted information may be revealed. Generally, the pseudo random number generators (PRNGs for short) of both the sender side and the receiver side are triggered by the same secret key K to generate the same streams for encrypting and decrypting messages.

However, the security level of a ciphertext strongly relies on the quality of the randomness, complexity, and lengthy periods of the produced random number [9]. Basically, how to design a random number generator to generate high-quality random number sequences is a technical challenge [10]. A well-designed random number generator must avoid producing the same random number sequences each time it is invoked. Otherwise, the security level of the ciphertext will be relatively lower.

2.2 Related Work

This section describes the SSL and IPsec.

2.2.1 Secure Socket Layer (SSL)

SSL is a security protocol, developed to secure the established Internet connection either between a web server and a client browser [11] or between a host and a gateway [12].

The secure hypertext transfer protocol (i.e., HTTPS) [13] adopts SSL to encrypt important information when people purchase something through e-commerce. The handshake process of SSL has six steps [14] which are listed in the Appendix A of this paper (the master secret keys are generated on both ends of a connection in the fourth step). Besides those mentioned above, SSL has two other disadvantages. One is that hackers may intercept the X.509 certificate transmitted through the wireless channels, and use the certificate to mimic a legal user to perform the consequent authentication. The second is that when one visits a website to purchase something, the website does not know who the visitor is. When malicious people capture a credit card number and then buy something with it through the websites of business stores, if the legal user can show that the transaction is not submitted by himself/herself, the business stores will incur financial loss.

2.2.2 Internet Protocol Security (IPsec)

IPsec as a network-layer security protocol authenticates and encrypts IP packets transmitted through the Internet. It has been used to protect the connections established between host and host, gateway and host, and gateway and gateway [15] of a system.

This protocol adopts the Internet Key Exchange (IKE for short) [16] as its key-exchange protocol, which first builds an IKE security association (SA for short), and then creates an IPsec SA through the channel protected by the IKE SA. IPsec typically supports two authentication methods, pre-share keys and digital signatures. With the pre-share keys, the administrator produces a key or a password string for each IPsec device. With the digital signature, a certificate identifies a IPsec site. Basically, two IPsec endpoints have to trust each other if a Certification Authority (CA for short) that they trust has signed their certificates. The two disadvantages of IPsec are mentioned above.

Besides, security can also be enhanced by using linguistic processes [17,18, 19]. Ogiela and

Ogiela [17] adopted mathematical linguistic methods to create secret sharing threshold algorithms. Leu and Ko [19] utilized critical values to cluster definition briefs which can also be applied to improve the detection accuracy of identifying who the possible hacker is in an enclosed environment when internal attack occurs [20].

Chapter 3 System Architecture

The WiSDC has six features, including (1) verifying whether a user is a legitimate one or not by consulting the AAA server; (2) generating internal keys and communication keys to prevent the DCC from being sent through the wireless channels; (3) preserving the Secrecy, Authenticity, Integrity and Nonrepudiation characteristics for those transmitted messages; (4) the number of key exchange steps is only four; (5) calculating 320 Traffic Encryption Keys (*TEK*s for short) and 319 new Traffic Encryption Keys (*NTEK*s for short) by using a feedback control process [21]; (6) encrypting plaintext into ciphertext by using a two-dimensional stream cipher technique.

Figure 1 shows the network topology of the WiSDC, in which wireless users can transmit messages through different Internet Service Providers (ISPs for short) to access the Internet services, implying that the WiSDC can adapt to different wireless environments, e.g., ISP-1 provides a WiFi platform, and ISP-2 offers 4G services. A user of the WiSDC can roam in this heterogeneous environment. Figure 2 illustrates the WiSDC architecture which consists of wireless users, Authenticators of different ISPs, and a home server. The home server consists of the AAA server and other servers, e.g., radius server and log server.

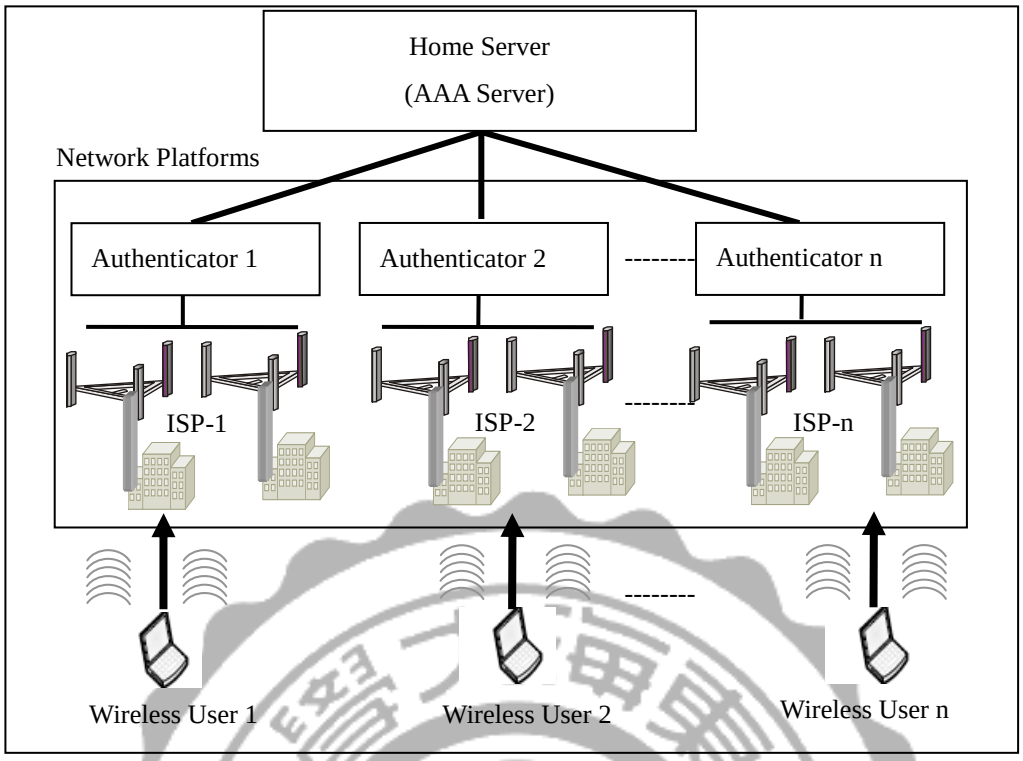

Figure 1. The network topology of the WiSDC

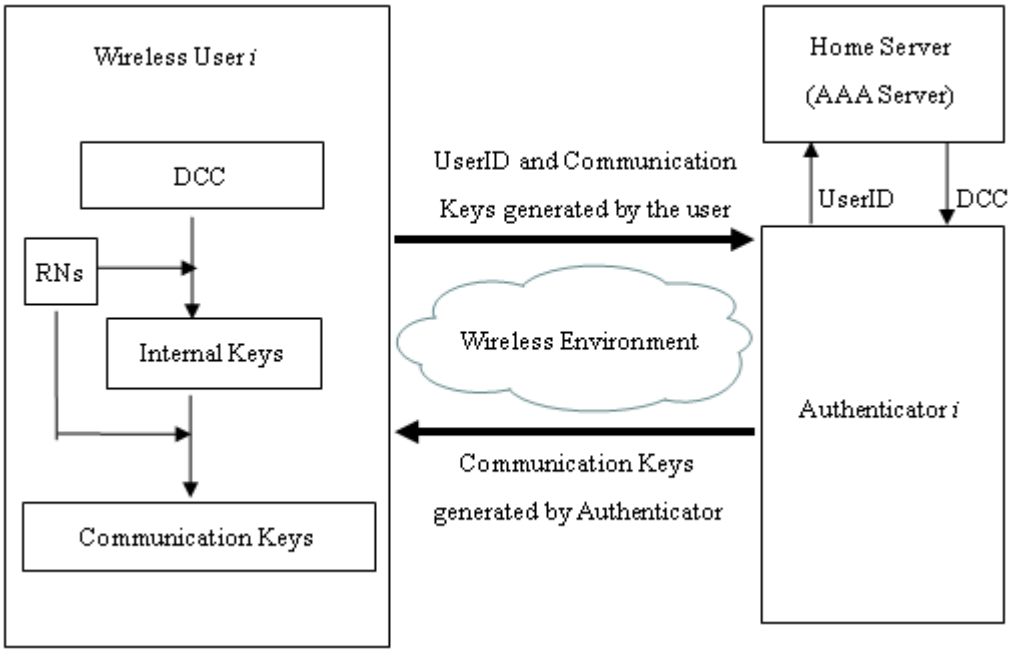

Figure 2. The WiSDC architecture (RNs stand for Random numbers)

3.1 The Data Connection Core

In the WiSDC, the high security level and the robust key exchange process are, respectively, achieved and developed by using the DCC.

The DCC consists of five parameters, including *UserID*, k_i, k_1, k_2, and k_3, which are stored both in a user device and AAA server when the user registers himself/herself with the AAA server, and in which k_i, k_1, k_2, and k_3 are the connection keys of the DCC where k_i is the identification key and k_1, k_2, and k_3 are message encryption keys. In a user device, the WiSDC, as shown in the left rectangle of Figure 2, named wireless user *i*, employs the DCC and four random numbers, including ψ_1, ψ_2, ψ_3 and ψ_4, to produce seven internal keys which are used to further generate four communication keys for later message encryption and decryption. Internal keys are keys utilized only internally in the user device or the authenticator without being delivered through the wireless channel. All the parameters of the WiSDC are the same length.

