

國立交通大學

資訊科學與工程研究所

碩士論文

藍芽環境中基於通行碼的認證機制



On The Security of Password-based Pairing Protocol in
Bluetooth

研究生：范家銘

指導教授：謝續平 教授

中華民國九十八年六月

藍芽環境中基於通行碼的認證機制

On The Security of Password-based Pairing Protocol in Bluetooth

研究生：范家銘

Student : Chia-Ming Fan

指導教授：謝續平 博士

Advisor : Dr. Shiuhyng Shieh

國立交通大學

資訊科學與工程研究所

碩士論文

A Thesis

Submitted to Institute of Computer Science
College of Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in

Computer Science

June 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年六月

藍芽環境中基於通行碼的認證機制

研究生：范家銘

指導教授：謝續平 博士

國立交通大學

資訊科學與工程研究所

摘要

藍芽在現今短距離傳輸上扮演著重要的角色。無論是行動裝置或是固定不動的設備，只要運用藍芽，在短距內便能自由的傳輸資料。當兩台俱備藍芽功能的裝置欲互相連結時，需要先經過裝置配對的過程，從安全的角度來看，這個過程主要是認證彼此並建立一把秘密金鑰。本篇論文針對藍芽中基於通行碼認證金鑰交換協定提出安全分析，發現新的認證機制非常易於遭受通行碼猜測攻擊，而我們也明確的點出其設計方法若僅微幅調整將無法完全的抵禦。接著，提出一個能夠有效防禦猜測攻擊的協定，安全層級提升且訊息交換量也可降低。此外，我們正規的證明提出的協定在 random oracle model 下是安全的。此協定不改變傳統藍芽使用者的操作習慣，且計算複雜度低，適合替代藍芽現有的通行碼驗證機制。

關鍵字: 基於通行碼認證金鑰交換機制、藍芽認證機制、藍芽裝置配對

On The Security of Password-based Pairing Protocol in Bluetooth

Student: Chia-Ming Fan Advisor: Dr. Shiuhyng Shieh

Department of Computer Science

National Chiao-Tung University

Abstract

Bluetooth is a popular wireless communication technique, providing connection between portable or stationary devices in close range. A procedure called pairing needs to be performed when two devices intend to connect with each other in order to form a trusted pair and generate secret keys to protect the link. There are several modes of Bluetooth pairing, and password-based is the most convenient and prevalent way. In this paper, we pointed out a potential vulnerability in the password-based pairing protocol of the latest Bluetooth v3.0, which makes password guessing possible. To cope with the problem, a new scheme is proposed which can mitigate the network threats, and is compatible with the hardware of legacy Bluetooth devices. Not only heuristic analysis and its physical meaning will be provided, but also formal proof will be given to make sure that our proposed protocol is secure. Note that our modification does not affect Bluetooth users' custom, which makes it a suitable replacement for the new Bluetooth pairing protocol.

Keywords: password-based authenticated key exchange, Bluetooth pairing, Bluetooth PIN authentication

誌謝

碩班對我來說，真的是一段難忘的路。

歡笑淚水交織，跌跌撞撞

坦白說，一點都不後悔!! 我覺得我學到了好多東西!!

想要謝謝的人不盡其數，一直給我鼓勵加油、給我鞭策，也包容我的任性、我的
駑鈍。

感謝我的頭頭~謝續平老師，在兩年來訓練我的邏輯思考、表達能力、以及做學
問態度和方法，讓我成長了很多。

在我每每迷失方向的時候，都不吝給予最有力、最及時援手的紀偉和子逸學長，
你們的大恩大德真的讓小弟沒齒難忘。

謝謝 yellow、佳純、碩二的同儕們~ Vic 媽許 雨芊 和 kuro，常常被我纏著問
問題，也都不厭其煩的和我討論。也謝謝碩一的學弟們~ 花生 阿不拉 秋人 梅
子綠 和 艾倫。因為有你們，讓我覺得碩班的生活比較不苦悶，也從你們身上學
到了很多。感恩啦~

謝謝孝盈，在迫在眉睫的時候教我怎麼做正規證明，讓我迅速的吸收學習。

謝謝一路陪我歡笑聽我泣訴的緋聞女主角玫靜，謝謝豆腐心刀子嘴的明華學姊，
我知道你很關心我的>///
謝謝奇葩麥可教了我很多做事應該有的態度以及腥
羶色情的知識~ 謝謝翹助理順瑩姊跟我歡笑 陪我買飲料和鬆餅... 哈

謝謝球隊的大家~ 如果沒有打球的話，我的壓力真的很難宣洩。

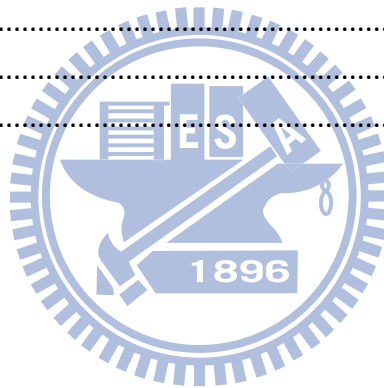
最後

感謝我的父母和老姊一路上都很支持我，尊重我做的每一個決定、I love you~

謝謝我的好麻吉們~ 蔡餅-佩江-舒晴-詩宜-振盛-瑛萱-616 兄弟們-胖威-一哥-
嘴砲布希-逼逼 等等 因為有你們的陪伴，我才更有繼續下去的勇氣和動力。

Table of Content

Chapter 1	Introduction	1
Chapter 2	Related Work	2
2.1	Authentication Issues of Bluetooth	3
2.2	Password-based Authenticated Key Exchange	4
Chapter 3	Bluetooth Pairing	5
3.1	Pre-2.1 version of Bluetooth Pairing.....	5
3.2	Current Pairing Protocol v3.0.....	6
Chapter 4	Proposed Scheme	18
4.1.	Design Principle: Randomness.....	18
4.2.	Protocol.....	20
Chapter 5	Security Analysis of the proposed scheme	25
5.1.	Heuristic Security Analysis	25
5.2.	Formal Security Analysis	26
Chapter 6	Comparison	37
Chapter 7	Conclusion.....	39
Chapter 8	Reference	40



List of Figures

Figure 3. 1 Piconet.....	6
Figure 3. 2 Public Key Exchange	10
Figure 3. 3 Authentication Stage 1	13
Figure 3. 4 Authentication Stage 2	14
Figure 3. 5 Link Key Calculation	14
Figure 4. 1 Proposed Authentication Stage 1.....	24
Figure 5. 1 Specification of the proposed scheme	32

List of Tables

Table 3. 1 Symbols	10
Table 4. 1 Symbols	21
Table 6. 1 Comparison of security issue	37
Table 6. 2 Comparison of performance issue.....	38



Chapter 1

Introduction

Bluetooth is a power-friendly wireless protocol, which is designed for short-range data transmission. Since utilizing the radio frequency (RF) technology, which can penetrate the obstacles, Bluetooth applications need not to be in line-of-sight with each other. In other words, devices can communicate with each other in different rooms as long as both parties can receive powerful enough signals.

Prevalent products in our lives, such as laptops, cell phones, hand-free headsets, printers, and digital cameras are often equipped with Bluetooth techniques. Short range data exchanging is often required in these applications. In addition, communication parties may not always be stationary, but portable devices. And Bluetooth is usually the common choice.

When two mobile devices intend to communicate with each other, an authentication and key agreement procedure is required to make a trusted pair. Current Bluetooth standard provides several ways for authentication, in the case that both devices have input but no output capabilities, or two devices cannot do pairing side by side, the most convenient and general way to form a trusted pair is using a shared secret, known as *Password Authenticated Key Exchange (PAKE)*.

PAKE problem is usually seen in the secure protocol that allows two participants to prove to each other that they indeed have the same password, and to derive a session key that would be used to build a secure channel. That is to say, it meets the mutual authentication and key exchange. Noted that the size of the pre-shared secret is pretty small and in a limited space since it required being human memorable (e.g. 6 decimal

digits). Hence, how to prevent this kind of protocols from guessing attack is a significant issue.

In this paper, we put our emphasis on the novel password authentication scheme of the current Bluetooth standard v3.0. We point out a fatal weakness that the adversary may learn the whole password easily with both off-line and on-line guessing attack and lead to impersonation. We also give a fully analysis on the protocol, and figure out the reason why it cannot completely defeat guessing attack. Our major goal is to strengthen the protocol with limited resources and keep most of the functions and parameters for fully compatible with the hardware of legacy Bluetooth devices. In our scheme, we give equal or even better security level while reducing the amount of message flows during authentication procedure. On the other hand, our scheme needs no extra device and software installed, and users can keep their operation behaviors. The complexity of our scheme does not grow much as well, and thus may be applied on other resource-constrained devices. Moreover, in order to make sure our proposed scheme is secure enough, we not only give heuristic security analysis, but also make rigorous proof using the mode of Ballare, Poincheval and Rogway[13].

The rest of the paper is organized as follows. Chapter 2 reviews other relevant research in the security issues of Bluetooth. Chapter 3 gives overview of the current Bluetooth Pairing Protocol. and points out the weakness of current standard. In Chapter 4 we demonstrate why the current standard cannot completely defeat guessing attack and propose a much more secure scheme with better performance. Security is evaluated in Chapter 5 and comparison is given in Chapter 6. Finally, we make conclusion in Chapter7.

Chapter 2

Related Work

In this chapter, we review the related research these years, including the Bluetooth security issues and password-based authenticated key exchange protocols. We also pointed out the main contribution and features of each study.

2.1 Authentication Issues of Bluetooth

Several studies for early version of Bluetooth are discussed in [3][4][6]. Shaked et al.[4] and Wong et. al [6] showed that how to crack the Bluetooth PIN by brute force. Jakobsson and Wetzel [3] pointed out the adversary could employ passive eavesdropping and man-in-the-middle attack. Most common suggestion is to use the larger size PIN during the authentication procedure and the modification of user behavior.

There are also researches that add the trusted third party or release public information to improve their authentication scheme. But the benefits of these approaches are not applicable to our environment.

Visual comparison is one of the novel methods of the additional channel, and there are several related research address its usability and security issues [25-27]. However, visual comparison is not prevalent and convenient enough then using a short password to authenticate, hence it is out of our discussion. [1][2] also maintain the password-based authentication but with creative modification, which is claimed to prevent man-in-the-middle attack and passive eavesdropping, but actually we found that one of its modification results in much vulnerability.

