Authenticated Key Exchange Protocols with Unbalanced Computational Requirements

Thesis submitted in accordance with the requirements of the University of Liverpool for the degree of Doctor in Philosophy by

Jie Zhang

August 2018

Abstract

Security is a significant problem for communications in many scenarios in Internet of Things (IoT), such as military applications, electronic payment, wireless reprogramming of smart devices and so on. To protect communications, a secret key shared by the communicating parties is often required. Authenticated key exchange (AKE) is one of the most widely used methods to provide two or more parties communicating over an open network with a shared secret key. It has been studied for many years. A large number of protocols are available by now. The majority of existing AKE protocols require the two communicating parties execute equivalent computational tasks. However, many communications take place between two devices with significantly different computational capabilities, such as a cloud center and a mobile terminal, a gateway and a sensor node, and so on. Most available AKE protocols do not perfectly match these scenarios.

To further address the security problem in communications between parties with fairly unbalanced computational capabilities, this thesis studies AKE protocols with unbalanced computational requirements on the communicating parties. We firstly propose a method to unbalance computations in the Elliptic Curve Diffie-Hellman (ECDH) key exchange scheme. The resulting scheme is named as UECDH scheme. The method transfers one scalar multiplication from the computationally limited party to its more powerful communicating partner. It significantly reduces the computational burden on the limited party since scalar multiplication is the most time-consuming operation in the ECDH scheme.

When applying the UECDH scheme to design AKE protocols, the biggest challenge is how to achieve authentication. Without authentication, two attacks (the man-in-the-middle attack and the impersonation attack) can be launched to the protocols. To achieve authentication, we introduce different measures that are suitable for a variety of use cases. Based on the authentication measures, we propose four suites of UECDH-based AKE protocols. The security of the protocols is discussed in detail. We also implement prototypes of these protocols and similar protocols in international standards including IEEE 802.15.6, Transport Layer Security (TLS) 1.3 and Bluetooth 5.0. Experiments are carried out to evaluate the performance. The results show that in the same experimental platform, the proposed protocols are more friendly to the party with limited computational capability, and have better performance than similar protocols in these international standards.

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisor Dr. Xin Huang for his support and guidance throughout my time of PhD research. He has made many contributions to this work. He has given me a lot of valuable guidance and suggestions for conducting the PhD research and completing the PhD thesis.

Secondly, I would like to thank to my second supervisor Professor Alan Marshall and my third supervisor Dr. Paul Craig for their valuable suggestion on my PhD research. I would also like to thank Dr. Dawei Liu, Dr. Wei Wang and Nian Xue for their help.

This work is supported by Department of Computer Science, University of Liverpool and Department of Computer Science and Software Engineering, Xi'an Jiaotong-Liverpool University. Special thanks to all the colleagues from these institutes.

Finally, I would like to thank my family and friends for their love and friendship. I would especially like to thank my husband Zilong Wei for his support and love.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Nowadays, large numbers of devices are equipped with communicating capability and are connected into the Internet of Thighs (IoT). Communications take place every minute between various devices in the IoT; and many of them occur between two devices with fairly different (or "unbalanced") computational capabilities, such as a smart phone and a cloud server, a sensor node and a base station, and so on. For these communications, security is often required since they can carry sensitive information, and authentication between the communicating devices are generally required. This thesis explores security solutions for communications in the IoT, and in particular, for communications between two devices with unbalanced computational capabilities.

1.1 Motivation

The motivation lies in two aspects: (1) the importance of security and (2) available solutions and their unsuitability for communications between devices with unbalanced computational capabilities.

1.1.1 Importance of Security

Information security is a vitally important problem in many applications in the IoT [29, 64, 72]. It generally involves two basic objectives: message authenticity (or message integrity) and confidentiality [55]. Details about these objectives and their importance in various IoT applications are introduced as follows.

- Authenticity (or integrity). Message authenticity (or integrity) guarantees a party is able to identify whether a message it receives was sent by a party claiming to have sent it, and was not modified in transit. It is the basic objective in many applications. For example, in a wireless reprogramming scenario, when an IoT device receives an update service, it should be able to make sure two things: (1) whether the service was sent by the service provider and (2) whether the update files were modified in transit. Lacking authenticity, attackers can implant malware into the device. For instance, Ronen *et. al* [86] break Philips IoT platform [84] through wireless firmware update; and Ling *et. al* illustrate firmware attacks to IoT through installing a malicious firmware on the smart plug [65].

- Confidentiality. Confidentiality guarantees sensitive information is only known by those authorized to know it. Many IoT applications, such as e-healthcare and payment, involve sensitive information that requires confidentiality. Firstly, people would not use these applications if confidentiality of privacy information is not guaranteed. Secondly, leakage of privacy information, such as the password of a user's bank account, will lead to serious losses. Thirdly, protecting the privacy of users is required by the law [98], and accords with ethic.

1.1.2 Available Solutions

Authenticated key establishment (AKE) is the underlying approach to security problems. It establishes shared keys for two or more parties to protect the communications between them. It has been studied for many years; and many schemes have been proposed. Typical types of available AKE scheme are introduced as follows.

Schemes based on symmetric cryptography

AKE schemes based on symmetric cryptography require the communicating parties have a pre-shared secret with each other (denoted as server-less schemes), or have shared secrets with a trusted center (denoted as server-based schemes). The international standard ISO/IEC 11770 Part 2 [52] specifies six server-less schemes and seven server-based schemes. A server-less scheme [89, 106, 105] generates a shared session key from the pre-shared secret between the parties. A server-based scheme [77] relies on a trusted center to distribute a shared session key for the parties; and a pre-shared secret is used to protect communication between the party and the trusted center. In both the two types of scheme, a session key is used to protect subsequent communications in the session.

The advantage of these schemes is that they are lightweight and affordable for computational limited devices. The limitations include: (1) in some cases it is inconvenient for devices to deploy and update pre-shared secrets; (2) the pre-shared secret should be stored in secure memory which is expensive.

Schemes based on asymmetric cryptography

AKE schemes based on asymmetric cryptography [88] require the parties acquire the public key of each other. The public keys are used to securely transport secret values [14, 76, 66]. The secret values can be directly used as a session key. Alternatively, the two parties can agree a session key based on the secret value.

The advantage of these schemes is that they do not require any pre-shared secret between the parties. The limitations include: (1) they often rely on public key infrastructure (PKI) [50] where there is a certificate authority (CA) issuing public key certificates for the parties; and maintaining the public key certificates is complicated and expensive; and (2) asymmetric cryptographic algorithms have high computational requirements and may overburden the limited devices.

Table 1.1: Key sizes for equivalent security levels (in bits).

Symmetric algorithm	ECC algorithm	DH/RSA
80	163	1024
128	283	3072
192	409	7680
256	571	15,360

Schemes based on (elliptic curve) Diffie-Hellman key exchange

Diifie-Hellman (DH) key exchange [33] is the basis for a vast range of AKE protocols. Its elliptic curve version, i.e., elliptic curve Diffie-Hellman (ECDH) key exchange, has become one of the most popular measures to design AKE protocols in IoT [75, 4, 108, 63, 110, 21]. Many international standards and specifications for security or communication, such the TLS standard, theIEEE standard 802.15.6 and the Bluetooth specification 5.0, include AKE schemes based on ECDH. In an ECDH-based AKE scheme, the parties generate secret values, compute public values, exchange the public values, and compute the shared secret respectively.

The advantage of ECDH-based schemes over asymmetric cryptography-based schemes is efficiency. Elliptic curve cryptographic (ECC) algorithms provide high security with relatively shorter keys; and they do not require any pre-shared secret. K. Lauter [61] compares key length for the same security level provided by ECC, symmetric cryptography algorithms and other asymmetric cryptographic algorithms such as DH key exchange and RSA (Table 1.1). As a result, ECDH-based AKE schemes often surpass other AKE schemes in IoT.

Summary of available schemes

As we presented above, ECDH-based AKE schemes are more suitable for IoT since they do not require pre-shared secret between the parties; and they are more efficient than other asymmetric cryptography based schemes. However, available ECDH-based AKE schemes require the communicating parties execute equivalent computational tasks. Therefore, they are not perfectly suitable for communications between two devices with unbalanced computational capabilities.