3.2 Parameters and functions

The parameters and functions utilized by the WiSDC are defined as follows.

3.2.1 The parameters

The parameters used by the WiSDC are defined and summarized below.

(1) *UserID* : the identity of a user.

(2) k_i, k_1, k_2, k_3 : the connection keys of the DCC.

(3) ψ_i ,i=1,2,3,4 : the random numbers generated by the user.

(4) a_i, i=1,2,3,4 : the communication keys generated by the user and transmitted to Authenticator through wireless channels.

(5) b_i, i=0,1,2,3,4,5, and c_0 : the seven internal keys internally generated and used by a user device and Authenticator themselves without sending them through wireless channels.

(6) ϕ_i ,i=1,2,3,4 : the random numbers generated by Authenticator.

(7) c_i, i=1,2,3,4 : the communication keys generated by the Authenticator and transmitted to the user through wireless channels.

(8) T_{nonce} : the timestamp of current time.

(9) TEK_i, $1 \leqq i \leqq 320$: the first set of traffic encryption keys used to encrypt transmitted data

messages.

(10) $NTEK_i$, $1 \leqq i \leqq 319$: the second set of traffic encryption keys created to encrypt transmitted data messages.

3.2.2 The functions

The functions employed by the WiSDC are defined as follows.

(1) Exclusive-or operator $\oplus$:

Encryption : $c = p \oplus K$,

Decryption : $p = c \oplus K$.

(2) Binary-adder $+_2$:

Encryption : $c = p +_2 K$, where p and K undergo binary addition, and ignore the carry generated by the addition of the most significant bits;

Decryption : $p = c -_2 K = \begin{cases} c - K, & if \ c \geq K \\ c + \overline{K} + 1, & if \ c < K \end{cases}$,

where $-_2$ denotes the binary subtraction, and $\overline{K}$ is the one's complement of K.

(3) $HMAC(k)$: a Hash-based message authentication code which is generated by performing a hash function on both a secret key k and a transmitted message to ensure the certification and integrity of this message.

Example 1 : If there is a message, $OP\text{-}code|T_{nonce}|UserID|a_1|a_2|a_3|a_4|HMAC((b_1 \oplus b_2)+_2 b_3)$, which is transmitted from a user to Authenticator, then $HMAC((b_1 \oplus b_2)+_2 b_3)$ is the authentication code generated by invoking a hash function to encrypt the plaintext, $OP\text{-}code|T_{nonce}|UserID|a_1|a_2|a_3|a_4$, with the key, $(b_1 \oplus b_2)+_2 b_3$.

(4) $RHSEXOR(X, Y) = RHS(X) \oplus Y$: $RHS(X)$ and Y are of the same size by truncating X's most significant bits where the length of X is longer than that of Y.

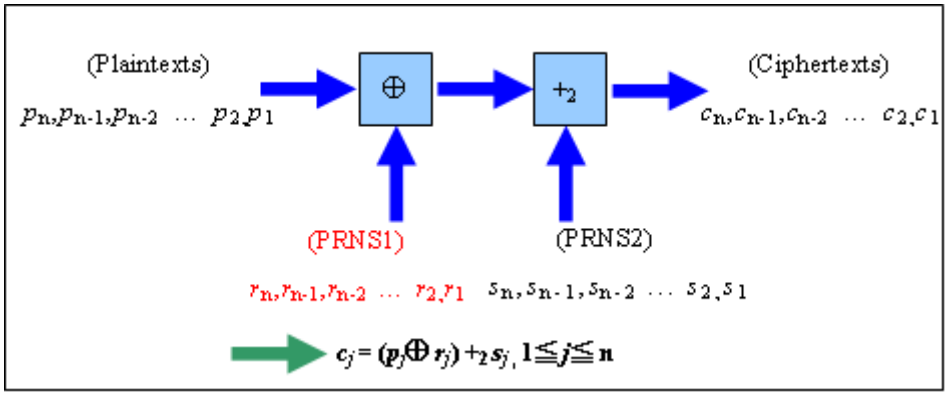

Figure 3. The flow chart of a two-dimensional stream cipher technique

In the WiSDC, plaintext as shown in Figure 3 is encrypted with two pseudo random number sequences (PRNS for short), e.g., PRNS1 and PRNS2, by using $\oplus$ and $+_2$. We call a ciphering approach that encrypts messages with this method a two-dimensional stream cipher technique.

3.3 The Prior Activities

The prior activities of a wireless communication are as follows. The user

(1) generates random parameters, ψ_1, ψ_2, ψ_3 and ψ_4;

(2) sequentially derives communication keys a_1~a_4, and internal keys, b_0~b_5, and c_0, from the four connection keys of the DCC where the sequence, in which the parameters are generated, is as follows.

$$a_1 = \{[k_i \oplus k_1] +_2 \psi_1] \oplus (k_1 +_2 k_2)\} +_2 (k_2 \oplus k_3),$$

$$b_0 = [(\psi_1 \oplus k_i) +_2 (\psi_1 \oplus k_1)] \oplus [(\psi_1 +_2 k_2) \oplus (\psi_1 +_2 k_3)],$$

$$c_0 = \{[(k_i \oplus b_0) +_2 k_1] \oplus (\psi_1 +_2 k_2)\} \oplus (b_0 +_2 k_3),$$

$$b_1 = [(\psi_1 \oplus k_i) +_2 (b_0 \oplus k_2)] \oplus (c_0 +_2 k_1),$$

$$b_2 = [(\psi_1 +_2 k_1) \oplus (b_1 +_2 k_3)] +_2 (c_0 \oplus k_2),$$

$$b_3 = [(\psi_1 \oplus k_2) +_2 (b_2 \oplus k_i)] \oplus (c_0 +_2 k_3),$$

$$b_4 = (b_3 \oplus c_0) +_2 [(b_1 +_2 (b_2 \oplus \psi_1)],$$

$$b_5 = (b_4 +_2 c_0) \oplus [b_3 \oplus (b_1 +_2 \psi_1)],$$

$a_2 = [(b_0 \oplus \psi_2) +_2 (b_1 \oplus b_2)] \oplus (b_3 +_2 b_4)$,

$a_3 = [(c_0 \oplus \psi_3) \oplus (b_2 +_2 b_3)] +_2 (b_4 \oplus b_5)$,

$a_4 = [(b_4 \oplus \psi_4) +_2 (b_5 \oplus b_3)] \oplus (b_1 +_2 b_2)$.

Since these keys are sequentially generated, i.e., b_0 is generated before c_0, then b_0 and c_0 are employed to generate b_1. After that, b_1 is invoked to generate b_2, etc, a parallel cracking method does not work.

(3) Defines the *OP-code*s listed in Table 1.

Table 1. The definitions of OP-codes employed

OP-code	Process	Explanation
1	Authentication request	Sent to Authenticator by user
2	Authentication reply	Sent to user by Authenticator
3	Data transmission	Sent to Authenticator by user
4	Data receiving	Sent to user by Authenticator
0, 5-256	Reserved	

3.4 The key exchange process

The sequence chart of the WiSDC is illustrated in Figure 4, in which steps 1~4 are the key exchange process [22], and steps 5 and 6 are the message encryption/decryption process.

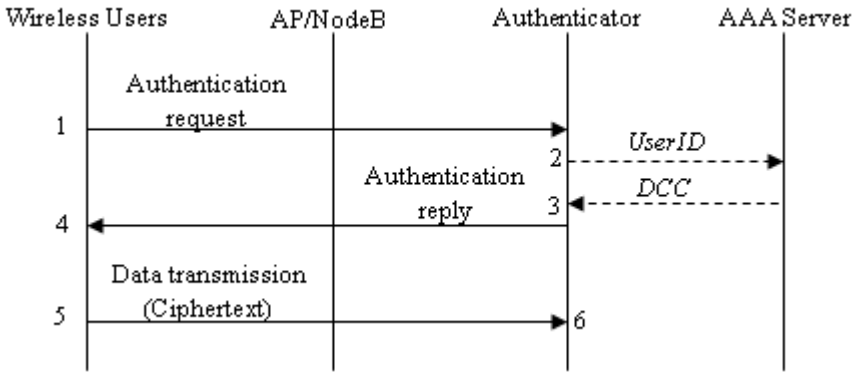

Figure 4. The sequence chart of key exchange and message encryption/decryption processes

Step 1: User records the status "authentication request" in its *OP-code* field, and sends an authentication request message, denoted by message 1, to Authenticator. The format of this message is

$OP\text{-}code|T_{nonce}|UserID|a_1|a_2|a_3|a_4|HMAC((b_1 \oplus b_2) +_2 b_3)$

After sending this message, the user sets its current status to "authentication reply".