The works of National Institute of Standards and Technology (NIST)[8] and NOKIA [7] listed comprehensive Bluetooth vulnerabilities of each version, but not much of these properly dealt with the authentication and key exchange procedure, and

lacks a rigorous security proof as well.

2.2 Password-based Authenticated Key Exchange

Since the seminal work of password-based authenticated key exchange (PAKE) scheme proposed by Bellare and Merritt[9], there have been several extensive research [10-21]. Among those PAKEs, Bellare and Rogaway model the two-party and three-party key distribution with provable security, and a refined version that includes forward secrecy and dictionary attack is published in 2000[13]. In this model, player instances are modeled as oracle available to the adversary and several attacks are modeled by oracle queries. Our model is derived from [13].

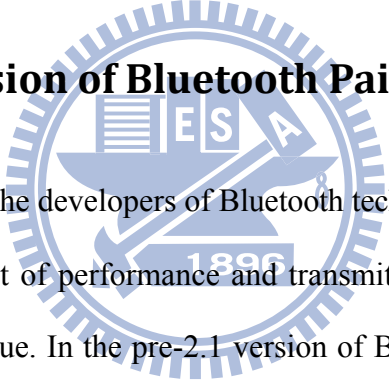


Chapter 3

Bluetooth Pairing

Bluetooth devices connected with each other forms a *piconet*, consisting of at least one *master* device and several *slave* devices. The device that initiates a connection by paging serves as *piconet master*, which conducts the initiating process to at most seven active slaves. Although only one device can be the master device for each piconet, time division multiplexing (TDM) allows a slave in one piconet to serve as a master for another piconet at the same time, which then formed a *scatternet*. We picture a *scatternet* that chained by three *piconets* in Figure 3.1.

3.1 Pre-2.1 version of Bluetooth Pairing



In the version 2.1 and 3.0, the developers of Bluetooth technology have not only made improvements on the aspect of performance and transmit speed, but also paid much attention on the security issue. In the pre-2.1 version of Bluetooth, the PIN (Personal Identification Number) is used to make sure the security of the pairing procedure, and is used to generate the link key as well. We give a practical example; in the case that two equipments (cell phones, for instance) intend to communicate using Bluetooth, each user are required to input the equivalent PINs into both devices by turns to authenticate each other and then generate a link key and encryption key for data transmission. Unfortunately, several researches had indicated the weakness of this security scheme, and the major reason is due to the limited size of PIN. The most common solution is to pick up a reasonable size PIN. However, due to the limitation of human memory, PIN cannot be much longer. Thus, using this kind of authentication scheme provides no protection of dictionary attack. On the other hand, since PIN is directly used to

generate the link key, it cannot provide forward secrecy. That is to say, if an adversary successfully cracks the PIN, he can calculate the link key in the previous session, and decrypts all the eavesdropped messages.

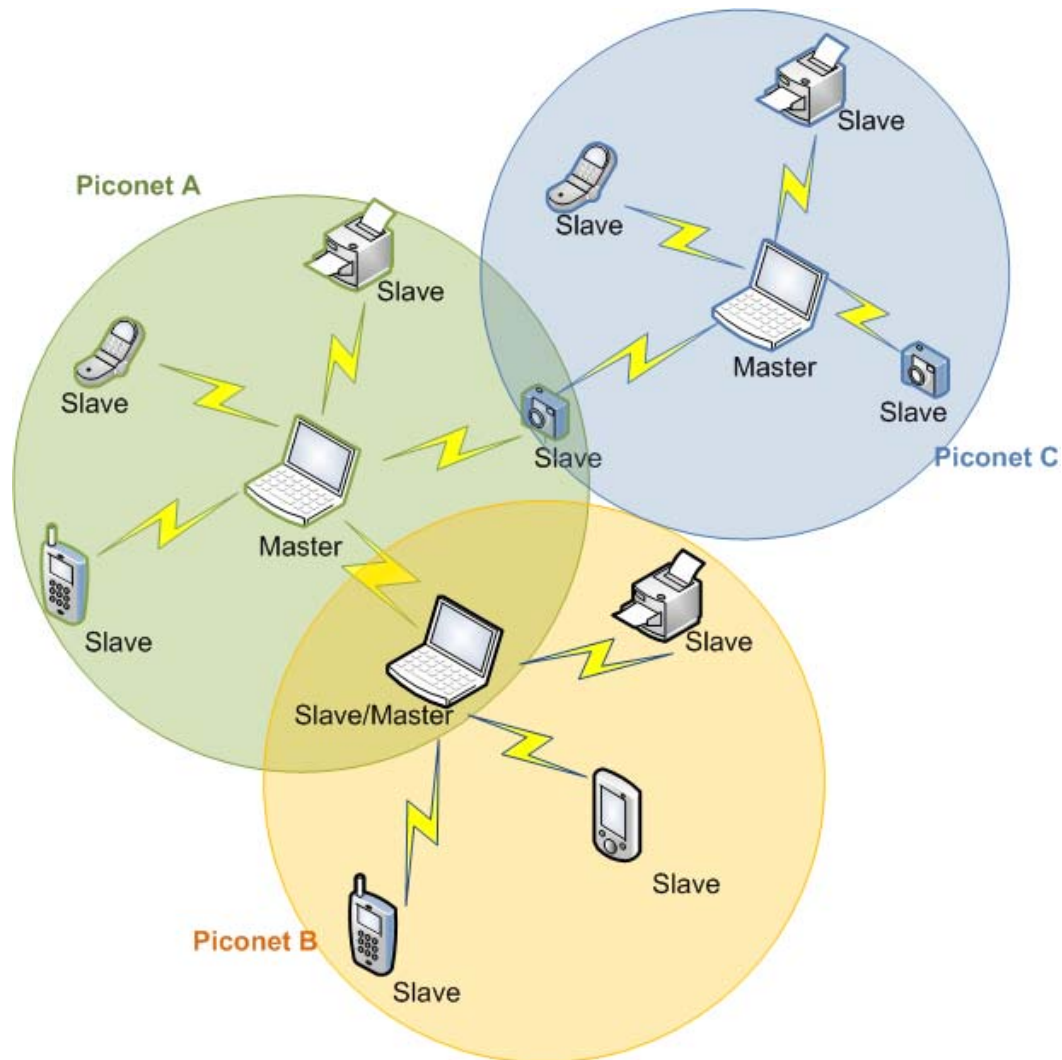


Figure 3. 1 Piconet

3.2 Current Pairing Protocol v3.0

Recently, The Bluetooth SIG (Special Interest Group), added a whole new pairing scheme in Bluetooth version 2.1[1] and also were remained in version 3.0[2], called *Secure Simple Pairing*, which employs Elliptic Curve Diffie-Hellman cryptography. Once two devices intend to connect with each other, public-private key pairs as well as

their input/output capabilities, unique addresses of devices, and several random nonces are required in order to generate the link key. One of the most important features is that even if the PIN is guessed by the adversary, the session key cannot be compromised.

3.2.1 Four association models

There are four association models in *Secure Simple Pairing*, classified according to I/O capabilities of devices. We briefly introduce these models in the following:

Numeric Comparison

This model is used in the situation that both devices have displays and “yes”/”no” buttons. A 6-digits number will show on the screen respectively in the end of the pairing procedure, and the user is expected to verify whether these two values are the same. The user may enter “yes” into both devices to confirm the pairing if the two values are equivalent. One of the important goals of this model is to prevent man-in-the-middle attack.

Just Work

This mode may be used where at least one device has no screen to display the number and no input capability either. This model greatly resembled to Numeric Comparison, but the difference is the number may not show on the display. The user may be asked to accept the connection without comparing the values. This model is usually used on head-set or public printer that would accept all requests as usual.

Passkey Entry

In the case that one device has keyboard, while the other one has no input capability but with a display. The number may show on the one device which has a screen and the user may inject these digits into the other device. The pairing succeed in condition that the

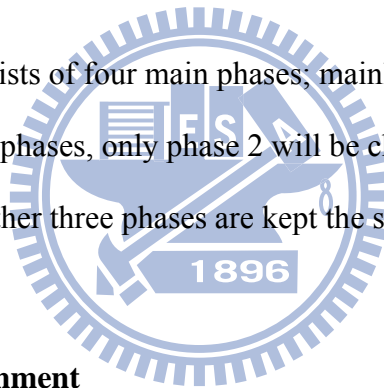
input values are match. *Passkey Entry* is also used in the situation that both devices have input but no output capability. Two participants input the pre-shared password into the device respectively, and the entire pairing protocol is to confirm whether these two values are the same.

Out-of-Band

This mode is suitable for the devices that support a channel other than Bluetooth (e.g. Near Field Communication). Devices could exchange the reliable data to make trusted pair via this channel.

3.2.2 Secure Simple Pairing

Secure Simple Pairing consists of four main phases; mainly establish an authenticated link key. Among these four phases, only phase 2 will be changed depends on different model mentioned above. Other three phases are kept the same no matter what model pairing protocol is using.



Phase 1 DHKey Establishment

Both devices generate their own Elliptic Curve Diffie-Hellman key pairs and send the public key to each other. Then they derived a Diffie-Hellman key (DHKey), which will be used in third and forth stages.

Phase 2 Authentication Stage 1

The main goal of this stage is to ensure that there is no man-in-the-middle attack occurs in the communication between both ends. And the procedure runs differently depends on the association model described in the previous section.

Phase 3 Authentication Stage 2

This step is for both sides to confirm they had successfully completed the exchanging

of key and nonce.

Phase 4 Link Key Calculation

The devices confirm pairing and derive a link key using the DHKey generated in *phase 1* and several information exchanged previously.

In this paper, we mainly focus on the *Passkey Entry Model*. And we run the entire Secure Simple Pairing with this specific model in the following. Notice that before the pairing procedure, two devices may obtain some information exchanged during the *device discovery* step to *service discovery* step, which is out of the scope of our discussion. We only list some of the information below:

- The unique Bluetooth device address
- The system clock of the partnered device
- The Bluetooth class of device
- The device name

To make the protocol more clear, we list the symbols in Table 3.1 based on the current standard.