1.2 Solution

This thesis aims to design more suitable AKE protocols for communications between two devices with unbalanced computational capabilities. In particular, we aim to unbalance computations in the ECDH key exchange scheme and design AKE protocols with unbalanced computational requirements. Our solution is to transfer scalar multiplications form one parity to its communicating partner. The resulting scheme is denoted as the UECDH key exchange scheme. Since scalar multiplication is the most time-consuming operation in ECDH, the UECDH key exchange scheme is anticipated to significantly reduce the computational burden on the limited party.

When applying the UECDH key exchange scheme to design AKE protocols, the biggest challenge is how to establish authentication. Without authentication, two types of attack will occur. The first one is the man-in-the-middle attack. It is inherited from DH and ECDH key exchange schemes which do not contain any authentication of the exchanged messages. The second one is the impersonation attack. It is caused partly by transferring computations and partly by lacking authentication.

1.3 Contributions

To achieve the aforementioned aim and overcome the challenge, we utilize four different authentication measures and design four sets of UECDH-based AKE protocols. The protocols are suitable for a variety of use cases. The main contributions of this thesis are summarized as follows.

- Proposal of the method to unbalance computations in ECDH key exchange. The method transfers one scalar multiplication from one party to another. It significantly reduces the computational burden on the limited party since scalar multiplication is the most time-consuming operation in ECDH.

- Design of four sets of UECDH-based AKE protocols. The protocols have two advantages over similar protocols in international standards or specifications including IEEE 802.15.6, TLS and Bluetooth 5.0. First, they are more suitable for communications between two devices with fairly different computational capabilities. Second, they achieve a better overall performance by letting the more powerful party to undertake computations on behalf of the limited one.

1.4 Publications

This thesis is partly based on the following publications. The contributions of the author are also listed.

1. Jie Zhang, Nian Xue, Xin Huang. A Secure System for Pervasive Social Network-based Healthcare. IEEE Access, Vol. 4, Pages 9239-9250, 2016.

   Contributions of the author: (1) design of the improved IEEE 802.15.6 display authenticated association protocol; (2) design of the protocol for blockchain consensus mechanism; (3) design of the healthcare blockchain; (4) security analysis for protocols; and (5) performance evaluation.

2. Jie Zhang, Xin Huang, Paul Craig, Alan Marshall, Dawei Liu. An Improved Protocol for the Password Authenticated Association of IEEE 802.15. 6 Standard That Alleviates Computational Burden on the Node. Symmetry, Vol. 8, No. 11, Pages 131, 2016.

   Contributions of the author: (1) illustration of attacks to the IEEE 802.15.6 password authenticated association protocol; (2) design of the improved security protocol; (3) security analysis for the improved protocol; and (4) performance evaluation.

3. Jiaren Cai, Xin Huang, Jie Zhang, Jiawei Zhao, Yaxi Lei, Dawei Liu, Xiaofeng Ma. A Handshake Protocol with Unbalanced Cost for Wireless Updating. IEEE Access, Vol. 6, No. 1, Pages 18570-18581, 2018.

Contributions of the author: (1) design of the handshake protocol; and (2) security analysis for the protocol.

4. Nian Xue, Xin Huang, Jie Zhang. S^2Net: A Security Framework for Software Defined Intelligent Building Networks. Trustcom/BigDataSE/ISPA, IEEE, 2016.

   Contributions of the author: (1) design of the authenticated OpenFlow association protocol; (2) desing of the secure OpenFlow message issuing protocol; and (3) security analysis for the protocols.

5. Nian Xue, Lulu Liang, Jie Zhang, Xin Huang. POSTER: A Framework for IoT Reprogramming. Proceedings of International Conference on Security and Privacy in Communication Systems, Springer, Pages 751-754, 2016.

   Contributions of the author: (1) design of OpenFunction authenticated handshake protocol; and (2) design of OpenFunction messaging protocol.

6. Xin Huang, Dawei Liu, Jie Zhang. An Improved IEEE 802.15. 6 Password Authenticated Association Protocol. Communications in China (ICCC), 2015 IEEE/CIC International Conference on, 2015.

   Contributions of the author: security analysis.

7. Ruiyang Xu, Xin Huang, Jie Zhang, Yulin Lu, Ge Wu, Zheng Yan. Software Defined Intelligent Building. International Journal of Information Security and Privacy, Vol. 9, No. 3, pages 84-99, 2015.

   Contributions of the author: (1) design of lightweight security mechanisms for OpenFlow; and (2) security analysis.

8. Jie Zhang, Xin Huang, Andi Xu, Mi Li, Sihan Wu. An Enable Bitcoin Transaction Automatic Ticketing Machine in Public Transport, Patent Number: ZL201620369250.7, 2016.

   Contributions of the author: (1) protocol design; and (2) system design.

9. Xin Huang, Kai Zheng, Jie Zhang, Nian Xue, Qiankun Sheng. An IoT-based Photovoltaic System, Patent Number: ZL201621140724.7, 2016.

   Contributions of the author: (1) protocol design; and (2) system design.

1.5 Thesis Outline

The rest of this thesis is organized as follows. Chapter 2 reviews the underlying cryptography knowledge and related work. Chapter 3 introduces the method of unbalancing computations in the ECDH key exchange scheme; and discusses security issues. The four sets of UECDH-based AKE protocols are presented in Chapter 4 to 7 successively. Chapter 4 presents the password UECDH-based AKE protocols; the security of the protocols is studied; and the performance is compared with a similar protocol in IEEE 802.15.6. In Chapter 5, the public key authenticated UECDH-based AKE protocols are presented; their security is studied; and a similar protocol in TLS is chosen as the benchmark to compare performances. In Chapter 6, the high bandwidth Out-of-Band (OOB) UECDH-based AKE protocols are presented; their security is studied; and the performance is compared with similar protocol in Bluetooth 5.0. In Chapter 7, the low bandwidth OOB UECDH-based AKE protocols are presented; their security are analyzed; and the performance is compared with similar protocols in IEEE 802.15.6 and Bluetooth 5.0. Finally, Chapter 8 summarizes the main contributions; and proposes future works.

Chapter 2

Preliminaries

This chapter reviews some underlying cryptography knowledge and related work. We firstly introduce the definition of elliptic curve and the concept of ECC. Secondly, we introduce cryptographic primitives that will be used in protocol design. Thirdly, we present definition, architecture, and communication, attack and security models of a general AKE protocol. Finally, we review AKE protocols in some international standards.

2.1 Elliptic Curves

2.1.1 Definition

Definition 2.1 (Elliptic Curves)**.** *An elliptic curve E over a field GF is defined by the Weierstrass equation as follows*

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{2.1}$$

where $a_1, a_2, a_3, a_4, a_5 \in GF$ and every point on E is non-singular (or "smooth"), that is, there is no point at which the curve has more than one distinct tangent lines.

The elliptic curve E is composed of all solutions (x, y) of Equation 2.1 and ∞ which is a point at infinity [46].

2.1.2 Operations

- Point Addition $+$. Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be two distinct points on an elliptic curve E. The point addition of P and Q is denoted by $P + Q$. The sum is also a point on E.

- Scalar Multiplication $\times$. Let t be an integer and P be a point on an elliptic curve E. The scalar multiplication between t and P is denoted by $t \times P$. The result is also a point on E. It means

$$t \times P = \underbrace{P + P + \cdots + P}_{t}.$$

When t is a large integer, computing a scalar multiplication is much more time-consuming than computing a point addition [3, 5, 103].

2.1.3 Elliptic Curves For Cryptography

One type of elliptic curve E that is suitable for cryptography is defined as follows [68]:

$$y^2 = x^3 + ax + b \mod p, \text{ with } a, b \in GF(p) \text{ and } 4a^3 + 27b^2 \neq 0 \tag{2.2}$$

where $GF(p)$ is a prime finite field. Its order p (denoting the number of elements in the finite field) is an odd prime. To determine an elliptic curve, the following parameters should be given:

- p. The order of $GF(p)$.

- r. The order of E.

- a and b. The coefficients.

- $G = (G_x, G_y)$. The base point of E.