Step 2: On receiving message 1, Authenticator checks to see whether $T_{receive} - T_{nonce} > \triangle T$ where $T_{receive}$ is the time point when message 1 is received and $\triangle T$ is a predefined threshold. If yes, implying that this is an replay attack, it discards this message and stops the key exchange process. Otherwise, Authenticator delivers *UserID* to the AAA server. The AAA server replies Authenticator with the *UserID*'s DCC.

Authenticator retrieves the random numbers, ψ_1, ψ_2, ψ_3 and ψ_4, implicitly carried in message 1 (i.e., in a_1, a_2, a_3, and a_4, respectively) by invoking the following process. It

(1) retrieves ψ_1 by decoding a_1 with the DCC. Let

$$A_1 = [a_1 -_2 (k_2 \oplus k_3)] \oplus (k_1 +_2 k_2),\ A_2 = (a_1 +_2 \overline{k_2 \oplus k_3} +_2 1) \oplus (k_1 +_2 k_2).$$

$$\psi_1 = \begin{cases} \{[a_1 -_2 (k_2 \oplus k_3)] \oplus (k_1 +_2 k_2)\} -_2 (k_i \oplus k_1), if\ a_1 \geq (k_2 \oplus k_3)\ and A_1 \geq (k_i \oplus k_1); \\ \{[a_1 -_2 (k_2 \oplus k_3)] \oplus (k_1 +_2 k_2)\} +_2 \overline{k_i \oplus k_1} +_2 1, if\ a_1 \geq (k_2 \oplus k_3)\ and A_1 < (k_i \oplus k_1); \\ [a_1 +_2 \overline{k_2 \oplus k_3} +_2 1) \oplus (k_1 +_2 k_2)] -_2 (k_i \oplus k_1), if\ a_1 < (k_2 \oplus k_3)\ and A_2 \geq (k_i \oplus k_1); \\ [a_1 +_2 \overline{k_2 \oplus k_3} +_2 1) \oplus (k_1 +_2 k_2)] +_2 \overline{k_i \oplus k_1} +_2 1, if\ a_1 < (k_2 \oplus k_3)\ and A_2 < (k_i \oplus k_1). \end{cases}$$

(2) generates its seven internal keys by sequentially invoking the following equations.

$$b_0 = [(\psi_1 \oplus k_i) +_2 (\psi_1 \oplus k_1)] \oplus [(\psi_1 +_2 k_2) \oplus (\psi_1 +_2 k_3)]$$

$$c_0 = \{[(k_i \oplus b_0) +_2 k_1] \oplus (\psi_1 +_2 k_2)\} \oplus (b_0 +_2 k_3),$$

$$b_1 = [(\psi_1 \oplus k_i) +_2 (b_0 \oplus k_2)] \oplus (c_0 +_2 k_1),$$

$$b_2 = [(\psi_1 +_2 k_1) \oplus (b_1 +_2 k_3)] +_2 (c_0 \oplus k_2),$$

$$b_3 = [(\psi_1 \oplus k_2) +_2 (b_2 \oplus k_i)] \oplus (c_0 +_2 k_3),$$

$$b_4 = (b_3 \oplus c_0) +_2 [(b_1 +_2 (b_2 \oplus \psi_1)],$$

$$b_5 = (b_4 +_2 c_0) \oplus [b_3 \oplus (b_1 +_2 \psi_1)],$$

(3) acquiring ψ_2 by using five of the seven internal keys, i.e., b_0~b_4, to decode a_2 where

$$\psi_2 = \begin{cases} \{[a_2 \oplus (b_3 +_2 b_4)] -_2 (b_1 \oplus b_2)\} \oplus b_0, if\ [a_2 \oplus (b_3 +_2 b_4)] \geq (b_1 \oplus b_2); \\ \{[a_2 \oplus (b_3 +_2 b_4)] +_2 \overline{b_1 \oplus b_2} +_2 1\} \oplus b_0, if\ [a_2 \oplus (b_3 +_2 b_4)] < (b_1 \oplus b_2). \end{cases}$$

(4) acquiring ψ_3 by using b_2~b_5 and c_0 to decode a_3 where

$$\psi_3 = \begin{cases} \{[a_3 -_2 (b_4 \oplus b_5)] \oplus (b_2 +_2 b_3)\} \oplus c_0, if\ a_3 \geq (b_4 \oplus b_5); \\ \{[a_3 +_2 \overline{b_4 \oplus b_5} +_2 1] \oplus (b_2 +_2 b_3)\} \oplus c_0, if\ a_3 < (b_4 \oplus b_5). \end{cases}$$

(5) acquiring ψ_4 by using b_1~b_5 to decode a_4 where

$$\psi_4 = \begin{cases} \{[a_4 \oplus (b_1 +_2 b_2)] -_2 (b_5 \oplus b_3)\} \oplus b_4, if\ [a_4 \oplus (b_1 +_2 b_2)] \geq (b_5 \oplus b_3); \\ \{[a_4 \oplus (b_1 +_2 b_2)] +_2 \overline{b_5 \oplus b_3} +_2 1\} \oplus b_4, if\ [a_4 \oplus (b_1 +_2 b_2)] < (b_5 \oplus b_3). \end{cases}$$

Authenticator verifies whether message 1 is issued by a legal user by checking to see whether $HMAC((b_1 \oplus b_2) +_2 b_3)_r = HMAC((b_1 \oplus b_2) +_2 b_3)_c$ or not where the subscript c (r) represents that the *HMAC*() is calculated by itself (retrieved from message 1). If not, Authenticator discards the fake message and stops the key exchange process. Otherwise, it goes to Step 3.

Step 3: Authenticator retrieves four random numbers ϕ_1, ϕ_2, ϕ_3 and ϕ_4 as the dynamic keys from its internal random number table. In this table, random numbers are generated and updated periodically. After that, Authenticator

(1) generates four communication keys c_1, c_2, c_3 and c_4 by invoking the following equations that respectively contain ϕ_1, ϕ_2, ϕ_3 and ϕ_4;

$$c_1 = [(\phi_1 \oplus \psi_2) +_2 b_1] \oplus (c_0 +_2 b_5);$$
$$c_2 = [(\phi_2 \oplus \psi_3) +_2 b_2] \oplus (\psi_2 +_2 b_1);$$
$$c_3 = [(\phi_3 \oplus \psi_4) +_2 b_3] \oplus (\psi_3 +_2 b_2);$$
$$c_4 = [(\phi_4 \oplus b_5) +_2 b_4] \oplus (\psi_4 +_2 b_3).$$

(2) sends an authentication reply message, denoted by message 2, to the user. The format of this message is

$$OP\text{-}code|c_1|c_2|c_3|c_4|HMAC((\phi_2 \oplus \phi_3) +_2 \phi_4),$$

in which *OP-code* contains current status of the key exchange process, i.e., authentication reply. After delivering this message, Authenticator sets its current status to “data transmission”;

(3) generates *TEKs* and *NTEKs*, where

$$TEK_{(i-1)\times 80+(j-1)\times 20+(k-1)\times 4+l} = [(\phi_i +_2 c_j) \oplus b_k] +_2 \psi_l;\ 1 \leq i, j, l \leq 4, 1 \leq k \leq 5$$

$$NTEK_i = [(TEK_i \oplus NTEK_{i-1}) +_2 (k_1 \oplus d_{i-1})] +_2 (k_2 \oplus d_{i-1}); 1 \le i \le 319;$$
$$d_i = [(TEK_i \oplus NTEK_{i-1}) +_2 (k_1 \oplus d_{i-1})] +_2 (k_3 \oplus NTEK_{i-1}); 1 \le i \le 319$$

in which $NTEK_0 = b_0, d_0 = c_0.$

*NTEK*s are derived from *TEK*s by using a feedback control process, which as shown in Figure 5 generates two outputs, $NTEK_i$ and d_i, when inputting TEK_i, $1 \le i \le 319$. $NTEK_i$ and d_i as the feedback parameters of the *i*th stage will be a part of the inputs of the (*i*+1)th stage.