Symbol	Definition
DHKey	Elliptic Curve Diffie Hellman Key
rx	Personal Identity Number of device X
Cx	Commitment value from device X
Cx _i	i th commitment value from device X
Nx	Unique random value from device X
Nx _i	i th unique random value from device X
PK _x , SK _x	Elliptic Curve Diffie Hellman public- secret key pair
IOcap_X	Input/output capabilities of device X
X	Unique Bluetooth address of device X
f1()	HMAC-SHA ₂₅₆ function used to generate commitment value Ca and Cb in Authentication Stage 1
f2()	HMAC-SHA ₂₅₆ function used to compute the link key
f3()	HMAC-SHA ₂₅₆ function used to compute check value in

	Authentication Stage 2
--	------------------------

Table 3. 1 Symbols

Phase 1: DHKey Establishment

Initially, each device produces its own Elliptic Curve Diffie Hellman public and secret key pair. This key pair can at any time be discarded and generate a new one for the will of the user. We denote the key pair of initiating device A as (PKa, SKa), and (PKb, SKb) for responding device B.

The *Secure Simple Pairing* procedure starts from the initiating device sending its public key to the responding device. And the responding device may replies its own public key back. After each device successfully obtains the public key of its partner, it computes the Elliptic Diffie Hellman Key (DHKey) where P192 shown in Figure 3.2 is the chosen Elliptic curve used in Bluetooth v3.0.

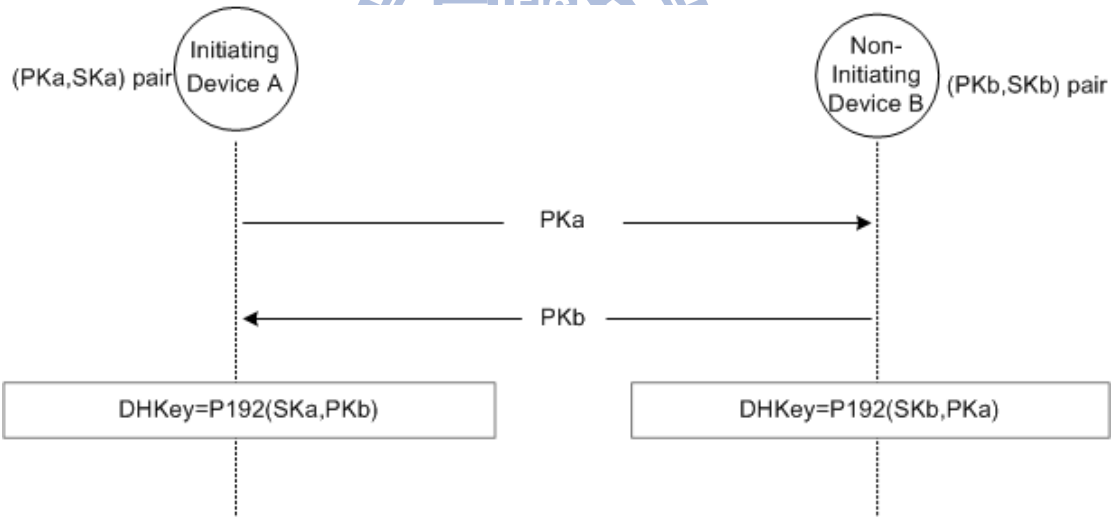


Figure 3. 2 Public Key Exchange

Phase 2: Authentication Stage 1

It is noteworthy that there is no mutual authentication in the first phase; any attacker can impersonate and deliver its own public key to the victim to generate a Diffie Hellman Key. In order to make trusted pair, both participants use a short pre-shared password (rx). In the current Bluetooth standard, rx is split into k pieces by bit, labeled rx_1, rx_2, \dots, rx_k . The authentication step should repeat k rounds with each bit. We

show the i^{th} round below.

Step 1 Devices A and B are respectively inputted PIN ra and rb . We defined $ra=ra_1|ra_2|...ra_k$ and $rb=rb_1|rb_2|...rb_k$.

Step 2 Each device chooses a 128-bits pseudo-random nonce, Na_i and Nb_i .

Step 3 Device A computes a commitment $Ca_i=fI (PKa, PKb, Nai, rai)$. By the same token, device B compute a commitment $Cb_i=fI (PKa, PKb, Nbi, rbi)$, where $fI ()$ is a function makes use of HMAC based on SHA-256.

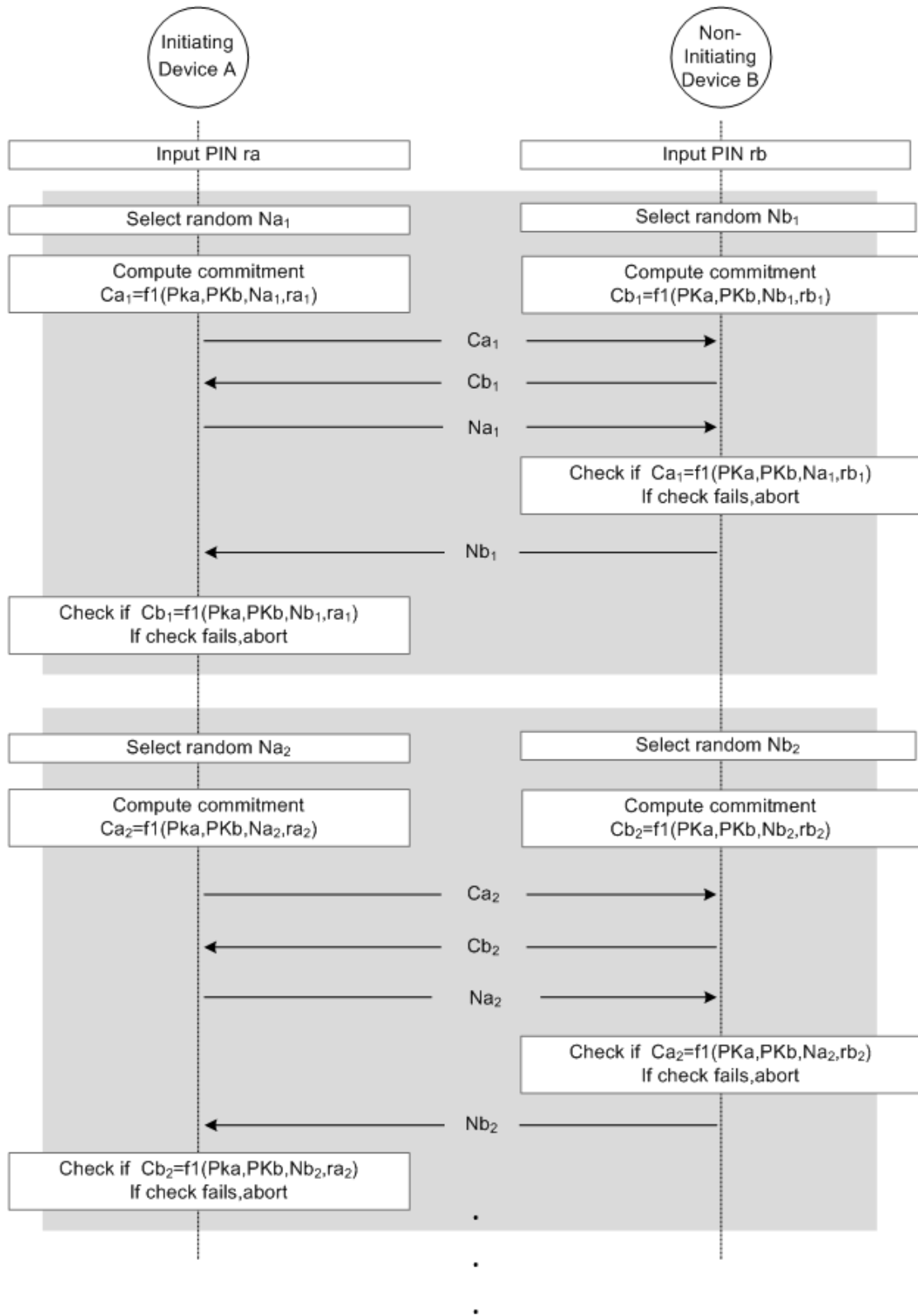
Step 4 Both parties then show their commitments to each other by turns. Moreover, the initiating device sends Nai generated in Step 2 to the other device.

Step 5 On receiving the above values, devices B checks if the received Ca_i is identical to $fI(PKa, PKb, Nbi, rbi)$, and responds Nbi to device A if the verification is correct, or terminate the procedure if check fails.

Step 6 Similar to the last step, device A checks if the received Cbi equal to $fI (PKa, PKb, Nbi, rbi)$, if check fails, then aborts.

Step 7 Step 2 to Step 6 should run k times, until all the bits of the PIN are authenticated.

Step 8 After completing k rounds, both devices set $Na=Na_k$ and $Nb= Nb_k$, which will be used in Authentication Stage 2.



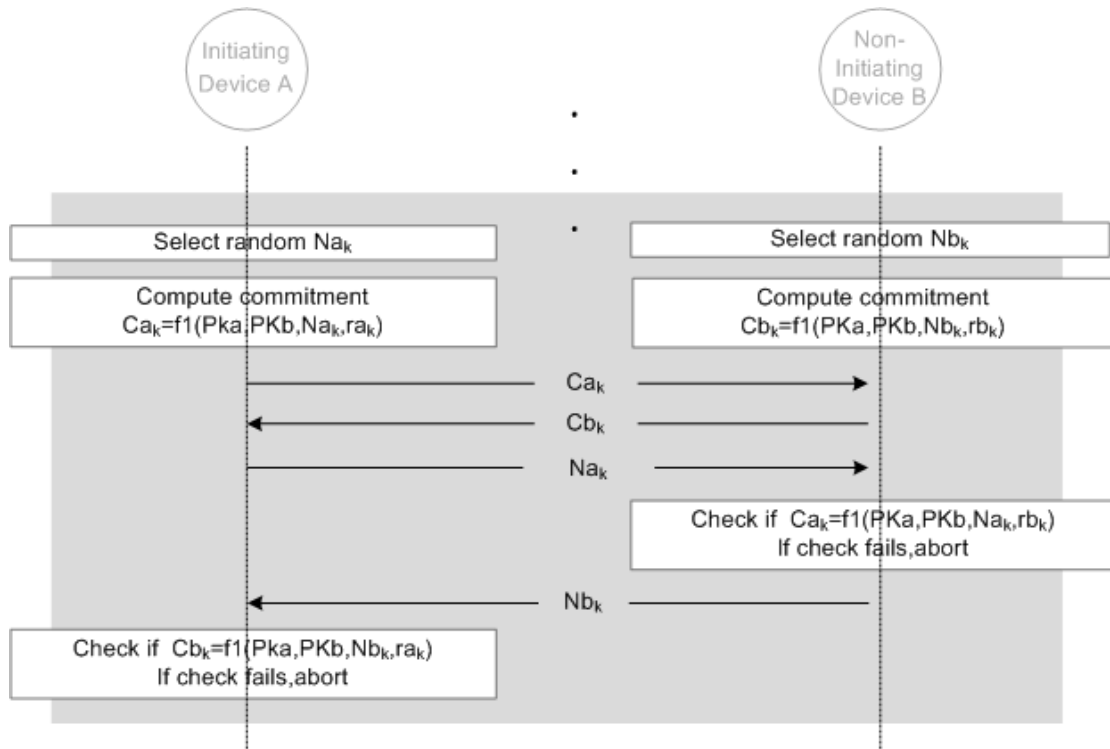


Figure 3. 3 Authentication Stage 1

Phase 3: Authentication Stage 2

After the *Authentication Stage 1*, comes to the second stage of authentication. This step is for both sides to confirm that they had successfully completed the exchanging of their public key and nonce in the previous step.