2.1.4 Standards

The National Institute of Standards and Technology (NIST) - Federal Information Processing Standards (FIPS) [79, 80] selects the following equation to define elliptic curves used for cryptography.

$$y^2 = x^3 - 3x + b \mod p \tag{2.3}$$

In Equation 2.3, the selection $a = -3$ for the coefficient of x is made for reasons of efficiency. In the standard, five example elliptic curves are specified based on Equation 2.3: Curve P-192, Curve P-224, Curve P-256, Curve P-384 and Curve P-521. The modular p of each curve is listed as follows.

- Curve P-192. $p = 2^{192} - 2^{64} - 1$.
- Curve P-224. $p = 2^{224} - 2^{96} + 1$.
- Curve P-256. $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$.
- Curve P-384. $p = 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$
- Curve P-521. $p = 2^{521} - 1$

For more information about these curves, please refer to [79, 80].

2.1.5 Difficult Problems

The elliptic curve discrete logarithm problem (ECDLP) and elliptic curve Diffie-Hellman problem (ECDHP) [46] are the underlying difficult problems for the security of ECC and ECDH-based schemes.

Definition 2.2 (Elliptic Curve Discrete Logarithm Problem (ECDLP))**.** *Let E be an elliptic curve defined over a finite field $GF(p)$, P be a point on E of order n, and Q be another point on E such that $Q = xP$ for some unknown $x \in [0, n-1]$. Given P and Q, the elliptic curve discrete logarithm problem (ECDLP) is to find x. We use the notation $ECDL_P(Q) = x$.*

Definition 2.3 (Elliptic Curve Diffie-Hellman Problem (ECDHP))**.** *Let E be an elliptic curve defined over a finite field $GF(p)$, P be a point on E of order n, and A and B be points on E such that $A = aP$ and $B = bP$ for some unknown $a, b \in [0, n-1]$. Given P, A and B, the elliptic curve Diffie-Hellman problem (ECDHP) is to find the point $C = abP$. We use the notation $ECDH_P(A, B) = C$.*

2.1.6 Advantages

A number of studies show that for the same level of security, the elliptic curve based systems are implemented with much smaller parameters [61]. This leads to significant performance advantages. As a result, elliptic curve based systems are widely adopted in recent years. For example, all of the latest versions of the TLS standard, IEEE 802.15.6 standard, Bluetooth standard and IEEE 802.15.4 (Zigbee) standard define elliptic curve based protocols.

2.2 Cryptographic Primitives

2.2.1 Message Authentication Code

A message authentication code (MAC) [58, 8] is defined by a pair of algorithms (MAC, VER). The MAC takes a message m and a secret key k as inputs, and outputs a MAC mac. The VER takes m, k and mac as inputs and outputs 1 (valid) or 0 (invalid). The algorithms are denoted as follows.

$$\begin{aligned} \mathrm{MAC}(k, m) &= mac \\ \mathrm{VER}(k, m, mac) &= \{0, 1\} \end{aligned} \tag{2.4}$$

2.2.2 Digital Signature

A digital signature scheme [54] is defined by a pair of algorithms (SIGN, VERY). The SIGN takes a message m and a private key sk as inputs, and outputs a signature σ. The VERY takes m, σ and the corresponding public key pk as inputs, and outputs 1 (valid) or

0 (invalid). The algorithms are denoted as follows.

$$\begin{aligned} \mathrm{SIGN}(sk, m) &= \sigma \\ \mathrm{VERY}(pk, m, \sigma) &= \{0, 1\} \end{aligned} \tag{2.5}$$

2.2.3 Elliptic Curve Diffie-Hellman Key Exchange

The elliptic curve Diffie-Hellman (ECDH) [57] key exchange is a scheme that allows two parties (each has a pair of elliptic curve public and private keys) to agree a shared secret over an insecure channel. Denote the parties by A and B. A and B share an elliptic curve E with the base point G. The ECDH key exchange scheme executes as follows.

1. A generates a random value $SK_A \in Z_q^*$, and computes $PK_A = SK_A \times G$. Then A sends PK_A to B.

$$A \rightarrow B : PK_A.$$

2. B generates a random value $SK_B \in Z_q^*$, and computes $PK_B = SK_B \times G$. Then B sends PK_B to A.

$$B \rightarrow A : PK_B.$$

3. A computes the shared secret through the following equation:

$$K = SK_A \times PK_B = SK_A \times SK_B \times G. \tag{2.6}$$

B computes the shared secret through the following equation:

$$K = SK_B \times PK_A = SK_A \times SK_B \times G. \tag{2.7}$$

2.3 Authenticated Key Exchange

2.3.1 Definitions

Definition 2.4 (Authenticated key establishment)**.** *Authenticated key establishment is a cryptographic mechanism that provides two or more parties communicating over an open network with a shared secret key.*

The shared secret key is used subsequently by cryptographic primitives as we described in Section 2.2 to achieve some security goals such as authentication, confidentiality or integrity. There are two types of authenticated key establishment protocol: authenticated key transport protocols and authenticated key agreement (or exchange) protocols [23]. They are introduced as follows.

- Key transport protocols [15, 13]. In a key transport protocol, the shared secret key is created by one party and securely transmitted to the second party.

- Key agreement (or exchange) protocols [53, 93, 69, 94, 90]. In a key exchange protocol, both parties contribute information which is used to derive the shared secret key.

This thesis is concerned with authenticated key exchange protocols with two parties. We also consider the utilization of additional channels with certain security features between the parties. This has been adopted by a number of AKE protocols such as the protocols in the IEEE standard 802.15.6, the Bluetooth specification 5.0 and the Zigbee specification 3.0. The authenticated key exchange in this thesis is formally defined as follows.

Definition 2.5 (Authenticate key exchange in this thesis)**.** *Authenticated key exchange (AKE) is a cryptographic mechanism that provides two parties communicating mainly over an open network with an authenticated shared secret key.*

2.3.2 Protocol Architecture

An AKE protocol is composed of the following three procedures.

- Initialization. This procedure initializes public and private parameters on both participants. The parameters are long-term values. This procedure does not have to be executed in each session of the protocol.
- Key exchange. This procedure generates ephemeral values, exchanges messages, and generates a shared secret for a pair of participants.
- Session keys derivation. This procedure derives the session keys from the shared secret on each participant. Different keys will be generated for different cryptographic primitives.

2.3.3 Communication Model

The communication model of an AKE protocol (Figure 2.1) is defined by a pair of participants and the communication channels between them. The participant who initiates the protocol is denoted by the initiator. The other participant is denoted by the responder. The thesis specifies the following two types of channels between the initiator and the responder.

- Normal channels. Normal channels are Dolev-Yao channels [35]. All messages transmitted via these channels can be overheard, deleted or modified by an attacker. Examples of normal channels include the Internet, Bluetooth [45, 48, 28], Wi-Fi [1, 12, 101] and Zigbee [7, 36, 26] networks. This thesis denotes the normal channels by $\rightarrow$.
- Out-of-band (OOB) channels. The OOB channels refer to empirical (or authentication) channels [78, 51, 6, 31, 20, 102]. All messages transmitted in these channels are authentic and cannot be faked or modified. Examples of OOB channels include human-controlled channels (such as scanning quick response (QR) code [56, 81], comparing short strings on displays or pressing buttons), human-controlled visible light

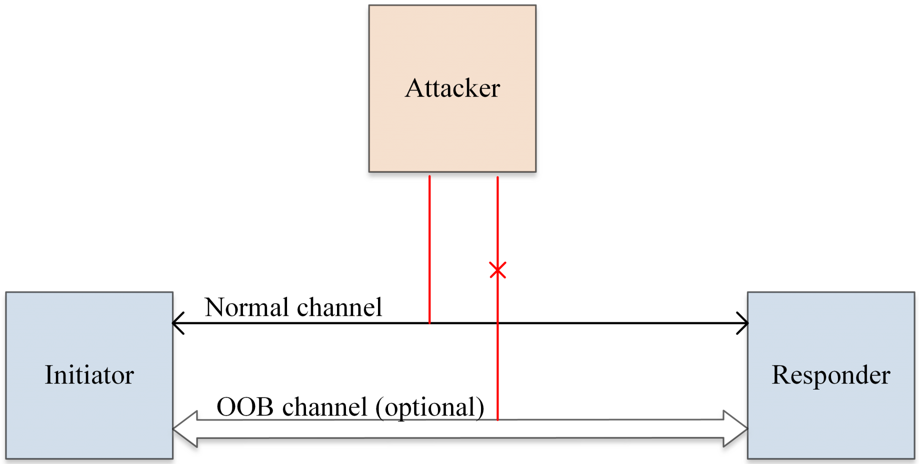

Figure 2.1: Communication model of AKE protocols.

channels and human body channels. Some of them are high bandwidth OOB channels that can transmit long strings such as a public key; and some OOB channels are low bandwidth channels that can only transmit short strings. This thesis denotes OOB channels (both high bandwidth and low bandwidth ones) by $\Rightarrow$.