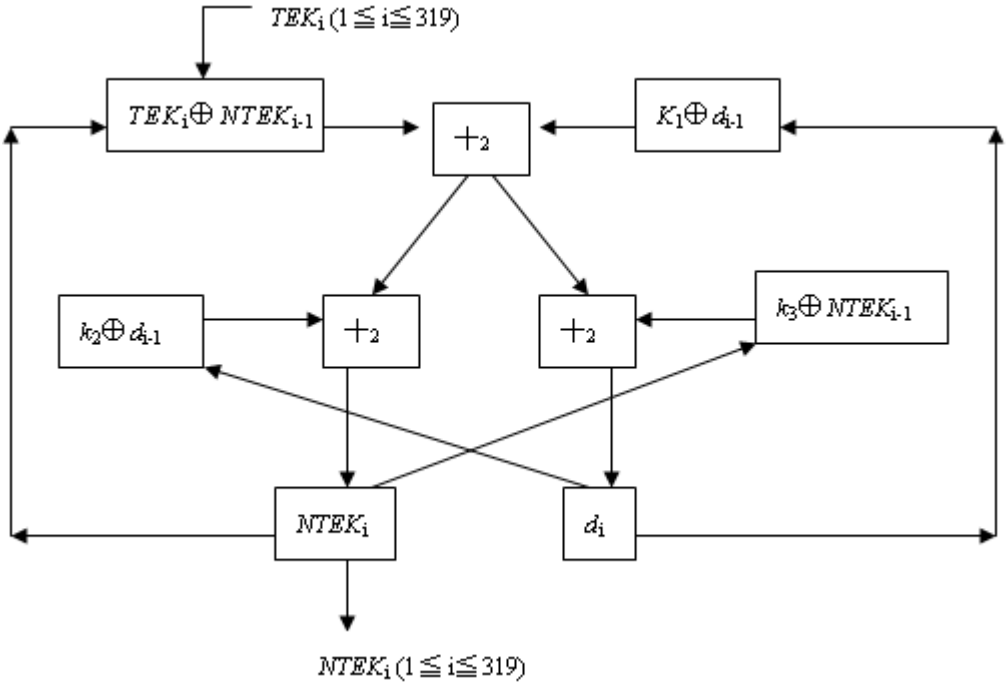

Figure 5. The process that generates 319 NTEKs by employing 319 TEKs from 320 TEKs and the feedback control process

Step 4: On receiving message 2, the user checks to see whether its current status, i.e., authentication reply, matches the one conveyed in the *OP-code* or not. If not, the user drops this fake message. Otherwise, the user utilizes the internal keys generated, including b_0, b_1, b_2, b_3, b_4, b_5, and c_0, to decode c_i so as to acquire ϕ_i, i=1,2,3,4, which together with ψ_2, ψ_3, and ψ_4, are used to decrypt message 2 where

$$\phi_1 = \begin{cases} \{[c_1 \oplus (c_0 +_2 b_5)] -_2 b_1\} \oplus \psi_2, \text{ if } [c_1 \oplus (c_0 +_2 b_5)] \ge b_1; \\ \{[c_1 \oplus (c_0 +_2 b_5)] +_2 \overline{b_1} +_2 1\} \oplus \psi_2, \text{ if } [c_1 \oplus (c_0 +_2 b_5)] < b_1. \end{cases}$$

$$\phi_2 = \begin{cases} \{[c_2 \oplus (\psi_2 +_2 b_1)] -_2 b_2\} \oplus \psi_3, \text{ if } [c_2 \oplus (\psi_2 +_2 b_1)] \ge b_2; \\ \{[c_2 \oplus (\psi_2 +_2 b_1)] +_2 \overline{b_2} +_2 1\} \oplus \psi_3, \text{ if } [c_2 \oplus (\psi_2 +_2 b_1)] < b_2. \end{cases}$$

$$\phi_3 = \begin{cases} \{[c_3 \oplus (\psi_3 +_2 b_2)] -_2 b_3\} \oplus \psi_4, \text{ if } [c_3 \oplus (\psi_3 +_2 b_2)] \geq b_3; \\ \{[c_3 \oplus (\psi_3 +_2 b_2)] +_2 \overline{b_3} +_2 1\} \oplus \psi_4, \text{ if } [c_3 \oplus (\psi_3 +_2 b_2)] < b_3. \end{cases}$$

$$\phi_4 = \begin{cases} \{[c_4 \oplus (\psi_4 +_2 b_3)] -_2 b_4\} \oplus b_5, \text{ if } [c_4 \oplus (\psi_4 +_2 b_3)] \geq b_4; \\ \{[c_4 \oplus (\psi_4 +_2 b_3)] +_2 \overline{b_4} +_2 1\} \oplus b_5, \text{ if } [c_4 \oplus (\psi_4 +_2 b_3)] < b_4. \end{cases}$$

The user verifies message 2 by checking to see whether $HMAC((\phi_2 \oplus \phi_3) +_2 \phi_4)_r = HMAC((\phi_2 \oplus \phi_3) +_2 \phi_4)_c$ or not. If not, the user discards the fake message and waits for a legal one. Otherwise, she/he generates *TEK*s, *NTEK*s and internally-used d_is where

$$TEK_{(i-1)\times 80+(j-1)\times 20+(k-1)\times 4+l} = [(\phi_i +_2 c_j) \oplus b_k] +_2 \psi_l; 1 \leq i, j, l \leq 4, 1 \leq k \leq 5.$$

and

$$NTEK_i = [(TEK_i \oplus NTEK_{i-1}) +_2 (k_1 \oplus d_{i-1})] +_2 (k_2 \oplus d_{i-1}); 1 \leq i \leq 319;$$
$$d_i = [(TEK_i \oplus NTEK_{i-1}) +_2 (k_1 \oplus d_{i-1})] +_2 (k_3 \oplus NTEK_{i-1}); 1 \leq i \leq 319.$$

in which $NTEK_0 = b_0, d_0 = c_0$. Now the user also sets its current status to "data transmission".

Step 5: The user

(1) encrypts plaintext to ciphertext. *If* $Plaintext = p_0\ p_1\ p_2 \cdots p_{n-1}$ *and* the corresponding $Ciphertext = c_0\ c_1\ c_2 \cdots c_{n-1}$, then

$$c_i = (p_i \oplus TEK_{j_1}) +_2 NTEK_{j_2}, 0 \leq i \leq n-1, \quad (1)$$

where $j_1 = (i + m_1) \bmod 320 + 1,\ j_2 = (i + m_2) \bmod 319 + 1,\quad 0 \leq m_1 \leq 319,\quad 0 \leq m_2 \leq 318.$ (2)

(2) records its current status, i.e., "data transmission", to *OP-code*;

(3) sends a data message, denoted by message 3, to Authenticator. The format of this message is

$$OP-code \mid UserID \mid RHSEXOR(\phi_1 \oplus \phi_2, m_1 // m_2) \mid (TEK_{m_2} \oplus NTEK_{m_1}) +_2 NTEK_{m_3} \mid Ciphertext,$$

where $m_3 = (m_1 + m_2) \bmod 319 + 1.$ After transmitting this message, the user sets its current status to "data receiving".

Step 6: Upon receiving the ciphertext, Authenticator checks to see whether its status, i.e., data transmission, matches the status conveyed in the *OP-code* or not. If not, it drops this fake message and waits for a legal one. Otherwise, Authenticator

(1) acquires m_1 and m_2 by decoding message 3 where

$m_1 // m_2 = RHSEXOR(\phi_1 \oplus \phi_2, RHSEXOR(\phi_1 \oplus \phi_2, m_1 // m_2))$ in which // represents concatenation;

(2) verifies whether message 3 is issued by a legal user or not by checking to see whether the $(TEK_{m_2} \oplus NTEK_{m_1}) +_2 NTEK_{m_3}$ conveyed in the message is equal to the $(TEK_{m_2} \oplus NTEK_{m_1}) +_2 NTEK_{m_3}$ calculated by itself or not. If not, Authenticator discards the fake message and waits for a legal one. Otherwise, it decrypts the message to acquire the plaintext p_i where

$$p_i = \begin{cases} (c_i -_2 NTEK_{j_2}) \oplus TEK_{j_1}, \text{ if } c_i \geq NTEK_{j_2}; \\ (c_i +_2 \overline{NTEK_{j_2}} +_2 1) \oplus TEK_{j_1}, \text{ if } c_i < NTEK_{j_2}, \end{cases}$$

in which $0 \leq i \leq n-1,\ j_1 = (i + m_1) \bmod 320 + 1,\ j_2 = (i + m_2) \bmod 319 + 1,$

$0 \leq m_1 \leq 319$， $0 \leq m_2 \leq 318$.

Chapter 4 Security Analysis

In this chapter, we analyze the security of the key exchange process, i.e., steps 1~4, and the two-dimensional stream cipher technique, and describe how the WiSDC effectively defends three common attacks, including eavesdropping attack, replay attack and forgery attack. The security levels of the WiSDC with those of SSL and IPsec are also compared.

4.1 Security of the key exchange process

Let X and Y be two keys, each of which is n bits in length. The probability p of recovering the value of (X, Y) from illegally intercepted $X \oplus Y$ on one trial is $p = \frac{1}{2^n}$ [23]. But what is the recovering probability of $X +_2 Y$?

Example 2: Let X,Y and Z be three keys, each of which is 4-bit in length, and let $Z = X +_2 Y$. All possible values of (X, Y) that meet Z = 0101 are listed in Table 2.

Table 2. All possible values of (X, Y) that meet Z = X $+_2$ Y = 0101

Without carry		With carry	
X	Y	X	Y
0000	0101	1111	0110
0001	0100	1110	0111
0010	0011	1101	1000
0011	0010	1100	1001
0100	0001	1011	1010
0101	0000	1010	1011
		1001	1100
		1000	1101
		0111	1110
		0110	1111

Lemma 1:

Assume that both the two keys *X* and *Y* are *m*-bit in length. The probability *p* of recovering the value of (*X*, *Y*) from illegally intercepted $X +_2 Y$ on one trial is $p = \frac{1}{2^m}$.

Proof:

When $Z = X +_2 Y$ is performed, the binary addition of the highest bits has two cases, with and without carry.

Case 1: If the case without carry occurs, $Z = X +_2 Y$ can be reduced to $Z = X + Y$, and the possible values of (*X*, *Y*) are (0, *Z*), (1, *Z*-1), (2, *Z*-2), …, (*Z*-1, 1) and (*Z*, 0), i.e., a total of *Z*+1 possibilities.