Step 1 Each device computes a new check value including a derived shared key (DHKey) and several value exchanged previously.

Step 2 The initiating device then sends its check value which is verified by the responding device. If the verification fails, the responding device aborts the procedure, otherwise, it replies with its check value.

Step 3 On receiving the check value, the initiating device checks if it is correct. If check fails, then the procedure afterwards should be canceled.

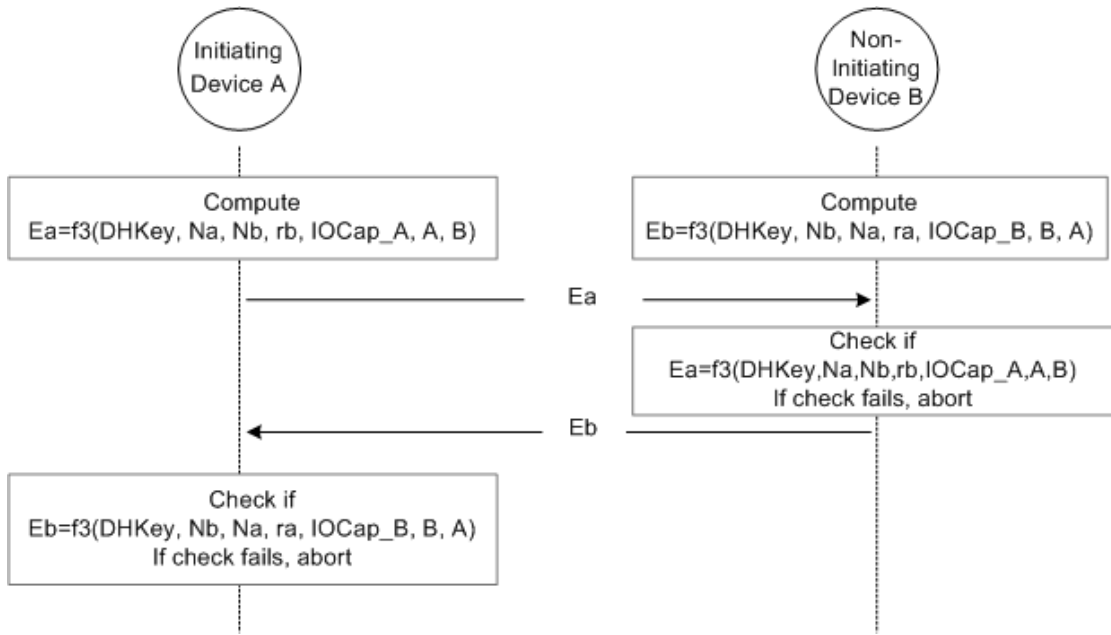


Figure 3. 4 Authentication Stage 2

Phase 4: Link Key Calculation

Whenever the pairing is confirmed, both devices then compute the link key using the derived shared key (DHKey), nonce, and Bluetooth device address. The nonce is used to make sure the freshness of the link key.

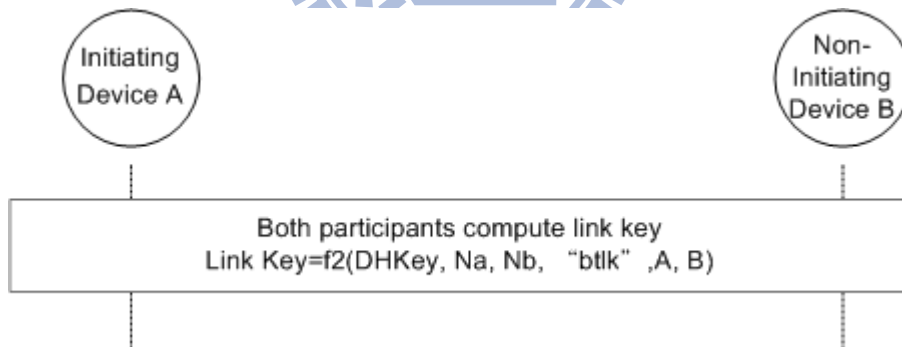


Figure 3. 5 Link Key Calculation

3.2.3 Weaknesses of Secure Simple Pairing

In this section, we analysis the weaknesses of Secure Simple Pairing protocol. The weaknesses can be divided into two categories, namely, security vulnerabilities and efficiency weaknesses. We will describe the security vulnerabilities and efficiency weaknesses in the follows.

A) Security Vulnerabilities

Although Secure Simple Pairing paid much attention on security issues, several security weaknesses are discovered, including passive off-line guessing attack and active on-line guessing attack. With these attacks, the adversary can impersonate a honest user with ease. The attacks are described as follows.

Off-line guessing attack

Dissimilar from the pre-2.1 version which uses the whole digits (usually 4-digits) during the authentication at a time, the PIN in the passkey model of Secure Simple Pairing reveals only single bit in each round. This modification is called “gradual disclosure” in the Bluetooth standard, and it claims to defeat man-in-the-middle attack. However, such kind of method results in much more vulnerable for the attacker to learn the whole string of password without great effort. We address a simple scenario in the following steps.

Step 1 In the case that an adversary, say D_M eavesdropped the pairing communication with passkey model between two honest devices, named D_A and D_B . Assumed that the PIN is k bits long, and the following values may be captured after the first two stages,

$$\{PK_a, PK_b, Ca_1, Cb_1, Na_1, Nb_1\}$$

$$\{PK_a, PK_b, Ca_2, Cb_2, Na_2, Nb_2\}$$

...

$\{PK_a, PK_b, Ca_k, Cb_k, Na_k, Nb_k\}$

Step 2 Recall that $Ca_i = fl(PK_a, PK_b, Na_i, ra_i)$ and $Cb_i = fl(PK_a, PK_b, Nb_i, rb_i)$, where fl is a HMAC function based on SHA-256 known by every Bluetooth v3.0 supported devices.

Step 3 The attacker picks up a serial of elements sent from one side, $\{PK_a, Ca_1, Ca_2, \dots, Ca_k, Na_1, Na_2, \dots, Na_k\}$ for example.

Step 4 The attacker then computes a commitment Ca_i' with $ra_i = 0$. If Ca_i' is identical to Ca_i , it indicates that the guessing bit is correct, otherwise $ra_i = 1$. The PIN will be learned in the final round by assembling ra_1, ra_2, \dots, ra_k .

Once the password between D_A and D_B was learned, D_M can easily impersonate D_A in some time with its own public-private key pairs.

On-line guessing attack

The authentication procedure is not only vulnerable to off-line guessing attack, but also cannot defeat on-line guessing attack efficiently. We address this kind of attack in the scenario below. The attacker intends to impersonate a trusted user with the victim.

Step 1 The attacker exchanges its public key with the victim, and then derived a DHKey in the first phase.

Step 2 During the *Authentication Stage 1*, since the attacker has no pre-shared PIN with the victim, he can make a guess and sets ra_1 to be "0", and computes a commitment which is checked by the victim.

Step 3 If the procedure aborts, the attacker may learn that ra_1 is guessed incorrectly and set it to "1" instead. Otherwise, ra_1 is confirmed to be "0".

Step 4 The attacker may repeat step2 to step 3 for k times until all the bits of PIN $(ra_1, ra_2, \dots, ra_k)$ are obtained.

To defeat the repeated attempts during the authentication, the Bluetooth standard offers a solution: for each authentication failure, the waiting interval shall be increased exponentially. However, since the default length of the password in Bluetooth is at most 20 bits, and each bit can be learned at most two trials, hence the attacker can obtain the entire PIN after at most $2*20$ trials.

B) Performance Weakness

In *Passkey Entry Model* of Secure Simple Pairing, we found that there are k rounds authentications in the second phase, where k is the length of PIN in bit. Each round consists of four messages exchanged and two times of Hash function computed in each device. Suppose that $k=20$, then we have to cost 80 messages exchanged and each device needs to compute commitments total 40 times. For the resource limited devices, this is not an efficient way.

Chapter 4

Proposed Scheme

As we mentioned in Chapter 3, *Numeric Comparison* is a very strong mode that defeats man-in-the-middle attack. Thus, we suggest that if both devices have input and output capabilities, *Numeric Comparison* mode should be the best choice. However, our proposed scheme is suitable in the case that at least one of the two devices has no output capability. Besides, when two participants cannot make pair side by side, our scheme is also the most convenient way to make trusted pairs. Note that in our scenario, the password will not be changed so often. For example, the password that used in the home network or in an office. In this chapter, we first demonstrate the reason why the current Bluetooth scheme cannot completely defeat off-line guessing and point out the principle that to defeat the threats. Then we elaborate our proposed scheme.

4.1. Design Principle: Randomness

We survey a large amount of password-based Key Exchange protocol and found a common characteristic. In the case that given some messages by eavesdropping, the adversary can make a guess of the short password and verify whether the guessing is correct, such kind of protocol is defined as vulnerable to off-line password-guessing attack. On the contrary, if the adversary cannot get any advantage from just guessing the password, the protocol is defined strong enough to defeat off-line password-guessing.