2.3.4 Attack Model

We at first present the basic assumptions. Then under the assumptions, we define the attack model that specifies what the attacker is able to do.

Basic assumptions

The attacker is unable to break the MAC and digital signature algorithms. That is, for a MAC computed as $mac = \mathrm{MAC}(k, m)$, the attacker finds it difficult to generate a mac' such that $\mathrm{VER}(k, m, mac') = 1$ without k; and for a signature computed as $\delta = \mathrm{SIGN}(sk, m)$, the attacker finds it difficult to generate a σ' such that $\mathrm{VERY}(pk, m, \sigma') = 1$ without sk. In addition, the attacker is unable to alter, insert, delay or delete messages transmitted via the OOB channels.

Attack model

- Basic ability. The attacker is able to control the communications over normal channels between the initiator and the responder. That is, the attacker can observe all messages sent, alter messages, insert new messages, delay messages or delete messages in the normal channels.
- Stronger ability 1. The attacker is able to obtain any previous session keys.
- Stronger ability 2. The attacker may compromise long-term secret keys of the initiator or the responder.

There are two well-known attacks that the attackers have some of the above abilities, that is, the man-in-the-middle attack [22] and the impersonation attack [109]. The two attacks are the most commonly encountered threats to AKE protocols.

- The man-in-the-middle attack (shown in Figure 2.2). In this attack, the attacker relays and modifies the communication between two parties who believe they are communicating with each other. To launch a man-in-the-middle attack, the attacker is able to intercept all messages transmitted between the two victimized parties and inject new ones.
- The impersonation attack (shown in Figure 2.3). In this attack, the attacker claims to be the first party and communicates with the second party. To launch an impersonation attack to an AKE protocol, the attacker have to compromise the authentication information (such as the long-term secret keys) of the first party. This is known as the key compromise impersonation attack.

2.3.5 Security Goals

Let A and B be two honest participants of an AKE protocol, that is, they execute the steps of the protocol correctly [16, 99]. Security goals are explained as follows.

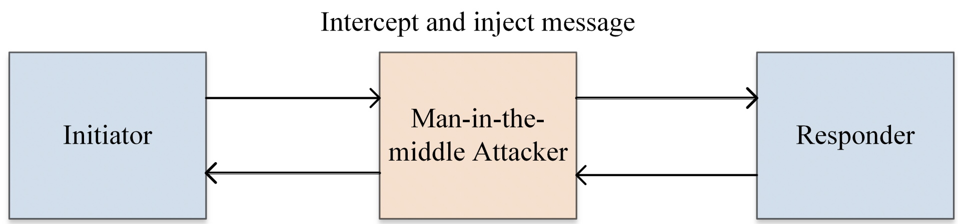

Figure 2.2: The man-in-the-middle attack.

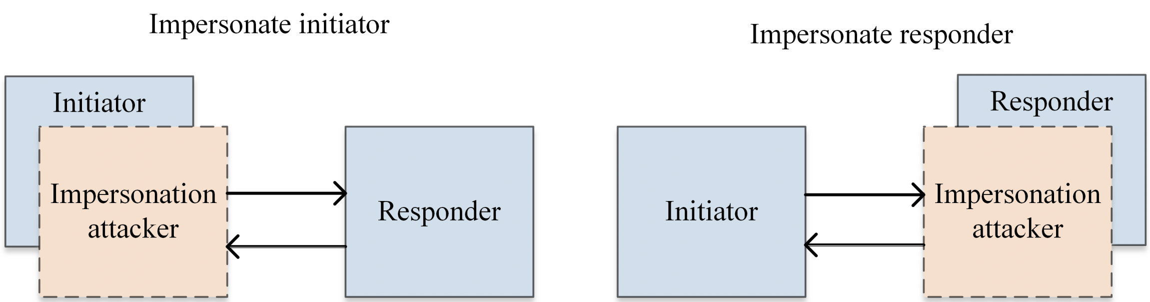

Figure 2.3: The impersonation attack

- Key authentication [10, 34, 41]. A completed run of an AKE protocol between A and B should produce a secret that is shared only by A and B other than any other party.

- Key confidentiality. A completed run of an AKE protocol between A and B should produce a secret that can not be computed by any other party aside from A and B.

- Key integrity. After a completed run of an AKE protocol between A and B, the secret computed by A should be equivalent with that computed by B.

- Key confirmation [67, 37, 70]. After a completed run of an AKE protocol between A and B, both A and B have receive evidence confirming that the other party knows the secret.

- Known-key security (key freshness) [43, 42]. Each run of an AKE protocol between A and B should produce a unique secret key (i.e., the session key). This attribute is also known as *key freshness*.

- Forward secrecy [2, 83, 32, 59, 18]. If long-term private keys of A and/or B are compromised, the secrecy of previous session keys is not affected.

In addition to the above goals, special security goals are also required by specific AKE protocols. Two special security goals involved in this thesis are introduced as follows.

- Resistance to dictionary attacks [73, 100, 11, 74]. This goal is required by an AKE protocol that utilizes passwords. It guarantees that an eavesdropper who can record the transcript of one or more sessions cannot eliminate a significant number of possible passwords.

- Resistance to combinatorial search attacks such as the birthday attack [40, 30, 9, 107]. This goal is required by an AKE protocol that uses short hash functions to generate authentication information. It guarantees that general or multiple-shot attacks give the attacker no advantage over guess.

2.3.6 Cost Model

Generally, the cost of an AKE protocol is evaluated through communicating and computational costs as follows.

- Communicating cost. The evaluation of communicating cost has two aspects:
  - Number of passes, i.e., the number of messages exchanged.
  - Communicating overhead, i.e., the overall number of bits transmitted.
- Computational Cost. The computational cost is evaluated as follows:
  - On-line computational cost on the initiator, i.e., the number of arithmetical operations required on the initiator.
  - On-line computational cost on the responder, i.e., the number of arithmetical operations required on the responder.
  - Overall computational overhead, i.e., the total number of arithmetical operations required.

In this thesis, we mainly focus on the computational cost, in particular, the on-line computational cost. It is studied via the following two methods:

- Theoretical evaluation. This method counts the number of time-consuming arithmetical operations involved in each run of the protocol.
- Experimental test. This method realizes and runs a prototype of the protocol to test computational time.

2.4 Authenticated Key Exchange In Standards

2.4.1 IEEE 802.15.6

IEEE 802.15.6 [97] is the international standard for wireless body area networks (WBANs) [62, 60, 24, 87, 92]. It includes a suite of ECDH-based authenticated association protocols (i.e., AKE protocols) that generate authenticated shared keys for a node and a hub. The protocols are briefly introduced as follows.

- Public key hidden association. The public key hidden association protocol is denoted as IEEE PK Hidden in this thesis. It requires the hub have the public key of the node in advance of running the protocol. The node's public key is kept secretly in the protocol to help to prevent third parties from launching impersonation attacks.
- Password authenticated association. The password authenticated association protocol is denoted as IEEE PW in this thesis. It requires the node and hub have a secretly shared password before the running the protocol. The password helps to keep third parties from launching impersonation attacks.
- Display authenticated association. The display authenticated association is denoted as IEEE Display in this thesis. It requires the node and hub each to have a display of a 5-digit decimal number before running the protocol. The display is a type of low bandwidth OOB channel that helps to keep attackers from launching man-in-the-middle attacks.