Case 2: If the case with carry occurs, since we ignore this carry of the most significant-bit addition, $Z = X +_2 Y$ can be expressed as $Z = X + Y - 2^m$, and due to $X + Y = Z + 2^m$, the possible values of (*X*, *Y*) are $(2^m - 1, Z + 1)$, $(2^m - 2, Z + 2)$, $(2^m - 3, Z + 3)$, …, $(Z + 2, 2^m - 2)$, and $(Z + 1, 2^m - 1)$, .i.e., a total of $2^m - Z - 1$ possibilities after ignoring the carry. Hence, for each *Z*, there is a total of (*Z*+1)+($2^m - Z - 1$) = 2^m possible values of (*X*, *Y*) that meet $Z = X +_2 Y$. The probability *p* of recovering the original value of (*X*, *Y*) on one trial is then $p = \frac{1}{2^m}$.

Also, hackers cannot retrieve the internal keys from the delivered messages. The only method for them to crack the WiSDC is to acquire the random parameter ψ_1 from the communication key a_1 conveyed in message 1. What is the recovering probability of ψ_1 from a known a_1?

Lemma 2:

Assume that both random parameter ψ_1 and communication key a_1 are *m*-bit in length. The probability *p* of recovering the value of ψ_1 from known a_1 is also $p = \frac{1}{2^m}$.

Proof:

According to previous description,

$a_1 = \{[k_i \oplus k_1] +_2 \psi_1] \oplus (k_1 +_2 k_2)\} +_2 (k_2 \oplus k_3)$

$= [(k_1' +_2 \psi_1) \oplus k_2'] +_2 k_3'$

where $k_1' = k_i \oplus k_1,\ k_2' = k_1 +_2 k_2$ and $k_3' = k_2 \oplus k_3$ (3)

If a_1 is known to hackers and Eq.(3) is employed to recover ψ_1, then $k_1', k_2', \text{and } k_3'$ must be obtained beforehand. However, $k_1', k_2', \text{and } k_3'$ are determined by the connection keys in the DCC, i.e., $k_1, k_2, k_3 \text{ and } k_i$ which are unknown to hackers. Also, different a_1s are generated by invoking different ψ_1s. Hence, the collection of a large number of a_1 is useless in cracking the connection keys and recovering ψ_1. Then, due to invoking three operations (i.e., two $+_2$s and one $\oplus$) shown in Eq.(3), the probability p of recovering $\psi_1, k_1', k_2', \text{and } k_3'$ from a_1 by using Eq.(3) is $\left(\frac{1}{2^m}\right)^3$ which is much smaller than $\frac{1}{2^m}$, the probability of blind guessing the value of ψ_1 on one trial when a_1 is known, showing that, no matter whether Eq.(3) is employed or not, the probability p of recovering the value of ψ_1 from a known a_1 is $p = \frac{1}{2^m}$.

The internal keys $b_0, c_0, b_1 \sim b_5$ which are derived from ψ_1 and connection keys are unknown to hackers. Similarly, the probability p of recovering the values of ψ_j from known a_j is $p = \frac{1}{2^m}$ for each $j,\ 2 \le j \le 4$.

An authentication code *HMAC*() used to authenticate a received message has two other characteristics, including nonrepudiation and data integrity.

Lemma 3:

In message 1, $HMAC((b_1 \oplus b_2) +_2 b_3)$ is an authentication code with three other security functions, including authentication, nonrepudiation and integrity.

Proof:

(Proof of authentication)

To correctly generate the key $(b_1 \oplus b_2) +_2 b_3$, the following two steps are required:

(1) Deriving ψ_1 from a_1 and the connection keys, i.e., $k_i, k_1, k_2, \text{and } k_3$.

(2) Deriving internal keys $b_0, c_0, b_1 \sim b_5$ from ψ_1 calculated above and the connection keys, implying that only the hackers who have acquired the connection keys in the DCC can correctly generate the key $(b_1 \oplus b_2) +_2 b_3$. Hence, only the legitimate user who has the connection keys in the DCC can generate correct $HMAC((b_1 \oplus b_2) +_2 b_3)$, i.e., $HMAC((b_1 \oplus b_2) +_2 b_3)_c = HMAC((b_1 \oplus b_2) +_2 b_3)_r$ where the subscripts c and r stand for calculation and received, respectively. Those illegitimate hackers who have no connection keys cannot achieve this.

(Proof of nonrepudiation)

From the analysis above, only the legitimate user can acquire the connection keys from AAA server and make $HMAC((b_1 \oplus b_2) +_2 b_3)_c = HMAC((b_1 \oplus b_2) +_2 b_3)_r$, implying that message 1 is sent by the legitimate user who has been authenticated by the AAA server.

(Proof of the integrity)

$HMAC((b_1 \oplus b_2) +_2 b_3)$ is the authentication code generated by invoking a hash function performed on the plaintext, *OP-code*|T_{nonce}|*UserID*|a_1|a_2|a_3|a_4|, with the key, $(b_1 \oplus b_2) +_2 b_3$. If either the plaintext or the key has been illegally tampered with, then $HMAC((b_1 \oplus b_2) +_2 b_3)_c \neq HMAC((b_1 \oplus b_2) +_2 b_3)_r$ since the value of $HMAC((b_1 \oplus b_2) +_2 b_3)$ cannot be correctly calculated by hackers who have no connection keys. Hence, if $HMAC((b_1 \oplus b_2) +_2 b_3)_c = HMAC((b_1 \oplus b_2) +_2 b_3)_r$, meaning message 1 has not illegally tampered with and the integrity has been maintained.

Besides $HMAC((b_1 \oplus b_2) +_2 b_3)$, T_{nonce} also provides a security function.

Lemma 4:

In message 1, both T_{nonce} and $HMAC((b_1 \oplus b_2) +_2 b_3)$ provide the security functions which can effectively defend the replay attacks.

Proof:

If hackers illegally duplicate message 1, and resend it, then T_{nonce} contained in this message is not current time so that $T_{received}$ - $T_{nonce} \geq \Delta T$ where ΔT is a predefined short time period. The message will be discarded by the Authenticator. If hackers modify T_{nonce} to current time, the

value of calculated $HMAC((b_1 \oplus b_2) +_2 b_3)$ will change, and also without connection keys $k_i, k_1 \sim k_3$, hackers cannot calculate the correct value of $HMAC((b_1 \oplus b_2) +_2 b_3)$. Hence, $HMAC((b_1 \oplus b_2) +_2 b_3)_c$ will not be equal to $HMAC((b_1 \oplus b_2) +_2 b_3)_r$, indicating that the security function which both T_{nonce} and $HMAC((b_1 \oplus b_2) +_2 b_3)$ provide can effectively defend the replay attacks.

In the WiSDC, plaintext is encrypted by using two different PRNSs, i.e., *TEK*s and *NTEK*s with periods of 320 and 319 units, respectively, where a unit is a key length which may be 512, 768, 1024 bits or other lengths. The length of TEK_{j1} or $NTEK_{j2}$ is one unit. But by invoking random index m_1 and m_2 (see Eq.(2)), and different periods of *TEK*s and *NTEK*s, they can produce 102,080 (= 320*319) different PRNSs, each of which is also 102,080 units in length in each repeated cycle of the generated stream for encrypting plaintext and decrypting ciphertext.

Lemma 5:

In the WiSDC, a given plaintext can be encrypted by using one of the 102,080 PRNSs, and each of the PRNSs has a period of 102,080 units.

Proof:

If $\text{Plaintext} = p_0\ p_1\ p_2 \cdots p_{n-1}$ and the corresponding $Ciphertext = c_0\ c_1\ c_2 \cdots c_{n-1}$, then based on Eqs. (1) and (2), the initial value of TEK_{j1} and $NTEK_{j2}$ are decided by m_1 and m_2, respectively. Different initial values of TEK_{j1} and $NTEK_{j2}$ will result in different PRNSs, thus generating different ciphertexts. There are 320 and 319 possible values of m_1 and m_2, respectively. By the Rule of Product, $320 \times 319 = 102080$ possible PRNSs can be generated, and the periods of j_1 and j_2 of a PRNG are 320 and 319, respectively. Hence, the period of the resulting PRNS is $320 \times 319 = 102080$ units.

After lemma 5, lemma 6 describes how to encrypt plaintext into ciphertext.

Lemma 6:

Let $Q = q_0 q_1 q_2 \cdots q_{n-1}$ be the plaintext which is a string of *n* characters, and each character, e.g., q_i, $0 \le i \le n\text{-}1$, is *m*-bits in length. Let PRNS1 and PRNS2 be two pseudo random number sequences, in which PRNS1 $= r_0 r_1 r_2 ... r_n r_{n+1} ...$, PRNS2 $= s_0 s_1 s_2 ... s_n s_{n+1} ...$, and both r_j and s_j are *m*-bit binary numbers, $j \ge 0$. Let c = $c_0 c_1 c_2 ... c_{n-1}$ be the ciphertext where $c_j = (q_j \oplus r_j) +_2 s_j, \quad 0 \le j \le n-1$. Then, the probability *p* of recovering $(q_0 q_1 q_2 ... q_{n-1}, r_0 r_1 r_2 ... r_{n-1}, s_0 s_1 s_2 ... s_{n-1})$ from illegally intercepted ciphertext $c_0 c_1 c_2 ... c_{n-1}$ on one trial is $p = \left(\frac{1}{4^m}\right)^n$.