We found that the most effective way is to add “randomness” into the authentication procedure, and adding a random nonce is a common solution.

Several password-based authentication protocols are added a random nonce into the computation, not only prevent replay attack, but also increase the security level against guessing attack. Among the authentication procedure, the short password is combined with the random nonce, and the mixed value then sent out to its partner to verify. Since the nonce is not transmitted in plaintext but hidden by some specific way before sending out, the passive attacker and even its partner cannot learn it. Therefore, anyone else cannot get any benefit by only guess the short password, and thus the scheme can protect the protocol from guessing attack. Note that we only put emphasis on off-line guessing attack, since on-line guessing can be easily defeated by taking the appropriate waiting interval between each trail.

We define a rule for the above condition, and list preliminaries first as following:

- $Z(j_1, j_2)$: represents any non-segregate function that combine two input value j_1 and j_2
- pw : the low-entropy password
- l : the size of the password in bit
- N : a random nonce

Rule 1 In the case that mutual authentication is achieved only according to the shared short password pw , off-line password-guessing attack can be defeated by adding a hidden randomness N that combined with the password by a non-segregate function, $Z(pw, N)$. Otherwise, the attack can successfully exploited by only $2^{|PW|}$ trials, where $|PW|$ is the length of the password in bit.

We take Bluetooth v3.0 pairing protocol for example. In the previous chapter, we found that the gradual disclosure is the major reason that makes the off-line and on-line guessing attack possible. Even if there exists random nonces, but they are sent in plaintext during the authentication. Therefore, these random values may not provide any extra protection but prevent replay attack

In order to fully compatible to the legacy system, we firstly analyze if it is possible to maintain all the functions and parameters but only give little modification. For example, we take the pre-v2.0 version pairing protocol into consideration, that using the entire password into computation. We simplify the procedure as follows,

Step 1 The initiating device A sends out a challenge Na combined with the short password pw_A (ex. keyed-hash and take one of the two value as the key), and the plaintext Na as well to the responding device B . Similarly, device B sends out a challenge Nb combined with the short password pw_B

Step 2 On receiving the two values, the responding device B then make a check. If check fails, then aborts the procedure, otherwise, device B replies its challenge Nb in plaintext.

Step 3 While device A receives the random nonce Nb , it makes a check as well. If check fails, then abort the procedure.

Obviously, we found that the protocol violates *Rule 1*. Although the random nonce is combined with the short password in hash function, it is disclosed during the protocol. Since the authentication takes the advantage of HMAC, a device is required to send out the input parameter in plaintext to its partner to make verification. Therefore, the adversary who intends to exploit off-line password guessing will easily succeed; he only need to guess l bits where l is the size of pw in bit. The number of trial is at most 2^l .

4.2. Protocol

Before we elaborate our scheme, we first state the assumption of the environment and the device as below.

- We keep most of the functions, parameters in *Secure Simple Pairing* in order to compatible to the current Bluetooth version. The devices does not need to install extra software and the operations does not required any changed.
- The information exchanged before pairing procedure, such as I/ O capability, device address, is adequate secure.
- There is neither server side, nor trusted third party in our Bluetooth environment.
- Password is still required to be human-memorable.

The design goals of our proposed scheme are listed below. Once we meet these certain goals, our proposed scheme can be said to be secure and efficient.

- Defeat password guessing attack,
- Mutual authentication between both two participants,
- Establishment of a link key upon successful authentication,
- Freshness assurance to the user of the established link key,
- Reduce the bandwidth and computational consumption.

For conciseness, when we describe our proposed scheme, we will use symbols listed in Table 7.1.

Symbol	Definition
DHKey	Elliptic Curve Diffie Hellman Key
PIN _x	Personal Identity Number of device X
N _x	Unique random value from device X
PK _x ,SK _x	Elliptic Curve Diffie Hellman public- secret key pair
X	Unique Bluetooth address of device X

Table 4. 1 Symbols

To meet the designed goals mentioned above, we propose a new authentication procedure for the *Passkey Entry model* of Secure Simple Pairing.

We found that the “gradual disclosure” mentioned in previous section is the main flaw leading to both on-line and off-line guessing attack. The attacker even needs not to brute force with great effort, but just learn each correct bit of PIN by at most two

attempts. To protect our scheme from guessing, we decide to cancel this kind of authentication procedure. However, recall that using the entire PIN into computation, as the scheme of early version, is still vulnerable to off-line guessing attack as we demonstrate in the previous chapter.

We take the constraint environment into consideration, such as no CA, server, and any public information. The most effective modification is to increase the size of the password used in the authentication phase. Nevertheless, we can not violate the assumption that the human-memorable password cannot be too large.

Therefore, we decide to take advantage of the long DHKey, which is derived in the first phase of the original pairing protocol. DHKey can only be obtained by the two participants and its size is up to 192 bits. We design a new authentication scheme using the entire PIN merged with the long DHKey, and run as a challenge-response scheme. In addition, we transmit the random nonce through an encrypt function, instead of the HMAC function before. These ideas will make the guessing number rise enormously. The proposed scheme is illustrated in Figure 4.1, and the messages are explained in detail as following,

Step 1 The initiating devices A and the responding device B are respectively entered 6-digits PIN_A and PIN_B . The PIN is then combined with the long DHKey which derived in the previous step, denoted as TK_A and TK_B . The initial device sends out an initial message to start the authentication protocol.

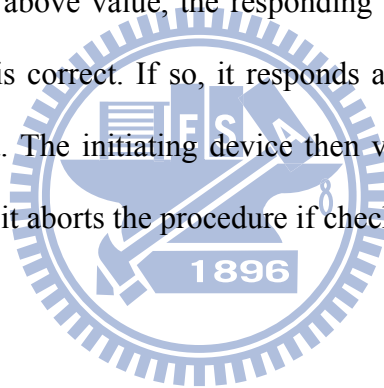
Step 2 The initiating devices A and the responding device B are respectively entered 6-digits PIN_A and PIN_B . The PIN is then combined with the long DHKey which derived in the previous step, denoted as TK_A and TK_B . The initial device sends out an initial message to start the authentication protocol.

Step 3 The initiating devices A and the responding device B are respectively entered 6-digits PIN_A and PIN_B . The PIN is then combined with the long DHKey which derived in the previous step, denoted as TK_a and TK_b . The initial device sends out an initial message to start the authentication protocol.

Step 4 On receiving the message, the responding device replies with a random nonce, N_b , encrypted by TK_b .

Step 5 The initiating device then decrypt the received message and get N_b . Afterward, it send back a encrypted value that composed of a random nonce N_a and N_b .

Step 6 On receiving the above value, the responding device make decryption and checks if the received N_b is correct. If so, it responds a hash of N_a . Otherwise, the procedure will be canceled. The initiating device then verifies whether the received value $H(N_a)$ is correct, and it aborts the procedure if check fails.



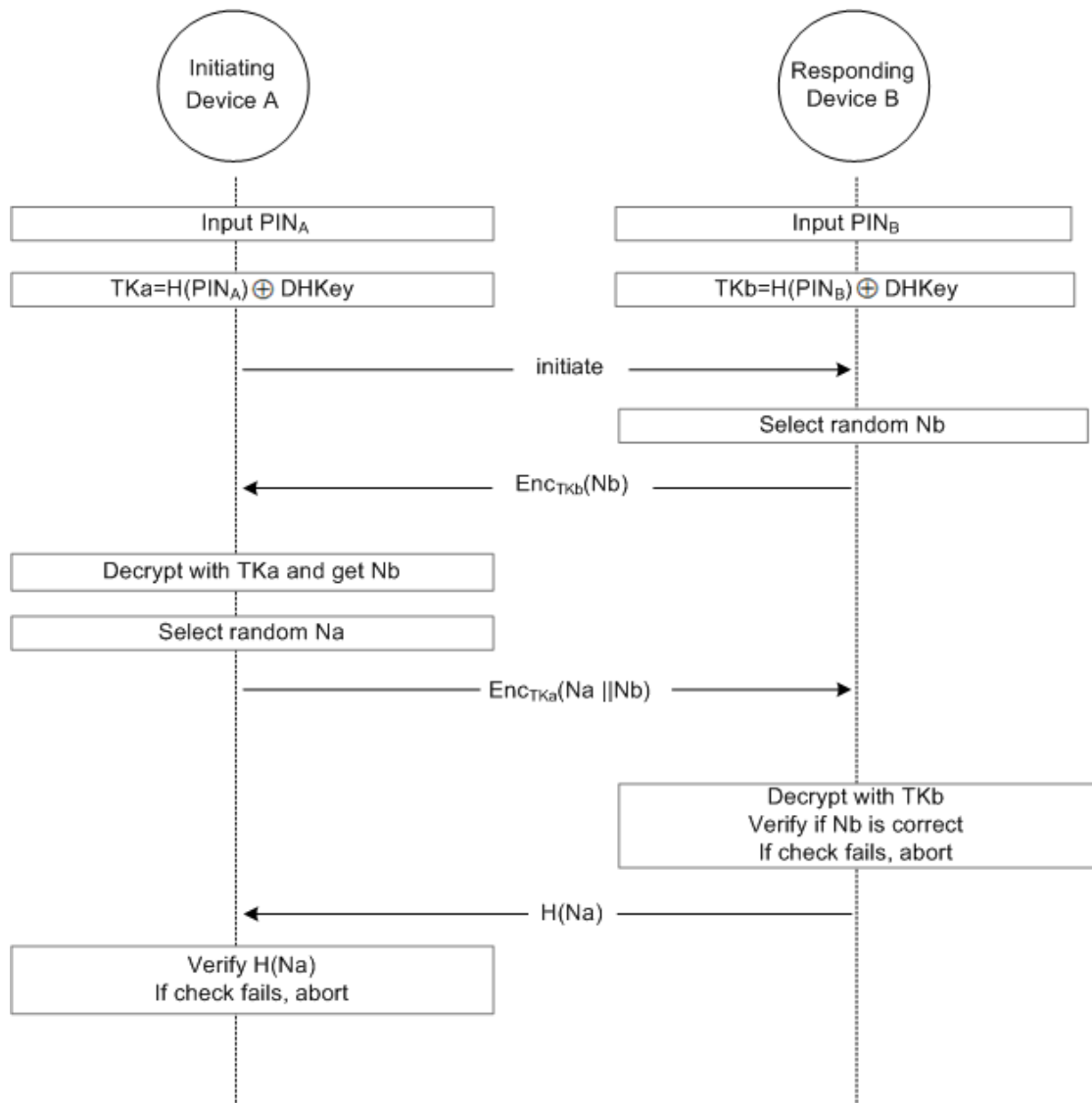


Figure 4. 1 Proposed Authentication Stage 1

Chapter 5

Security Analysis of the proposed scheme

In this chapter, we demonstrate the heuristic security analysis in section 5.1 and formally prove in section 5.2 to make sure the proposed scheme can withstand several known attacks. Our formal analysis and the communication model is based on the work of Chang et al.[10], and the work of Ballare, Poincheval and Rogaway's model[13].