2.4.2 Bluetooth

Bluetooth wireless technology (or Bluetooth for short) is a short-range, robust and low cost communications system which aims to replace the cable(s) connecting portable and/or fixed electronic devices. The Bluetooth security model includes five distinct security features: pairing, bonding, device authentication, encryption and message integrity. The pairing, bonding and device authentication constitute an AKE protocol that establishes shared and authenticated keys for devices. The encryption and message integrity are related with secure communications protected by the shared keys. We summarize the ECDH-based AKE protocols in Bluetooth Specification version 5.0 (denoted as Bluetooth 5.0) [85] as follows.

- Numeric comparison association. The numeric comparison association is denoted as Bluetooth Display in this thesis. It requires both devices to be capable of displaying a six digit number and both to be capable of having the user enter "yes" or "no".

- Just works association. The just works association is similar with the numeric comparison association. It is suitable for scenarios where at least one of the devices does not have a display for a six digit number. It also uses numeric comparison scheme.

- Out-of-Band (OOB) association. The OOB association is denoted as Bluetooth OOB in this thesis. It requires the communicating devices can establish OOB channels that provide different security properties compared with the Bluetooth radio channel.

- Passkey entry association. The Passkey entry association is suitable for scenarios that one device has a keyboard but does not have a display; and the other device has a display for a six digit number. It is essentially a numeric comparison scheme. The user is shown a six digit number on the device with a display, and is then asked to enter the number on the other device.

2.4.3 Transport Layer Security (TLS)

The Transport Layer Security (TLS) [71] defines cryptographic protocols that provide communications security over a computer network. It involves a suit of handshake protocols that establish shared secret keys for a server and a client. After the handshake, the shared keys are used to protect the application layer traffic. The latest version of TLS specification, i.e., TLS 1.3, involves the ECDH-based handshake protocol. The protocol requires the two parties have authenticated public key of each other. This is often realized through a public key certificate. In this thesis, we denote this protocol as TLS PK Authenticated.

2.5 Chapter Summary

In this chapter, we presented preliminary cryptography knowledge and related work. In particular, we formally defined the general architecture, communication model, attack model, secure goals and cost model of an AKE protocol. We also summarized ECDH-based AKE protocols in international standards or specifications for communication or security. These protocols will be set as the benchmarks for protocols proposed in this thesis.

Chapter 3

Unbalancing ECDH Key Exchange

This chapter introduces the method to unbalance computations in the ECDH key exchange scheme. In particular, the scalar multiplications are transferred from one party to its communicating partner. We firstly present the scenario where communications take place between two devices with fairly different computational capabilities. Secondly, we illustrate how to transfer one scalar multiplication from the initiator to the responder, and how to transfer oppositely. The resulting schemes are named UECDH key exchange schemes. Thirdly, we discuss two severe attacks to the UECDH schemes. The first one is the man-in-the-middle attack. It is inherited from the ECDH key exchange scheme. The second one is the impersonation attack. It is caused partly by the transferring of computational tasks. Under the impersonation attack, the attacker can impersonate the party who undertakes more computational tasks. Lacking authentication mechanisms is the main reason leading to these attacks. Therefore, we introduce a number of authentication measures that help to remove these attacks.

3.1 Scenario

3.1.1 Background

In the rest of this thesis, we refer the communications between two devices with unbalanced computation resources as “unbalanced communications”. The background of the unbalanced communications is summarized as follows.

Unbalanced communications in the past. Mobile terminals were once constrained devices with limited computational resources and poor power supply. There was considerable interest in designing security protocols for communications between a mobile terminal and a server (or base station). A typical solution is transferring computational tasks from the mobile terminal to the server during the key establish processes between them. For example, the protocol in [82] lets the server carry out one exponentiation on behalf of the mobile terminal in the widely applied Diffie-Hellman (DH) key exchange process. Since the exponentiation is a time-consuming operation, the computational cost on the mobile terminal is significantly reduced.

Unbalanced communications in the present. The modern society has witnessed the tremendous increase in the availability of computational resources. Nowadays, mobile terminals are able to offer quite impressive computational resources. However, there still are large numbers of devices with limited computational power, such as battery-powered and wirelessly connected sensors that are widely used in environment monitoring, water-quality monitoring, eHealth and so on. Moreover, these computationally limited devices are even much less powerful than the mobile terminals that were used many years before. As a result, ECDH key exchange protocols are widely adopted in recent years since they provide higher level of security with less computational requirements; and there is an urgent requirement for unbalancing computations in the ECDH key exchange protocols.

3.1.2 Features

The core feature of the scenario is the unequal computational resources of the communicating parties. In addition, the scenario should also has two other features. We summarize the features as follows.

- Unequal computational resources. The two communicating parties have significantly unequal computational resources. It is inconvenient or infeasible for the computationally limited party to undertake heavy computational tasks. This feature requires unbalancing computations of protocols for the scenario.

- Demand for security. The communications require security, such as authentication between the parties, and authentication, integrity and confidentiality of the messages. This feature requires security mechanisms for the scenario.

- Security vulnerability of main communication channels. The main communication channels are vulnerable to attacks. This feature requires security mechanisms, especially key establishment, for the scenario.

3.2 Unbalancing Computations In ECDH Protocol

Two methods are introduced to unbalance the computations in the ECDH key exchange scheme. The first one lets the responder carry out one scalar multiplication on behalf of the initiator. The second one lets the initiator to undertake one scalar multiplication on behalf of the responder. Since the scalar multiplication is the most time-consuming operation in the ECDH scheme, the methods significantly reduce the computational cost on the initiator or the responder. The two methods are named as UECDH. The resulting schemes are named as UECDH schemes.

3.2.1 Initialization

Before the execution of the UECDH schemes, the initiator and the responder shall possess their private and public keys respectively. The private keys should be integers belonging to the same finite field Z_q^*. The public keys should be points on the same elliptic curve E with the base point G. Denote the initiator by A and the responder by B. Formally, the initialization procedure generates the following values:

- Common parameters shared by A and B: $comm = (Z_q^*, E, G)$.

- Private and public keys of A: (SK_A, PK_A) where $SK_A \in Z_q^*$ and $PK_A = SK_A \times G$. SK_A is secretly held by A. PK_A is a public information that can be obtained by B.

- Private and public keys of B: (SK_B, PK_B) where $SK_B \in Z_q^*$ and $PK_B = SK_B \times G$. SK_B is secretly held by B. PK_B is a public information that can be obtained by A.

Note that the initialization does not belong to the UECDH key exchange process since the parameters and keys are long-term values. These values need not to be generated in every execution of the UECDH key exchange schemes.

3.2.2 Transferring Computational Tasks From A to B

Suppose the initiator A is a computationally limited device, and the responder B is much more powerful than A. The following UECDH scheme transfers one scalar multiplication from A to B.

1. A generates a random value $R_A \in Z_q^*$, and computes

$$U_A = R_A + SK_A.$$

Then A sends U_A and PK_A to B, i.e.,

$$A \rightarrow B : U_A, PK_A$$

2. B generates a random value $R_B \in Z_q^*$, and computes

$$U_B = R_B + SK_B$$

and

$$T_B = U_B \times G.$$

Then B sends T_B and PK_B to A, i.e.,

$$B \rightarrow A : T_B, PK_B$$

Table 3.1: Comparison of scalar multiplication on A and B.

Scalar multiplication on	A	B	Total
ECDH	2	2	4
UECDH in Section 3.2.2	1	3	4

3. A computes the shared secret through the following equation:

$$K_A = R_A \times (T_B - PK_B).$$

B computes the shared secret through the following equations:

$$T_A = U_A \times G,$$

$$K_B = R_B \times (T_A - PK_A).$$

The above scheme requires one scalar multiplication on A, and three scalar multiplications on B (Table 3.1). Compared with the ECDH scheme, it significantly reduces the computational cost on the initiator A.

Below we will prove that $K_A = K_B$.