Proof:

The ciphertext c_j is generated by using the formula $c_j = (q_j \oplus r_j) +_2 s_j$, $0 \le j \le n-1$. Due to invoking two operators to calculate c_j, the probability p_j of acquiring the right values of (q_j, r_j, s_j) from the illegally intercepted c_j on one trial is $\frac{1}{4^m} (= \frac{1}{2^m} \times \frac{1}{2^m})$. Since each triple (q_j, r_j, s_j) is independent from others, implying that $p_0 = p_1 = ... = p_{n-1}$, the probability *p* of acquiring the right values of $(q_0 q_1 q_2 ... q_{n-1}, r_0 r_1 r_2 ... r_{n-1}, s_0 s_1 s_2 ... s_{n-1})$ from the illegally intercepted ciphertext $c_0 c_1 c_2 ... c_{n-1}$ on one trial is $p = p_0 p_1 \cdots p_{n-1} = \left(\frac{1}{4^m}\right)^n$.

4.2 Cryptanalysis of Attacks

The WiSDC can defend eavesdropping, forgery and replay attacks.

4.2.1 Eavesdropping attack

Eavesdropping is one type of attack which due to the wireless nature is not easily discovered. Hackers may maliciously intercept the messages sent by users, and analyze the messages to acquire useful information.

In the WiSDC, hackers can only acquire communication keys $a_1 \sim a_4$ from the illegally intercepted message 1. However, a_1 is generated by using random parameter ψ_1 and the

connection keys in the DCC, $a_2, a_3, \text{and } a_4$ are derived from random parameters $\psi_2, \psi_3, \text{and } \psi_4$ and internal keys $b_0, c_0, b_1 \sim b_5$. However, these internal keys are also generated by invoking random parameter ψ_1 and the connection keys. Hence, the only method to acquire useful information from the transmitted message is recovering ψ_1 from a_1. By Lemma 2, the probability p of recovering the value of ψ_1 from known a_1 is $p = \frac{1}{2^m}$, showing that ψ_1 is well protected so that hackers cannot easily crack the communication keys, solve the transmitted messages and acquire the plaintext. That means the plaintext is secure.

4.2.2 Forgery attack

Hackers often masquerade themselves as legitimate users or an Authenticator to acquire the authentication information. Namely, if a system does not provide a mutual authentication, a hacker may be considered as a legitimate user (the Authenticator), and then the messages sent to the Authenticator (the users) will be treated as legal ones.

Lemma 3 shows that the key exchange mechanism of the WiSDC preserves mutual authentication, implying that only the one who has the DCC can correctly generate the dynamic authentication key $(b_1 \oplus b_2) +_2 b_3$. The forged messages generated by hackers, who do not have the DCC, cannot pass the authentication and will be discarded by the user or Authenticator. Hence, the WiSDC can defend a forgery attack effectively.

4.2.3 Replay attack

When intercepting an authentication message, hackers will tamper with it and send it to users or Authenticator to gain trust. Hackers may also send duplicate messages two or more times to users or Authenticator, making the receiver confused about which messages are the legal ones.

Lemma 4 shows that the duplicated message 1 sent to Authenticator cannot pass the authentication test. Furthermore, sending the duplicated message 2 to user is also useless since the time point of sending the duplicated message 2 is later than the time point when the original

one was delivered. When the user receives message 2 from the legitimate Authenticator and message 2 passes the authentication test, the internal state of the user will be set to the next state. But the state carried in the *OP-code* of the duplicated message 2 remains in its original state, which cannot meet the state of the receiver. The other duplicated messages have the similar phenomenon. Hence, the WiSDC can effectively defend the replay attack.

4.3 The security level comparison

The compared protocols include the SSL, IPsec and WiSDC.

4.3.1 Eavesdropping Attack

In the WiSDC, from the time point when message1 is sent to the moment when the ciphertext message, i.e., message 3, is delivered, all transmitted messages as shown and discussed above are all well protected, implying that the WiSDC can effectively defend the eavesdropping attack. However, in the SSL, the messages delivered in its steps 1 and 2 (see Appendix A of this paper) are transmitted through the air without any protection, indicating that the eavesdropping attack on the SSL is somewhat effective.

In the IPsec, the messages sent in the first four steps of the IKE main mode are also transmitted through the air without any protection (see Appendix B of this paper), indicating that the eavesdropping attack on IPsec is also somewhat effective.

4.3.2 Forgery Attack

In the SSL, the forgery server attack is effective and is described in the following processes.

Process 1: Hackers first collect the certificate of a server S from the air in step 2 (see Appendix A), and acquire S's public key from S's X.509 certificates. Then the hackers now own S's certificates and the corresponding public key.

Process 2: Some time later, when the client tries to issue a new key exchange process, and the

hackers receive the information sent to S by the client in step 1 of the new exchange process, the hackers reply to the client with S's certificate obtained in process 1, and arbitrary RNs.

Process 3: Hackers receive the encrypted pre-master key and the ciphertext encrypted by using the master key sent to S. Now, the forgery server attack has been partially performed, i.e., once *p* and *q* of the employed RSA are known to the hackers, SSL will be successfully cracked.

In the IPsec, the forgery attack is effective and described in the following processes.

Process 1: Hackers collect the source IP of the responder from the air in step 2 (see Appendix B).

Process 2: Some time later, when the initiator issues a new IKE process by sending message 1 to the responder. On receiving the message, the hackers reply with message 2, which carries the source IP obtained in process 1, to the initiator.

Process 3: While hackers receive message 3 sent by the initiator in step 3 of the new IKE process, they reply with message 4, which contains X_b and N_r generated by the hackers, to the initiator. The common secret key (CSK for short) is now owned by both the hackers and the initiator, implying that a forgery server attack has been partially performed, i.e., once the IP's corresponding pre-shared key is known to the hackers, IPsec will be cracked.

4.3.3 Replay Attack

When the WiSDC is evaluated, a replay attack can occur only during the delivery of the first control message, i.e., message 1. This attack does not work in step 2 and the consequent steps since the duplicated message is now out of state and date.

The messages delivered in the first steps of SSL and IPsec are not well protected to defend the replay attack since all messages transmitted in their first steps are not encrypted. A replay attack can be effectively issued on both of them.

4.4 Summary

This section summarizes and compares the security characteristics of the SSL, IPsec and WiSDC. Before the generation of the master secret key, the SSL does not provide authentication between the client and server, and IPsec enforces only device authentication. No user authentication between the initiator and responder is performed. The WiSDC provides a mutual authentication between the user and authenticator.

Chapter 5 Simulation

The simulation was performed in a client-server environment. The program is developed by using Java. The hardware specifications of the test-bed are listed in Table 4.

Table 4. The specifications of the test-bed utilized to simulate the user device and Authenticator.

Component	User	Authenticator
CPU	Intel E6500 2.93 GHz	Intel i7-3770 3.40 GHz
RAM	2GB	16GB
Platform	Windows 7	Windows 7

5.1 Simulation Model

With the proposed scheme, we simulated the internal keys, communication keys and the ciphertext transmitted between the user and Authenticator through two different wireless systems, including the IEEE 802.11b and a 3.5G system named the High Speed Downlink Packet Access (HSDPA for short) [24], without employing encryption methods. The timings of key generation for both the user and Authenticator were evaluated to see whether they were reasonable and acceptable or not. Also, we computed the individual cost of steps 1~4, and studied the times required by RSA and Diffie-Hellman PKDS [25] respectively invoked by SSL and IPsec. Last, we calculated the required data transmission times when messages of different key sizes were delivered through the two employed wireless systems.

5.2 Simulation Analysis

5.2.1 Timings required for encryption

In the authentication phase, we chose 512, 768, and 1024 bits as the key lengths to evaluate the times consumed to generate the internal keys, i.e., b_0, b_1, b_2, b_3, b_4, b_5, and c_0, on both the user and Authenticator ends, communication keys, i.e., a_1, a_2, a_3, and a_4, on the user end, and communication keys, i.e., c_1, c_2, c_3 and c_4, on Authenticator end. Table 5 lists the simulation results, in which the time required to generate internal keys were small, ranging between 6.05 μs on Authenticator end and 41.68 μs on user end when key length was 1024 bits, since only two encryption operators $\oplus$ and $+_2$ were used. No complicated functions, such as exponential functions [26] or factorial functions [27], were employed.

In this study, a feedback control process was deployed to generate *NTEK*s and a hidden parameter d_i to increase the security level of *NTEK*s. Due to the generation of d_i, the times consumed to produce *NTEK*s as shown in Table 6 were longer than those required to generate *TEK*s.