5.1. Heuristic Security Analysis

In this section, we check our protocol heuristically by analyzing whether it can defeat some well known attacks, including on-line and off-line password guessing, man-in-the-middle and replay attack.

- **Security against On-line Guessing Attack**

Noted that we emerge the PIN with a long DHKey together, and add a long nonce as well in the proposed scheme. Although DHKey is not authenticated and may established by anyone, the random and hidden long nonce may help to make the total guessing number rises enormously. Recall that in the Bluetooth standard, the waiting interval between each attempt failure increases exponentially, and this will give effective protection.

- **Security against Off-line Guessing Attack**

The password-based protocol is vulnerable to off-line guessing attack due to the size of the shared key is too small. The adversary may gather information among the honest execution, and give repeated guessing by brute force. In our proposed protocol, we combined the PIN and DHKey together and an encrypted random nonce into the mutual authentication procedure. The passive attacker cannot get any advantage from only guessing the password.

- **Security against Man-in-The-Middle attack**

The combination of DHKey and password shared among Alice and Bob is used to defeat man-in-the-middle attack. Whenever Eve intends to impersonate one of the two honest participants, she needs to input the correct password to pass the authentication phase. Otherwise, the protocol will be aborted.

- **Security against Replay Attack**

Consider a situation that Eve intercepts the communication between Alice and Bob by eavesdropping; she then initiates a new session with Alice or Bob later (we take Bob for example). After generating the DHKey with the victim, Eve may send the previously intercepted message to Bob in order to try to deceive Bob that he is connecting with Alice. However, the second step of *Authentication Stage 1* requires a random nonce generated by its partner, and thus, this attack may not happen.

5.2. Formal Security Analysis

In this section, we analyze our protocol in a formal way. We first list some security requirements, and then prove our protocol to meet these properties. The three requirements are listed below:

- **Freshness assurance:** This requirement ensures that the authenticated link key cannot be established with the replayed messages.
- **Mutual authentication:** This requirement ensures that both participants communicate with the correct partner. And both participants believe that they are the only one that possesses an authenticated session key.
- **Guessing attack resistance:** This requirement ensures that the possibility for an attacker to successfully employ on-line and off-line password guessing is negligible.

Note that there are several models that can be used for formal proof, such as GNY, SVO, VO, AT, BPR model. However, only BPR model can be used to prove our protocol to meet “guessing attack resistance”, which is regarded as the essential property of password-based authentication protocol. Thus, we adopt the BPR model to accomplish our proof.

We first describe the model including the characteristics of participating entities and the capabilities of the adversary. Then we make the definition of goals within this model. Afterward, we give the description of the protocol in Figure 5.2, and finally prove the protocol satisfies goals.

(A) Security Model

In our model, the adversary \mathcal{A} is given great capabilities. It can control all communications among participants. All terms, including *Protocol Participants*, *Long-Lived Key*, *Oracle States*, *Session ID*, *Partner ID*, and *Oracle Queries* used in the model are formally defined as follows.

Participants: Protocol P enables two participants authenticate and establish a link key. A participant may have many instances, called *oracles*, involved in distinct, probably concurrent executions of protocol P . We denote the instant i of a participant A as Π_A^i .

Long-Lived Keys: Two parties share the symmetric password pw , which defined as the *long-lived key* with each other. Notice that the pw is chosen randomly from a bounded set, and thus can be found in polynomial time.

Session IDs: The session ID (SID) for oracle is a unique name of the session. We

define SID (Π_A^i) as the concatenation of all flows exchanged with oracle Π_A^i .

Partner IDs: Partner ID (PID) is indicated the other oracle communicated with. For example, we denote the oracle Π_A^i communicates with another one B as $PID(\Pi_A^i)=B$. It is noteworthy that the SIDs and PIDs are public information, and thus available to adversary

Oracle State: An oracle Π_A^i accepts, denoted as $ACC(\Pi_A^i) = true$, when it has enough information to compute the session key (SK). At any time the oracle can accept messages and it accepts at most once in a single execution. As soon as the Π_A^i accepts, the *SID*, *PID* and *SK* are defined. When an oracle sends or receives the last message of the protocol, receives an invalid message, or misses an expected message, the state will be set to terminated, denoted as $TERM(\Pi_A^i)=true$. Once the oracle Π_A^i is terminated, it will not send out any more messages.

Oracle Queries: The capabilities of the adversary are modeled by oracle queries. During the execution of the protocol, the adversary has endless supply of oracles and is able make various queries to them to model the possible attacks. We show these available queries as follows,

- **Send (Π_A^i, m):** This query sends message m to oracle Π_A^i , which models the active attack. The oracle would compute the response in processing the message m and send back to the adversary. A query $Send(\Pi_A^i, start)$ initializes the key exchange algorithm.

- **Reveal (Π_A^i):** This query models the misuse of the session key. The oracle Π_A^i returns the session key SK to the adversary if it has accepted and actually hold the session key.
- **Execute (Π_A^i, Π_B^i):** This query models the adversary get access to the honest communication (all message flows) between the two oracles Π_A^i and Π_B^i by eavesdropping. This query may at first seem useless, since the adversary can already obtain an honest execution among oracles using **Send** queries. Yet, it is essential for properly dealing with passive guessing attacks.
- **Test (Π_A^i):** This query models the semantic security of the session key. \mathcal{A} can only make at most one Test query and only available if the instance A is Fresh (defined in the next section) in that the session key is not obviously known to the adversary. On receiving this query, the oracle Π_A^i flips a coin $b \in \{0,1\}$. If $b=1$, then SK is released to the adversary. If $b=0$, a random value with length $|\text{SK}|$ is returned.
- **Hash (m):** In the ideal hash model, the adversary \mathcal{A} get the hash value by making queries to a random oracle. On receiving the query, the random oracle will check the record set, *H-table*, to find out whether m has been queried before. If so, it will then send back the previously computed result; otherwise, it generate a random value r and returns to \mathcal{A} , and store (m,r) into the *H-table*.
- **Encrypt (k,m):** In the ideal cipher model, the adversary get the corresponding ciphertext by sending the encryption query. On receiving the query, the oracle will check the record *E-table*, to find whether (k,m,c) existed. If so, the oracle will then reply the ciphertext c ; otherwise, a random ciphertext c will returns to \mathcal{A} , and store the (k,m,c) into the *E-table*.
- **Decrypt (k,c):** In the ideal cipher model, the adversary get the

corresponding result by sending the this query. On receiving the query, the oracle will check the record E -table, to find whether (k,m,c) existed. If so, the oracle will then reply the plaintext m ; otherwise, it returns a random value to \mathcal{A} , and store the (k,m,c) into the E -table.

Partnering: In our proposed protocol, we say that two oracles Π_A^i and Π_B^j are partnered if both oracles have been accepted and the following conditions hold:

- (1) $SK(\Pi_A^i)=SK(\Pi_B^j)$,
- (2) $SID(\Pi_A^i)\cap SID(\Pi_B^j)\neq \emptyset$,
- (3) $PID(\Pi_A^i)=B$ and $PID(\Pi_B^j)=A$,
- (4) No other oracle accepts $SK=SK(\Pi_A^i)=SK(\Pi_B^j)$.

Freshness: An oracle Π_A^i is said to be *fresh* if the following conditions are satisfied:

- (1) Π_A^i has been accepted
- (2) Neither Π_A^i nor its partner have been asked for a Reveal query.
- (3) No oracle has been asked for a Corrupt query before Π_A^i is accepted.

AKE Security: In our protocol P , we say that the adversary \mathcal{A} wins if she asked a single **Test** query to a fresh oracle and correctly guesses the bit b used in the game. We denote the AEK Advantage of \mathcal{A} as Adv_P^{AKE} ; the advantage is taken over all bit tosses. The advantage of \mathcal{A} in the game is given by $Adv_P^{AKE} = 2Pr(\text{win})-1$, and protocol P is AKE-secure if Adv_P^{AKE} is negligible.

(B) Security Assumption

Our security proof is based on the Elliptic Curve Diffie-Hellman (ECDH) assumption. Let \mathbb{G} be a cyclic finite group of prime order q with G as generator. A (t,ε) -ECDH attacker in \mathbb{G} is a probabilistic machine \mathcal{B} running within time t that given random (aG, bG) to compute abG with probability at least $\varepsilon = \text{Succ}_{\mathbb{G}}^{\text{ECDH}}(\mathcal{B})$. We denote that $\text{Succ}_{\mathbb{G}}^{\text{ECDH}}(t)$ is the maximal success probability running within time t . The ECDH-Assumption states that $\text{Succ}_{\mathbb{G}}^{\text{ECDH}}(t) \leq \varepsilon$ for any t/ε not too large.

In Figure 5.2, we describe the initial state and how the oracles in our protocol behave in response to the oracle queries. We omit the detail of our protocol here since it is clearly described in Chapter 4.