Proof.

$$\begin{aligned}
K_A &= R_A \times (T_B - PK_B) \\
&= R_A \times ((R_B + SK_B) \times G - PK_B) \\
&= R_A \times (R_B \times G + SK_B \times G - PK_B) \\
&= R_A \times (R_B \times G + PK_B - PK_B) \\
&= R_A \times (R_B \times G) \\
&= R_A \times R_B \times G
\end{aligned}$$

$$
\begin{aligned}
K_B &= R_B \times (T_A - PK_A) \\
&= R_B \times ((R_A + SK_A) \times G - PK_A) \\
&= R_B \times (R_A \times G + SK_A \times G - PK_A) \\
&= R_B \times (R_A \times G + PK_A - PK_A) \\
&= R_B \times (R_A \times G) \\
&= R_B \times R_A \times G \\
&= R_A \times R_B \times G \\
&= K_A
\end{aligned}
$$

$\square$

3.2.3 Transferring Computational Tasks From B to A

Suppose the initiator A has abundant computational resources, and the responder B is a computationally limited device. The following UECDH protocol transfers one scalar multiplication from B to A.

1. A generates a random value $R_A \in Z_q^*$, and computes

$$U_A = R_A + SK_A$$

and

$$T_A = U_A \times G.$$

Then A sends T_A and PK_A to B, i.e.,

$$A \rightarrow B : T_A, PK_A.$$

Table 3.2: Comparison of scalar multiplication on A and B.

Scalar multiplication on	A	B	Total
ECDH	2	2	4
UECDH in Section 3.2.3	3	1	4

2. B generates a random value $R_B \in Z_q^*$, and computes

$$U_B = R_B + SK_B.$$

Then B sends U_B and PK_B to A, i.e.,

$$B \to A : U_B, PK_B.$$

3. A computes the shared secret through the following equations:

$$T_B = U_B \times G,$$

$$K_A = R_A \times (T_B - PK_B).$$

B computes the shared secret through the following equation:

$$K_B = R_B \times (T_A - PK_A).$$

The above scheme successfully transfers the computation of T_B from B to A. As a result, A undertakes three scalar multiplications, and B undertakes only one scalar multiplication (Table 3.2). Compared with the ECDH scheme, this protocol significantly reduces the computational cost on the limited responder B. The shared secrets computed by A and B are equivalent. The proof is similar as the one in Section 3.2.2.

3.3 Security Issues

The UECDH key exchange schemes in Section 3.2.2 and Section 3.2.3 are vulnerable to two attacks: the man-in-the-middle attack and the impersonation attack. We illustrate these attacks, identify the reasons causing these attacks, and introduce four methods to remove these attacks as follows.

3.3.1 The Man-In-The-Middle Attack

The vulnerability to the man-in-the-middle attack is inherited from the ECDH key exchange scheme. The fundamental limitation of the ECDH scheme is that it does not contain any authentication of the exchanged messages. Similarly, the UECDH schemes in Section 3.2.2 and Section 3.2.3 do not contain any authentication of the exchange messages. It leads to the vulnerability to the man-in-the-middle attack. Denote the man-in-the-middle attacker by C, and the private and public keys of C by SK_C and PK_C. The attacks to the two UECDH schemes are described as follows. They are also illustrated in Figure 3.1 and Figure 3.2.

The man-in-the-middle attack to the UECDH scheme in Section 3.2.2

1. A generates a random value $R_A \in Z_q^*$, and computes

   $$U_A = R_A + SK_A.$$

   Then A sends U_A and PK_A to B, i.e.,

   $$A \to B : U_A, PK_A$$

2. C firstly intercepts (U_A, PK_A).

   Secondly, C generates a random value $R_C \in Z_q^*$, and computes

   $$U_C = R_C + SK_C$$

At last, C sends a forged message (U_C, PK_C) to B, i.e.,

$$C \rightarrow B : U_C, PK_C$$

3. B generates a random value $R_B \in Z_q^*$, and computes

$$U_B = R_B + SK_B$$

and

$$T_B = U_B \times G.$$

Then B sends T_B and PK_B to A, i.e.,

$$B \rightarrow A : T_B, PK_B$$

4. C firstly intercepts (T_B, PK_B).

Then C computes

$$T_C = U_C \times G.$$

Finally, C sends a forged message (T_C, PK_C) to A, i.e.,

$$C \rightarrow A : T_C, PK_C$$

5. A computes the shared secret through the following equation:

$$K_A = R_A \times (T_C - PK_C).$$

B computes the shared secret through the following equations:

$$T_C = U_C \times G,$$

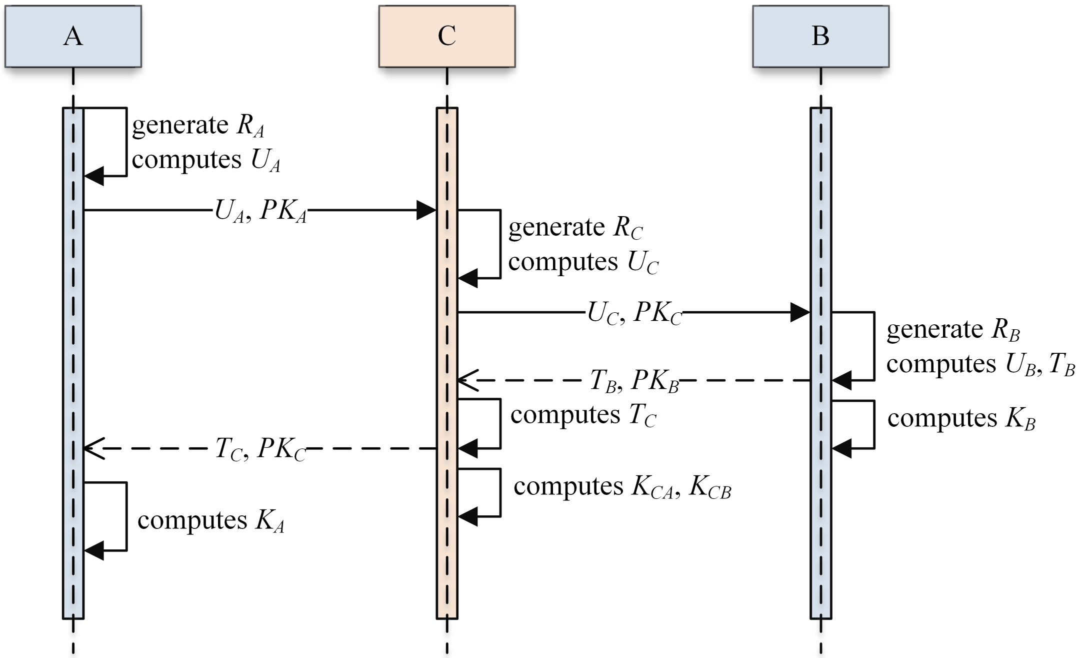

Figure 3.1: The man-in-the-middle attack to the UECDH scheme in Section 3.2.2.

$$K_B = R_B \times (T_C - PK_C).$$

C computes the shared secrets with A and B through the following equations:

$$T_A = U_A \times G,$$

$$K_{CA} = R_C \times (T_A - PK_A),$$

$$K_{CB} = R_C \times (T_B - PK_B).$$

Now we will prove that $K_A = K_{CA}$ and $K_B = K_{CB}$.

Proof.

$$
\begin{aligned}
K_A &= R_A \times (T_C - PK_C) \\
&= R_A \times ((R_C + SK_C) \times G - PK_C) \\
&= R_A \times (R_C \times G + PK_C - PK_C) \\
&= R_A \times (R_C \times G) \\
&= R_A \times R_C \times G
\end{aligned}
$$

$$
\begin{aligned}
K_{CA} &= R_C \times (T_A - PK_A) \\
&= R_C \times ((R_A + SK_A) \times G - PK_A) \\
&= R_C \times (R_A \times G + PK_A - PK_A) \\
&= R_C \times R_A \times G = K_A
\end{aligned}
$$

$$
\begin{aligned}
K_B &= R_B \times (T_C - PK_C) \\
&= R_B \times ((R_C + SK_C) \times G - PK_C) \\
&= R_B \times (R_C \times G + PK_C - PK_C) \\
&= R_B \times (R_C \times G) \\
&= R_B \times R_C \times G
\end{aligned}
$$

$$
\begin{aligned}
K_{CB} &= R_C \times (T_B - PK_B) \\
&= R_C \times ((R_B + SK_B) \times G - PK_B) \\
&= R_C \times (R_B \times G + PK_B - PK_B) \\
&= R_C \times R_B \times G = K_B
\end{aligned}
$$

□

Therefore, after the above attack, C shares a secret $K_{CA} = K_A$ with A and a secret $K_{CB} = K_B$ with B. However, both A and B think they share a secret with each other.