Table 5. The timings required to generate the internal keys and communication keys on both the user and Authenticator ends in the authentication phase

	Key generation times (μs)					
	User end			Authenticator end		
Size / Key	512	768	1024	512	768	1024
b_0	20.62	31.06	41.68	5.11	8.18	11.98
c_0	19.10	29.20	38.62	4.38	6.85	10.67
b_1	13.92	21.29	28.95	3.60	5.43	7.38
b_2	17.44	26.58	35.25	3.99	6.28	9.73
b_3	13.82	21.58	28.87	3.62	5.79	7.96
b_4	12.48	19.21	28.49	3.19	4.66	6.52
b_5	12.78	19.04	25.56	2.87	4.35	6.05
a_1	18.95	28.26	38.69			
a_2	14.23	21.46	28.88			
a_3	14.04	21.29	28.86			
a_4	14.09	21.42	29.00			
c_1				3.03	4.66	6.47
c_2				2.91	4.28	6.24
c_3				3.19	4.72	6.32
c_4				3.19	4.73	6.48

Table 6. The timings required to calculate the TEKs and NTEKs on both the user and Authenticator ends in the authentication phase

	Key generation times (ms)					
	User end			Authenticator end		
Size / Keys	512	768	1024	512	768	1024
$TEK_{1\text{-}320}$	1.93	2.92	3.90	0.13	0.19	0.26
$NTEK_{1\text{-}319}$	4.92	7.67	10.34	0.30	0.44	0.60

In the data transmission phase, a set of credit card information as the plaintext of 512, 768, or 1024 bits long was encrypted with TEK_{j1} and $NTEK_{j2}$ by the user and decrypted by Authenticator. The timings required are listed in Table 7, in which the time spent encrypting plaintext of 1M bits in length was only 12.59 (= 1000 * 12.59 μs) ms on the user end, and that consumed to decrypt the ciphertext was 3.5 (= 1000 * 3.5 μs) ms on the Authenticator end, showing that in the WiSDC system, the two-dimensional stream cipher technique can efficiently encrypt/decrypt plaintext/ciphertext.

Table 7. The timings required to encrypt messages by the user and decrypt messages by Authenticator by using the two-dimensional stream cipher technique in the data transmission phase

Plaintext size	Encryption time (User end)	Decryption time (Authenticator end)
512 bits	6.12 μs	1.5 μs
768 bits	9.24 μs	2.6 μs
1024 bits	12.59 μs	3.5 μs

Table 8. Time consumed by each of the four WiSDC steps

	Times consumed (ms)			
Size / Step	512	768	1024	Remark
1	0.319	0.481	0.658	User end
2	0.054	0.086	0.129	Authenticator end
3	0.44	0.643	0.89	Authenticator end
4	6.938	10.733	14.473	User end

Table 9. The times required to perform the RSA encryption/decryption and generate the Diffie-Hellman PKDS public key and common secret key

	Key generation times (ms)					
	User end			Authenticator end		
Size / Functions	512	768	1024	512	768	1024
RSA	2.51	7.80	17.38	0.59	1.67	3.75
Diffie-Hellman PKDS	0.76	2.33	4.99	0.17	0.47	1.00
DH : Common secret key	2.56	7.93	17.69	0.55	1.65	3.53

Furthermore, the times for generating the random parameters and the keys, and retrieving keys of 521, 768 and 1024 bits long on each of the four WiSDC steps are listed in Table 8.

To effectively compare the efficiencies of the SSL, IPsec and WiSDC, the timings required to perform the RSA encryption/decryption and generate the Diffie-Hellman PKDS public key and common secret key were studied. The results are listed in Table 9.

SSL employs an RSA encryption function on user end and an RSA decryption function on Authenticator end. The RSA encryption function on key length=1024 bits consumed 17.38 ms (see Table 9) which is larger than 15.131 (=0.658+14.473) ms (see Table 8), the sum of the times consumed by steps 1 and 4 of the WiSDC. The time required to generate the RSA decryption key on the Authenticator end is 3.75 ms (see Table 9), which is larger than the sum of the times consumed by steps 2 and 3 of the WiSDC, i.e., 1.019 (=0.129+0.89, see Table 8) ms, showing

that the WiSDC is more efficient than SSL.

IPsec invokes both the Diffie-Hellman PKDS public key function and the common secret key function for both the user and Authenticator ends. However, the sum of the time consumed to generate a public key and a common secret key on the user end is 22.68 (=4.99+17.69, see Table 9) ms, which is larger than the time (15.131(=0.658+14.473) ms) consumed by the WiSDC. The time required to generate a public key and a common secret key on the Authenticator end is 4.53 (=1.0+3.53) ms, which is large than the time consumed by the WiSDC, i.e., 1.019 (=0.129+0.89) ms, indicating that the WiSDC is also more efficient than IPsec.

5.2.2 Transmission rate analysis

In the following simulation, 802.11b (Bandwidth = 11 Mbps) was used to send message 1, and HSDPA (Bandwidth = 3.6 Mbps) was deployed to deliver message 2. Different lengths of message 1 and message 2 were tested. As mentioned above, message 1 is composed of *OP_code*, T_{nonce}, *UserID*, a_1~a_4 and *HMAC*(), and the components of message 2 include *OP_code*, c_1~c_4 and *HMAC*(). The lengths of *OP_code*, T_{nonce} and *UserID* were 8, 48 and 20 bits long, respectively. Different lengths of the communication keys a_1~a_4 and c_1~c_4, including 512 bits, 768 bits and 1024 bits, were tested, individually. The sizes of the components of message 1 and message 2 are listed in Table 10. The times required to transmit message 1 and message 2 are shown in Table 11.

Table 10. The sizes of message 1 and message 2

Packet sizes / Key lengths		Message1 (bits)	Message2 (bits)
OP_code		8	
T_{nonce}		48	
UserID		20	20
a_1~a_4 / c_1~c_4 / *HMAC*()	512 bits	2560	2560
	768 bits	3840	3840
	1024 bits	5120	5120

Table 11. The times required to transmit message 1 and message 2 on different lengths of keys

Media / Key Length	802.11b (11Mbps)	HSDPA (3.6Mbps)
512 bits	0.24 ms (=(76+2560)/11 M)	0.76 ms (=(20+2560)/3.6 M)
768 bits	0.36 ms (=(76+3840)/11 M)	1.1 ms (=(20+3840)/ 3.6 M)
1024 bits	0.47 ms (=(76+5120)/11 M)	1.4 ms (=(20+5120)/ 3.6 M)

Now, we can conclude that the WiSDC has many advantages. First, the system can adapt to different types of security systems that need to deliver communication keys, such as a_1~a_4 and c_1~c_4 in their key exchange processes. Second, it spends less time to generate keys compared with the times consumed by the SSL and IPsec. Third, the WiSDC is employing the two-dimensional stream cipher technique is more efficient in encrypting/decrypting the

plaintext/ciphertext than the SSL and IPsec which respectively utilize the RSA and Diffie-Hellman PKDS as their key exchange techniques. Lastly, the WiSDC spends less time to transmitting messages through the wireless environments, including the IEEE 802.11b and the invoked 3.5G system. In summary, our mechanism is more suitable for key exchange and the delivery of messages than SSL and IPsec at least in these two employed wireless environments.

Chapter 6 Conclusions and Future work

In this study, the WiSDC employs the DCC to pre-establish a virtual connection between the user and AAA home server before communication starts. Further, three other secure mechanisms are developed to protect messages delivered through the air. The first mechanism is producing internal keys from random numbers and connection keys, and then generating communication keys from the internal keys. Communication keys, instead of the DCC, are transmitted through the air. The second is reducing key exchange steps to lower the probability of important information being captured. Moreover, internal keys and communication keys are produced by using two elementary operators, i.e. $\oplus$ and $+_2$, to enhance the key generation efficiency compared with the key generation efficiencies when the SSL and IPsec are tested. The third is employing a two-dimensional stream cipher technique to encrypt/decrypt the transmitted messages.

The theoretical analysis shows that the WiSDC has the following advantages, including (1) preserves the Secrecy, Authenticity, Integrity and Nonrepudiation characteristics for the transmitted messages; (2) effectively defends several above mentioned attacks; (3) has higher security level and execution efficiency than those of the SSL and IPsec.

So, if an Internet Service Provider adopts the WiSDC as its security mechanism, the customers' communication can be more securely protected, and packets can be more efficiently transmitted. Also, our techniques can be integrated with the standard of 802.1x and HSDPA individually so as to strengthen their key exchange time and reduce the key exchange steps before data messages are transmitted through the wireless environment.

In the future, we would like to apply the WiSDC to a cloud network [28,29] to enhance the security of the channels between clients/local servers and remote servers, and derive the reliability model [30,31] and behavior model [32] for the WiSDC so that users can determine their system reliabilities and behaviors, respectively, before using them. These constitute our future research.

References

[1] W. Chou, "Inside SSL: the secure sockets layer protocol," IT Professional, vol. 4, Issue 4, July /August 2002, pp. 47－52.

[2] S. Kolli, M. Zawodniok "Energy-efficient multi-key security scheme for wireless sensor network," the IEEE Conference on Local Computer Networks, 2009, pp.937－944.

[3] R. Perlman and C. Kaufman, "Key exchange in IPSec: analysis of IKE," Internet Computing IEEE, vol. 4, Issue 6, Nov. /Dec. 2000, pp.50－56.