Initialization	<ul style="list-style-type: none"> ● Elliptic Curve public-private key pair (SK_A, PK_A), (SK_B, PK_B) are pre-installed or generated by the device. ● Assume that both participants have already exchange the I/O capability and address during the device discovery procedure. We denote the I/O capability of X as $IOcap_X$, and the address of device X as X ● $ACC(\Pi_A^i) \leftarrow \text{false}$, ● $TERM(\Pi_A^i) \leftarrow \text{false}$, ● $SK(\Pi_A^i) \leftarrow \text{null}$, ● $SID(\Pi_A^i) \leftarrow \text{null}$, ● $PID(\Pi_A^i) \leftarrow \text{null}$.
Execute (Π_A^i, Π_B^j)	
<ol style="list-style-type: none"> 1. $Send_1(\Pi_A^i, \text{start})$ //the initiating device will run the first phase while receiving $msg_out_1 \leftarrow PK_A$; //the start message return msg_out_1; 2. $Send_2(\Pi_B^j, m_1)$ //Π_B^j receive the first message in phase 1 $\langle M_1 \rangle \leftarrow m_1$; $DHKey = M \cdot SK_B$; $TK_B = H(PIN_B) \oplus DHKey$ $msg_out_2 \leftarrow PK_B$; return msg_out_2; 3. $Send_3(\Pi_A^i, m_2)$ //Π_A^i receive the first message in phase 1 $\langle M_2 \rangle \leftarrow m_2$; $DHKey = M \cdot SK_A$; $TK_A = H(PIN_A) \oplus DHKey$ $msg_out_3 \leftarrow \text{"initial"}$ return msg_out_3; 4. $Send_4(\Pi_B^j, m_3)$ //Π_B^j received the first message in phase 2 $\langle M_3 \rangle \leftarrow m_3$; if $M_3 = \text{"initial"}$, $N_b \xleftarrow{R} Z_q^*$, $msg_out_4 \leftarrow Enc_{TK_b}(N_b)$; return msg_out_4; else $TERM(\Pi_B^j) \leftarrow \text{true}$; 	

```

5. Send5( $\Pi_A^i, m_4$ )
   <M4>  $\leftarrow m_4$ ;
   Nb  $\leftarrow$  DecTKa(M4);

   Na  $\xleftarrow{R}$  Zq*;

   msg_out4  $\leftarrow$  EncTKa(Na || Nb);
   return msg_out4;

6. Send6( $\Pi_B^j, m_5$ )
   <M5>  $\leftarrow m_5$ ;
   <Na, Nb'>  $\leftarrow$  DecTKa(M5);
   if Nb' = Nb;
     msg_out5  $\leftarrow$  H(Na);
     return msg_out5;
   else TERM( $\Pi_B^j$ )  $\leftarrow$  true;

7. Send7( $\Pi_A^i, m_6$ ) //  $\Pi_A^i$  receive the first message in phase 3
   <M6>  $\leftarrow m_6$ ;
   If H(pwa, Nb, DHKey, PKA, PKB) = M6;
     Ea  $\leftarrow$  H2(DHKey, Na, Nb, pwb, IOcapA, BD_ADDRA, BD_ADDRB);
     msg_out6  $\leftarrow$  Ea;
     return Ea;
   else TERM( $\Pi_A^i$ )  $\leftarrow$  true;

8. Send8( $\Pi_B^j, m_7$ ) // generate the session key
   <M7>  $\leftarrow m_7$ ;
   if H2( DHKey, Na, Nb, pwb, IOcapA, BD_ADDRA, BD_ADDRB) = M7
     Eb  $\leftarrow$  H2(DHKey, Nb, Na, pwa, IOcapA, BD_ADDRB, BD_ADDRA);
     msg_out7  $\leftarrow$  Eb;
     return msg_out7;

     SK( $\Pi_B^j$ )  $\leftarrow$  H3(DHKey, Na, Nb, BD_ADDRA, BD_ADDRB);
     SID( $\Pi_A^i$ )  $\leftarrow$  m1, msg_out2, m3, msg_out4, m5, msg_out6, m7
   else TERM( $\Pi_B^j$ )  $\leftarrow$  true;

9. Send9( $\Pi_A^i, m_8$ ) //end of the protocol
   <M8>  $\leftarrow m_8$ ;
   if H2( DHKey, Nb, Na, pwa, IOcapA, BD_ADDRB, BD_ADDRA) = M8;
     SK( $\Pi_A^i$ )  $\leftarrow$  H3(DHKey, Na, Nb, BD_ADDRA, BD_ADDRB);
     SID( $\Pi_A^i$ )  $\leftarrow$  msg_out1, m2, msg_out3, m4, msg_out5, m6, smg_out7;

```

Figure 5. 1 Specification of the proposed scheme

5.2.1 Security Proof

In this section, we prove our protocol is secure in the random oracle using ECDH assumption. We use the model mentioned before and accomplished the proof with the AKE security goal. And the protocol is said to be secure if the advantage for an adversary is negligible.

Theorem 1. Let \mathcal{A} be an adversary against the AKE security of our proposed protocol P within time t , after q_{se} Send queries and q_h Hash queries. Then we have

$$\text{Adv}_P^{\text{AKE}}(t, q_{se}, q_h) \leq \frac{q_{se}}{2^\ell + k} + q_{se} q_h \text{Succ}_{\mathbb{G}}^{\text{ECDH}}(t')$$

Proof: We found that there are two ways that might lead our protocol to be attacked the AKE security. First, \mathcal{A} might obtain the long lived key by mounting on-line or off-line password guessing attack. Second, \mathcal{A} might break the ECDH problem to get the temporally DHKey, which is also a significant element of generating session key. We analyze the probabilities of these two conditions in turn bellow.

Password Guessing Attack

During the *Authentication* procedure, the guessable secrets are the shared password pw and the *random nonce*. If the password is ℓ bits and random nonce is k bits. The probability of on-line guessing attack λ is bounded by q_{se} and N

$$\lambda \leq \frac{q_{se}}{2^\ell + k}$$

This attack can be prevented by increasing the waiting interval between the repeated trials.

On the other hand, the adversary will not get any advantage from off-line password guessing. Since the intercept message $\text{Enc}_{\text{TK}}(N)$ cannot be verified. And it also meet the *Rule 1* mentioned in previous chapter.

ECDH Attack

We assume the proposed protocol is insecure in the sense that the adversary \mathcal{A} can distinguish the session key given by the Test query. The algorithm \mathcal{B} can compute the Elliptic Curve Diffie-Hellman key by using the attacker \mathcal{A} as a subroutine.

\mathcal{B} plays a role of a simulator that response to all the queries from adversary \mathcal{A} and is given the challenge $\psi = (G, P=aG, Q=bG)$. Let ϵ be the probability that the output is identical to abG , within time t' .

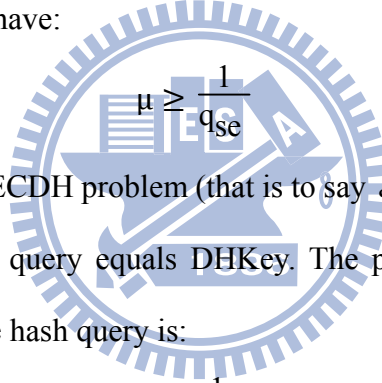
\mathcal{B} picks up a random nonce i from $[1, q_{se}]$, and sets a counter cnt , initially set to 0. \mathcal{B} uses a record set H -table, and set to \emptyset as well, then starts running \mathcal{A} . The response messages to the oracle queries made by \mathcal{A} are explained below.

1. When \mathcal{A} makes Send_1 query, \mathcal{B} answers msg_out_1 based on the protocol, and increases the counter cnt by 1. If cnt is equal to i , \mathcal{B} answers using the challenge $P=aG$ from the challenge ψ . When \mathcal{A} makes Send_2 query, \mathcal{B} answers msg_out_2 . When \mathcal{A} makes Send_3 query, if the input is the flow corresponding to the challenge ψ , \mathcal{B} answers using the element bG from the challenge ψ . Otherwise, \mathcal{B} answers what the protocol says to.
2. When \mathcal{A} makes $\text{Reveal}(\Pi_A^i)$ query, \mathcal{B} checks whether the oracle Π_A^i has been accepted and is fresh. If so, \mathcal{B} answers using the session key SK.
3. When \mathcal{A} makes $\text{Corrupt}(A)$ or $\text{Execute}(\Pi_A^i, \Pi_B^j)$ query, \mathcal{B} answers in a straightforward way.
4. When \mathcal{A} makes $\text{Hash}(m)$ query, \mathcal{B} will find out whether m has been queried before. If so, it will then send back the previous result; otherwise, it will generate a random value r and return to \mathcal{A} , and add (m, r) into the H -table.
5. When \mathcal{A} makes $\text{Encrypt}(k, m)$ query, \mathcal{B} will find out whether m has been queried before. If so, it will then send back the previous result; otherwise, it will

generate a random value r and return to \mathcal{A} , and add (k,m) into the E -table.

6. When \mathcal{A} makes Decrypt (k, c) query, \mathcal{B} will find out whether c has been queried before. If so, it will then send back the previous result; otherwise, it will generate a random value m and return to \mathcal{A} , and add (k,m) into the E -table.
7. When \mathcal{A} makes Test(Π_A^i) query, \mathcal{B} answers in a straightforward way. However, if the session key has to be constructed from the challenge ψ , \mathcal{B} answers with a random string with length $|\text{SK}|$.

The simulation is perfectly indistinguishable from the execution of the real protocol except for one execution in which the challenge ψ involved. The probability α that \mathcal{B} correctly guess the Session Key \mathcal{A} will use Test(Π_A^i) is the probability of \mathcal{B} correctly guess the value i . Then we have:



$$\mu \geq \frac{1}{q_{se}}$$

Suppose \mathcal{A} has broke the ECDH problem (that is to say \mathcal{A} learns the DHKey). There must be at least one Hash query equals DHKey. The probability δ of \mathcal{B} correctly chooses among the possible hash query is:

$$\delta \geq \frac{1}{q_h}$$

From the above description, we found that the probability $\text{Succ}_{\mathbb{G}}^{\text{ECDH}}(\mathcal{B})$ of the \mathcal{B} successfully outputs abG from the challenge ψ is the probability ε that \mathcal{A} breaks the AKE security multiplied by the probability μ that \mathcal{B} correctly guesses the moment at witch \mathcal{A} breaks the AKE security multiplied by the probability δ that \mathcal{B} correctly chooses among all the possible hash queries.

$$\text{Succ}_{\mathbb{G}}^{\text{ECDH}}(\mathcal{B}) = \varepsilon \times \mu \times \delta \geq \varepsilon \times \frac{1}{q_{se}} \times \frac{1}{q_h}$$

We also need to show that if two participants are partnered, they both accept and the password is confirmed to be correct, then the link key $\text{SK}(\Pi_A^i), \text{SK}(\Pi_B^i)$ established in the final phase are identical with the negligible probability. In the case that A and B

are partnered, this implies that session id $SID(\Pi_A^i)=SID(\Pi_B^i)$, and they both received the public key of each other. Note that the accept state only appears when E_a and E_b in Send8 and Send9 query pass the checks. The correctness holds only when two participants possess different random nonce and yet they both accept. We found that the probability of using different random nonce but output the same session key is negligible since the session key generation function is a pseudorandom function.

$$\eta < \text{negl}(n)$$

And thus the concrete security of the protocol is:

$$\text{Adv}_P^{\text{AKE}}(t, q_{se}, q_h) = \lambda + \varepsilon \leq \frac{q_{se}}{2^{\ell + k}} + \frac{q_{se}}{2^{\ell + k + j}} + q_{se} q_h \text{Succ}_{\mathbb{G}}^{\text{ECDH}}(t')$$

Since the advantage of the attacker $\text{Adv}_P^{\text{AKE}}$ is negligible, we successfully prove our protocol that meets requirements.