The man-in-the-middle attack to the UECDH scheme in Section 3.2.3

1. A generates a random value $R_A \in Z_q^*$, and computes

$$U_A = R_A + SK_A,$$

and

$$T_A = U_A \times G.$$

Then A sends T_A and PK_A to B, i.e.,

$$A \rightarrow B : T_A, PK_A$$

2. C firstly intercepts (T_A, PK_A).

Then C generates a random value $R_C \in Z_q^*$, and computes

$$U_C = R_C + SK_C,$$

$$T_C = U_C \times G.$$

At last, C sends a forged message (T_C, PK_C) to B, i.e.,

$$C \rightarrow B : T_C, PK_C$$

3. B generates a random value $R_B \in Z_q^*$, and computes

$$U_B = R_B + SK_B$$

Then B sends U_B and PK_B to A, i.e.,

$$B \rightarrow A : T_B, PK_B.$$

4. C firstly intercepts (T_B, PK_B).

   Then C sends a forged message (U_C, PK_C) to B, i.e.,

$$C \rightarrow B : U_C, PK_C$$

5. A computes the shared secret through the following equations:

$$T_C = U_C \times G,$$

$$K_A = R_A \times (T_C - PK_C).$$

   B computes the shared secret through the following equation:

$$K_B = R_B \times (T_C - PK_C).$$

   C computes the shared secrets with A and B through the following equations:

$$K_{CA} = R_C \times (T_A - PK_A),$$

$$T_B = U_B \times G,$$

$$K_{CB} = R_C \times (T_B - PK_B).$$

Similarly as we proved before, $K_{CA} = K_A$ and $K_{CB} = K_B$. Therefore, after the above attack, C shares a secret $K_{CA} = K_A$ with A and a secret $K_{CB} = K_B$ with B. However, both A and B think they share a secret with each other.

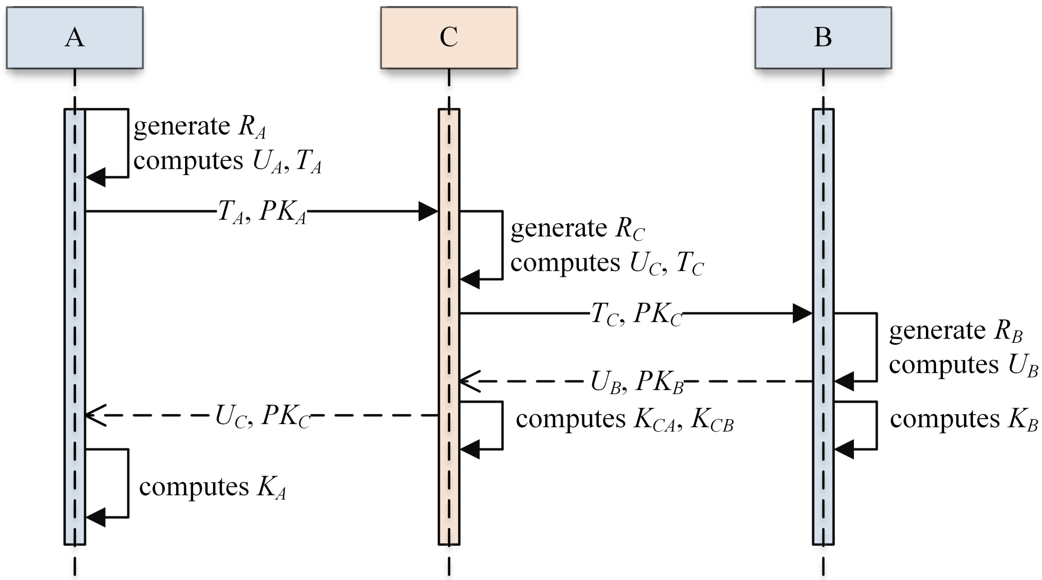

Figure 3.2: The man-in-the-middle attack to the UECDH scheme in Section 3.2.3.

3.3.2 The Impersonation Attack

The impersonation attack is caused partly by the transferring of scaler multiplication from A to B (Section 3.2.2) or from B to A (Section 3.2.3). In particular, in the UECDH scheme in Section 3.2.2, A computes the shared secret from R_A, T_B and PK_B. The computation of the secret does not involve the long-term secret of B (i.e. SK_B); therefore, an attacker can impersonate B and execute the scheme with A. Similarly, in the UECDH scheme in Section 3.2.3, B computes the shared secret from R_B, T_A and PK_A. The computation of the secret does not involve the long-term secret (i.e., SK_A) of A; as a result, an attacker can impersonate A and execute the scheme with B.

Denote the impersonation attacker by C, and the private and public keys of C by SK_C and PK_C. These attacks are described as follows. They are also illustrated in Figure 3.3 and Figure 3.4.

The impersonation attack to the UECDH scheme in Section 3.2.2

1. A generates a random value $R_A \in Z_q^*$, and computes

$$U_A = R_A + SK_A.$$

Then A sends U_A and PK_A to B, i.e.,

$$A \to B : U_A, PK_A$$

2. C firstly intercepts and blocks (U_A, PK_A).

Secondly, C generates a random value $R_C \in Z_q^*$, and computes

$$T_C = R_C \times G + PK_B$$

At last, C impersonates B and sends (T_C, PK_B) to A, i.e.,

$$C \to A : T_C, PK_C$$

3. A computes the shared secret through the following equation:

$$K_A = R_A \times (T_C - PK_B).$$

C computes the shared secrets with A and B through the following equations:

$$T_A = U_A \times G,$$

$$K_C = R_C \times (T_A - PK_A),$$

Below we will prove that $K_A = K_C$.

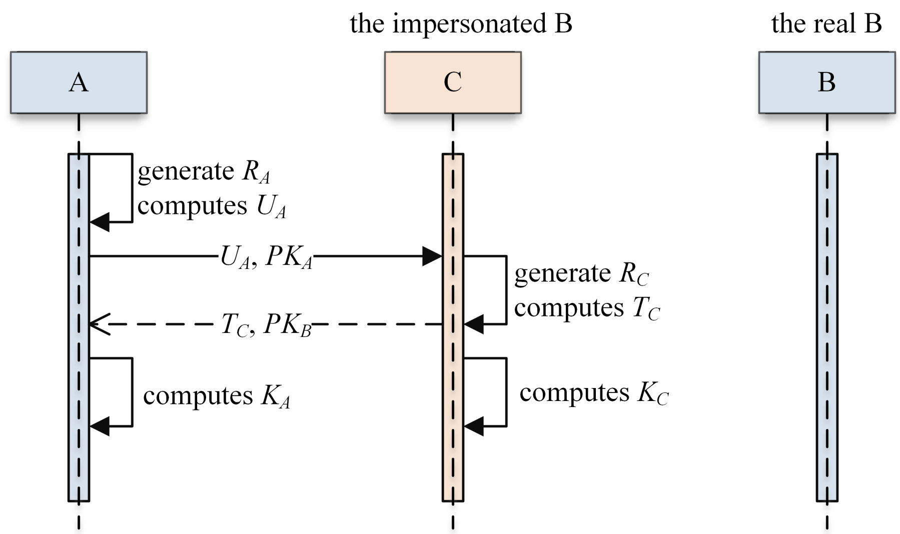

Figure 3.3: The impersonation attack to the UECDH scheme in Section 3.2.2.

Proof.

$$\begin{aligned} K_A &= R_A \times (T_C - PK_B) \\ &= R_A \times (R_C \times G + PK_B - PK_B) \\ &= R_A \times (R_C \times G) \\ &= R_A \times R_C \times G. \end{aligned}$$

$$\begin{aligned} K_C &= R_C \times (T_A - PK_A) \\ &= R_C \times ((R_A + SK_A) \times G - PK_A) \\ &= R_C \times (R_A \times G + PK_A - PK_A) \\ &= R_C \times (R_A \times G) \\ &= R_C \times R_A \times G \\ &= K_A \end{aligned}$$

□

Therefore, after the above attack, C establishes a shared secret $K_A = K_C$ with A.

However, A thinks he (or she) shares a secret with B.