[4] K.C. Wei, Y.L. Huang and F.Y. Leu, “A Secure Communication over Wireless Environments by using a Data Connection Core,” The IEEE International Workshop on Mobile Commerce, Cloud Computing, Network and Communication Security, July 2012, pp.570-575.

[5] Y.L. Huang, K.C. Wei, F.Y. Leu, "Constructing a Secure Point-to-Point Wireless Environments by Integrating Diffie-Hellman PKDS and Stream Ciphering Without Certificate Authorities," the International Conference on Complex, Intelligent and Software Intensive Systems, Feb. 2010, pp. 384－390.

[6] C. De Canniere, A. Biryukov and B. Preneel, "An Introduction to Block Cipher Cryptanalysis," Proceedings of the IEEE, vol. 94, Issue 2, Feb. 2006, pp. 346－356.

[7] F. Anstett, G. Millerioux, and G. Bloch, "Chaotic Cryptosystems: Cryptanalysis and Identifiability," IEEE Transactions on Circuits and System I: Regular Papers, vol. 53, Issue 12, Dec. 2006, pp.2673－2680.

[8] C. Kefei, G.. Meng and G. Ruijie, "Analysis and Research on HTTPS Hijacking Attacks," 2010 International Conference on Networks Security Wireless Communications and Trusted Computing, 2010, pp.223－226.

[9] Y.F. Huang, F.Y. Leu and K.C. Wei, "Constructing a Secure Point-to-Point Wireless Environment by Integrating Diffie-Hellman PKDS and Stream Cipher," the International Conference on Complex, Intelligent and Software Intensive Systems, 2010, pp. 384－390.

[10] Y.L. Huang, C.H. Lin and K.L.i Wen, "A Pseudorandom Number Generator Based on Grey

System Theory," Far East Journal of Mathematical Sciences, vol. 35, Issue 1, pp. 1－17, September 2009.

[11] A.C. Weaver, "Secure Socket Layer," Computer, vol. 39, Issue 4, April 2006, pp. 88－90.

[12] D.V. Bhatt, S. Schulze, G.P. Hancke, "Secure Internet access to gateway using secure socket layer," IEEE Transactions on Instrumentation and Measurement, vol. 55, Issue 3, June 2006, pp. 793－800.

[13] F. Callegati, W. Cerroni and M. Ramilli, "Man-in-the-Middle Attack to the HTTPS Protocol," IEEE Security & Privacy, vol. 7, Issue 1, Feb. 2009, pp. 78－81.

[14] L. Du, X. Hu, Y. Li and G. Zhao, "A CSK based SSL handshake protocol," IEEE International Conference on Network Infrastructure and Digital Content, 2009, pp. 600－603.

[15] Wiki, IPsec, http://en.wikipedia.org/wiki/IPsec

[16] H. Haddad and H. Mirmohamadi, "Comparative evaluation of successor protocols to Internet Key Exchange (IKE)," the International Conference on Industrial Informatics, 2005, pp. 692－696.

[17] M.R. Ogiela and U. Ogiela, “The Use of Mathematical Linguistic Methods in Creating Secret Sharing Threshold Algorithms,” Computers and Mathematics with Applications, vol. 60, no. 2, 2010, pp. 267-271.

[18] J.S. Albus and A.M. Meystel, “Engineering of Mind: An Introduction to the Science of Intelligent Systems, Willey, 2001.

[19] F.Y. Leu and C.C. Ko, “An Automated Term Definition Extraction System Using the Web Corpus in the Chinese Language,” Journal of Information Science and Engineering, vol. 26, no.2, March 2010, pp. 505-525.

[20] F.Y. Leu, “An Inner and Incoming Intrusion Detection and Remote Protection System,” International Transaction on Computer Science and Engineering, vol. 55, no. 1, August 2009, pp. 81-96.

[21] R. Senkerik, Z. Oplatkova, I. Zelinka and D. Davendra, “Synthesis of Feedback Controller for Three Selected Chaotic Systems by Means of Evolutionary Techniques: Analytic Programming, Mathematical and Computer Modelling, vol. 57, Issues 1–2, January 2013, pp. 57–67.

[22] D. Yang and B. Yang, “A Novel Multi-factor Authenticated Key Exchange Scheme with Privacy Preserving,” Journal of Internet Services and Information Security, vol. 1, Issue 2/3, August 2011, pp. 44-56.

[23] Y.F. Huang, F.Y. Leu, C.H. Chiu and I.L. Lin, "Improving Security Levels of IEEE802.16e Authentication by Involving Diffie-Hellman PKDS," Journal of Universal Computer Science, vol. 17, no. 6, March 2011, pp. 891－911.

[24] S. Parkvall, E. Englund, M. Lundevall and J. Torsner, "Evolving 3G mobile systems: broadband and broadcast services in WCDMA," IEEE Communications Magazine, vol. 44, Issue 2, Feb. 2006, pp.30－36.

[25] Y.L. Huang and F.Y. Leu,“Constructing a Secure Point-to-Point Wireless Environment by Integrating Diffie-Hellman PKDS RSA and Stream Ciphering for Users Known to Each Other,” Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, vol. 2, no. 3, September 2011, pp. 96-107.

[26] Wiki, Exponential function, http://en.wikipedia.org/wiki/Exponential_function

[27] Wiki, Factorial function, http://en.wikipedia.org/wiki/Factorial_function

[28] S. Mary Ho and H. Lee, “A Thief among Us: The Use of Finite-State Machines to Dissect Insider Threat in Cloud Communications,” Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, vol. 3, no. 1/2, March 2012, pp. 82-98.

[29] Y. Lu and G. Tsudik, “Privacy-Preserving Cloud Database Querying,” Journal of Internet Services and Information Security, vol. 1, Issue 4, November 2011, pp. 5-25.

[30] F.Y. Leu, F.L. Jenq and F.C. Jiang, “A Path Switching Scheme for SCTP Based on Round

Trip Delays," Computers and Mathematics With Applications, vol. 62, Issue 9, November 2011, pp. 3504–3523.

[31] R. Whitaker, "Validation Examples of the Analytic Hierarchy Process and Analytic Network Process," Mathematical and Computer Modelling, vol. 46, Issues 7–8, October 2007, pp. 840–859.

[32] M.C. Lee, F.Y. Leu and Y.P. Chen, "An Adaptive Data Replication Algorithm based on Star-based Data Grids," Future Generation Computer Systems, vol. 28, issue 7, July 2012, pp. 1045–1057.

Appendix A

The six key exchange steps of the SSL are as follows.

Step 1. The client sends information, such as SSL version number, the list of encryption algorithms and a random number, called RNc, to the server.

Step 2. The server sends the chosen information, including the highest SSL version supported by both ends, the certificate of the server, the dedicated encryption algorithm, and a random number, called RNs, to the client.

Step 3. The client authenticates the server with the server's X.509 certificate. If authenticated, the client acquires the server's public key, generates the pre-master secret key encrypted with the server's public key, and then sends the ciphertext to the server.

Step 4. The server decrypts the ciphertext by using its private key and acquires the pre-master secret key generated by the client. The master secret key is calculated by using RNc, RNs and pre-master secret key on both the client and server sides.

Step 5. The client sends two messages including negotiation successful and handshake finished to the server.

Step 6. The server also sends two messages, including negotiation successful and handshake finished, to the client.

From now on, the client and the server have finished the handshake process and then established a secure connection to transfer the confidential data to each other through the Internet by using the master secret key.

Appendix B

The six steps of IKE main mode with pre-shared key are described as follows.

Step 1. The initiator generates a cookie, defines many pairs of the proposal, transforms payloads, and sends message1 to the responder.

Step 2. The responder generates a cookie, chooses the dedicated pair of the proposal, transforms the payload sent by the initiator, and sends message 2 to the initiator.

Step 3. The initiator generates a public key X_a by using the Diffie-Hellman algorithm and a nonce N_i, and sends message 3 to the responder.

Step 4. The responder generates the public key X_b by using the Diffie-Hellman algorithm and a nonce N_r, and sends message 4 to the initiator.

Now, the two peers have the same six parameters, including pre-shared key, cookies of initiator, cookies of responder, N_i, N_r, and the common secure key (CSK for short) calculated by using X_a and X_b with the Diffie-Hellman algorithm.

Step 5. The initiator generates three session keys named SKEYID_d, SKEYID_a and SKEYID_e, encrypts its identity and the hash value (for authentication) by using the SKEYID_e, and sends message 5 to the responder.

Step 6. The responder generates three session keys SKEYID_d, SKEYID_a and SKEYID_e by itself, decrypts message 5 to acquire the initiator's identity and the hash value by using the SKEYID_e, checks to see whether the hash value carried in message 5 is equal to the one calculated by itself or not. If yes, the initiator is authenticated. The responder further encrypts the identity by using the SKEYID_e, and sends message 6 to the initiator.

The initiator decrypts message 6 to acquire the responder's identity and the hash value by using the SKEYID_e, and checks to see whether the hash value received from the responder is equal to the one calculated by itself or not. If yes, the mutual authentication is successfully performed. The process of IKE main mode ends.