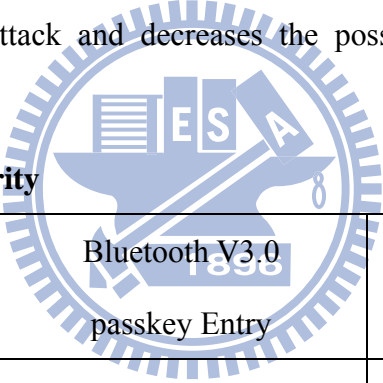
Chapter 6

Comparison

In this section, we briefly compare the security concern and performance of our proposed protocol with the current Bluetooth standard. We consider the following factor: whether defeats off-line / on-line guessing attack, man-in-the-middle attack, replay attack, etc. We also compare the number of hash, HMAC, encrypt, decrypt and message flows of both protocols.

Comparing with Bluetooth v3.0 passkey entry, our scheme not only keeps the original security features that can defeat man-in-the-middle and replay attack, but also defeats off-line guessing attack and decreases the possibility of on-line guessing attack.

On the aspect of security



	Bluetooth V3.0 passkey Entry	Our scheme
Off-line-guessing	N	Y
On-line-guessing	N	Y ^{*1}
Man-in-the-middle	Y	Y
Replay attack	Y	Y

*1 :Under the original scheme: increase the waiting interval between each trial

Table 6. 1 Comparison of security issue

On the aspect of performance, the message flows of Bluetooth v3.0 depends on the size of the password in bits. For example, there are four messages needed to verify each bit, and thus totally requires $4*k$ flows, where k is the password size. Our scheme only requires 4 messages and the computation cost is not much than the original

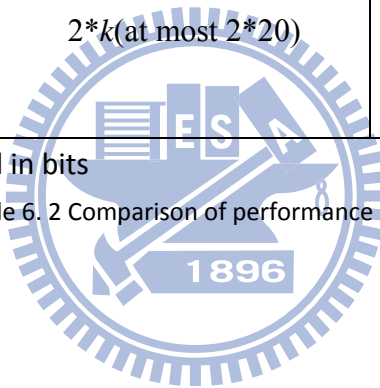
protocol. We use 1 encrypt, 1 decrypt and 1 hash operation only, while the v3.0 passkey entry requires $2*k$ hash operations. That is to say, if the password is 20 bits, it needs 40-times hash computation. Obviously, our scheme is much secure with better performance.

On the aspect of performance

	Bluetooth V3.0 passkey Entry	Our scheme
Number of flows	$4*k^2$ (at most $4*20$)	4
Number of encrypt and decrypt of each participant	0	2
Number of hash of each participant	$2*k$ (at most $2*20$)	1

*2 : k is the size of password in bits

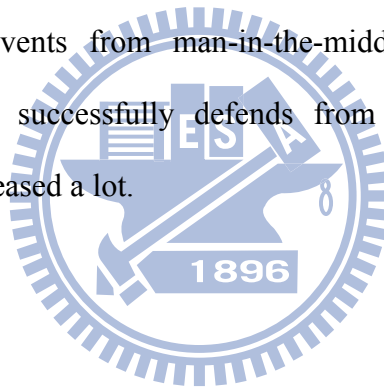
Table 6. 2 Comparison of performance issue



Chapter7

Conclusion

In this paper, we first introduce the weaknesses of the PIN-based authentication model of Secure Simple Pairing in the current Bluetooth Standard, namely, on-line / off-line guessing attack and the inefficiency of messages required. Then we demonstrate the reason why the authentication protocol cannot completely protect from off-line guessing attack. And we propose our scheme, which can meet higher security level and works more efficiently. We also provide heuristic and formal security analysis and compare with the original version in the standard. In our analysis, our protocol keeps forward secrecy, and prevents from man-in-the-middle attack as conventional. Additionally, the protocol successfully defends from guessing attack while the required messages are decreased a lot.



Chapter 8

Reference

- [1] Bluetooth. SIG, “Core specification v2.1+EDR,” Bluetooth SIG, Tech.Rep., 2007.
- [2] Bluetooth. SIG, “Core specification v3.0+EDR,” Bluetooth SIG, Tech.Rep., 2009.
- [3] Jakobsson, M., Wetzel, S, “Security weaknesses in Bluetooth”, Naccache, D, CT-RSA 2001. LNCS, vol. 2020, pp. 176–191. Springer, Heidelberg ,2001.
- [4] Shaked, Y., Wool, A, “Cracking the Bluetooth PIN,” In: MobiSys. The Third International Conference on Mobile Systems, Applications, and Services, pages 39–50 , 2005.
- [5] Wong, F.L., Stajano, F., Clulow, J, “Repairing the Bluetooth pairing protocol,” Proc. of the International Conference on Security Protocols. LNCS. Springer, Heidelberg, 2005.
- [6] J. Suomalainen, J. Valkonen, N. Asokan, “Security Associations in Personal Networks – A Comparative Analysis,” Nokia Research Center Technical Report NRC-TR-2007-004, 2007.
- [7] Karen Scarfone of NIST and by John Padgette of Booz Allen Hamilton, ”Guide to Bluetooth Security: Recommendations of the National Institute of Standards and Technology,” 2008.
- [8] S. Bellovin and M. Merritt. Encrypted Key Exchange: Password-Based Protocols Secure against Dictionary Attacks. Proc. of the Symposium on Security and Privacy, pages 72-84. IEEE, 1992.
- [9] Chang, T.Y., Yang, W.P., Hwang, M.S, “Simple Authenticated Key Agreement

- and Protected Password Change Protocol,” *Computers & Mathematics with Applications*. Vol. 49. No. 5-6. Pages 703-714, 2005.
- [10] E. J. Yoon and K. Y. Yoo, “A New Simple Authenticated Key Agreement and Protected Password Change Protocol,” In *Proceedings of EUC Workshops 2005*. LNCS 3823, pages 955-964, 2005.
- [11] Yeh, H.T., Sun, H.M, “Simple Authenticated Key Agreement Protocol resistant to Password Guessing Attacks,” *ACM SIGOPS Operation Systems Review*. Vol. 36. No. 4. Pages 14-22, 2002.
- [12] M. Bellare, D. Pointcheval, and P. Rogaway, “Authenticated Key Exchange Secure Against Dictionary Attacks,” In *Eurocrypt '00*, LNCS 1807, pages 139–155. Springer-Verlag, Berlin, 2000.
- [13] M. Bellare and P. Rogaway, “The AuthA Protocol for Password-Based Authenticated Key Exchange,” *Contributions to IEEE P1363*. March 2000.
- [14] E. Bresson, O. Chevassut, and D. Pointcheval, “Security Proofs for Efficient Password-Based Key Exchange,” In *Proc. of the 10th CCS*. ACM Press, New York, 2003.
- [15] V. Boyko, P. MacKenzie, and S. Patel, “Provably Secure Password-Authenticated Key Exchange Using Diffie-hellman,” In B. Preneel, editor, *Proceedings EUROCRYPT2000*, pages 156–171, 2000.
- [16] C. Gehrman, C. Mitchell and K. Nyberg, “Manual Authentication for Wireless Devices,” *RSA Cryptobytes*, vol.7, pages 29-37, 2004.
- [17] S. Vaudenay, “Secure Communications over Insecure Channels Based on Short Authenticated Strings,” In *CRYPTO 2005*, Springer-Verlag (LNCS 3621), pages 309-326, 2005.
- [18] Emmanuel Bresson , Olivier Chevassut , David Pointcheval, “Security proofs for an efficient password-based key exchange,” *Proceedings of the 10th ACM*

- conference on Computer and communications security, 2003
- [19] Oded Goldreich , Yehuda Lindell, “Session-Key Generation Using Human Passwords Only,” Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology, pages 408-432, 2001.
- [20] J. Katz, R. Ostrovsky, and M. Yung, “Efficient password-authenticated key exchange using human-memorable passwords,” In B. Pfitzmann, editor, EUROCRYPT 2001, volume 2045 of LNCS, pages 475–494. Springer-Verlag, 2001.
- [21] K. Kobara and H. Imai, “Pretty-simple password-authenticated key-exchange under standard assumptions,” IEICE Transactions, E85-A(10): pages 2229–2237, 2002.
- [22] L. Gong, M. Lomas, R. Needham, J. Saltzer, “Protecting poorly chosen secrets from guessing attacks,” IEEE J. Sel. Areas Comm. 11 (5) pages 648–656, 1993.
- [23] C. Gehrman, C. Mitchell and K. Nyberg, “Manual Authentication for Wireless Devices,” RSA Cryptobytes, vol. 7, pages 29-37, 2004.
- [24] J.H. Hoepman, “The Ephemeral Pairing Problem,” In Financial Cryptography, Springer-Verlag (LNCS 3110), pages 212-226, 2004.
- [25] J.H. Hoepman, “Ephemeral Pairing on Anonymous Networks,” In the 2nd Conference on Security in Pervasive Computing, Springer-Verlag (LNCS 3450), pages 101-116, 2005.
- [26] Li Gong, Roger Needham, and Raphael Yahalom, “Reasoning about belief in cryptographic protocols,” In Proceedings of the IEEE Symposium on Research in Security and Privacy, pages 234-248,1990.