The impersonation attack to the UECDH scheme in Section 3.2.3

1. C generates a random value $R_C \in Z_q^*$, and computes

$$T_C = R_C \times G + PK_A.$$

   Then C impersonates A and sends T_C and PK_A to B, i.e.,

$$C \to B : T_C, PK_A$$

2. B generates a random value $R_B \in Z_q^*$, and computes

$$U_B = R_B + SK_B$$

   Then B sends U_B and PK_B to C, i.e.,

$$B \to A : U_B, PK_B$$

3. C computes the shared secret through the following equations:

$$T_B = U_B \times G,$$

$$K_C = R_C \times (T_B - PK_B).$$

   B computes the shared secret through the following equation:

$$K_B = R_B \times (T_C - PK_A).$$

Bellow we will prove that $K_C = K_B$.

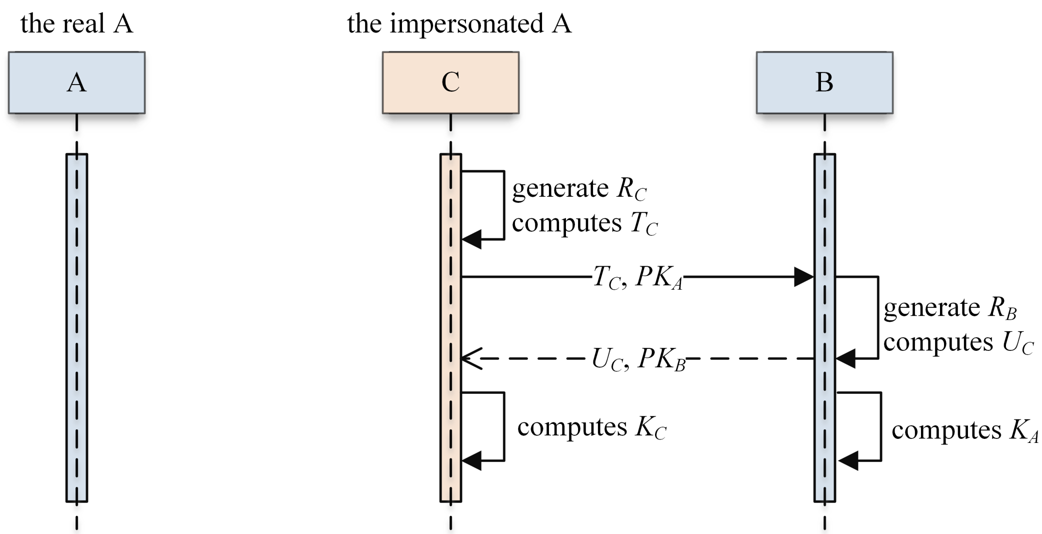

Figure 3.4: The impersonation attack to the UECDH scheme in Section 3.2.3.

Proof.

$$
\begin{aligned}
K_C &= R_C \times (T_B - PK_B) \\
&= R_C \times ((R_B + SK_B) \times G - PK_B) \\
&= R_C \times (R_B \times G + PK_B - PK_B) \\
&= R_C \times (R_B \times G) \\
&= R_C \times R_B \times G.
\end{aligned}
$$

$$
\begin{aligned}
K_B &= R_B \times (T_C - PK_A) \\
&= R_B \times ((R_C \times G + PK_A) - PK_A) \\
&= R_B \times (R_C \times G) \\
&= R_B \times R_C \times G \\
&= K_C.
\end{aligned}
$$

□

Therefore, after the above attack, C establishes a shared secret $K_C = K_B$ with B.

However, B thinks he (or she) shares a secret with A.

3.4 Solutions To The Security Issues

We have discussed the reasons for the security issues in the UECDH schemes. In order to solve these issues, we introduce a number of solutions to remove each attack. These solutions are not absolutely separated. It is recommended to combine and integrate some of them. This is illustrated through specific protocols in the following four chapters (4, 5, 6 and 7).

3.4.1 Removing The Man-In-The-Middle Attacks

The vulnerability to the man-in-the-middle attacks is caused by the lack of authentication messages in the schemes. Therefore, a direct method to remove this attack is adding authentication information in the exchanged messages. Bellow are three typical methods to establish authentication information.

- The method based on pre-shared secrets. This method requires the two parties securely share a secret in advance. With the shared the secret, the two parties can use MAC to authenticate the identities and exchanged messages.

- The method based on authenticated public keys. This method requires the two parties have the authenticated public keys of each other. The two parties can use digital signature to authenticate each other and the exchange messages.

- The method based on OOB channels with appropriate security features. This method requires the two parties can establish OOB channels in addition to the basic communicating channels. The OOB channels are used to establish and transfer authentication information.

3.4.2 Removing The Impersonation Attacks

The vulnerability to the impersonation attacks is caused partly by the lack of the long-term secret key of one party (who carries out more scalar multiplications) in computing the shared secret. As a result, this party can be impersonated by the attacker. The authentication methods introduced in Section 3.4.1 can help to remove the impersonation attacks. Therefore, the recommended methods to remove the impersonation attacks include:

- The method based on pre-shared secrets.
- The method based on authenticated public keys.
- The method based on OOB channels with appropriate security features.

3.4.3 Discussion

The man-in-the-middle attack and the impersonation attack are not completely separated from each other. More specifically, in the man-in-the-middle attack, the attacker impersonates A and communicates with B; meanwhile, the attacker impersonates B and communicates with A. The proposed solutions to remove the two attacks are not separated neither. The advisable solution should combine the solutions to remove both the two types of attacks.

3.5 Chapter Summary

The majority of AKE protocols require equivalent computational cost on the parties. However, in practice, many communications take place between two parties with fairly different computational resources, for example, a mobile phone and a cloud server, a sensor and a base station, and so on. It is significant to reduce the computational cost on a computationally limited device in an AKE protocol. An ingenious method has been illustrated by the unbalanced DH key exchange scheme. It transfers the time-consuming exponentiation from the computationally limited party to its much more powerful communicating partner.

The rapid development of communicating technologies interconnects numerous devices including many computationally limited sensors. It is highly recommended to base the security mechanisms for sensors on elliptic curve cryptographic schemes. For example, the ECDH-based AKE protocols are adopted by many communicating techniques and standards such as the Bluetooth, IEEE 802.15.6 and so on. In this context, it is significant to study how to unbalance computations in ECDH key exchange scheme and ECDH-based AKE protocols.

In this chapter, we studied how to unbalance computations in the ECDH key exchange scheme. Two UECDH key exchange schemes were proposed; and two attacks to the schemes were illustrated. The solutions to remove these attacks were discussed. In the following four chapters, we will apply these solutions to design UECDH-based AKE protocols that are resistant to these attacks.

Chapter 4

Password UECDH-based AKE Protocols

Password is a short pre-shared secret to establish authentication. It can be remembered by humans; and does not require secure memory which is often expensive. Therefore, password-based AKE protocols are popular in IoT scenarios where the devices are capability-limited and unable to securely store long pre-shared secrets. For example, the IEEE Standard 802.15.6 includes a password authenticated association protocol.

This chapter introduces password UECDH-based AKE protocols. The two parties share a password in advance of the protocols; and the password is input by users in each session of the protocols to achieve authentication. Firstly, we provide an overview of the communication model, attack model and security model of the protocols. Secondly, we will present the two password UECDH-based AKE protocols: Protocol I-A which requires less scalar multiplications on A than on B and Protocol I-B which requires less scalar multiplications on B than A. Thirdly, we analyze the security of the protocols according to the attack model and security model. In particular, we illustrate how the man-in-the-middle and impersonation attacks to the protocols fail. At last, we compare the performance of the two protocols with the IEEE PW protocol through theoretical evaluation and experimental test.

4.1 Overview

4.1.1 Communication Model

The communication model of a password UECDH-based AKE protocol is specified as follows.

- Participants. In each session of the protocol, there are two participants. The participants are denoted by their identities A and B. A is the initiator, and B is the responder. In particular, A and B have significantly different computational capabilities.

- Channels. The channels between A and B are normal channels.

4.1.2 Attack Model

The following assumptions and ability specification define what an attacker to a password UECDH-based AKE protocol is able and unable to do.

- Basic assumption 1. The attacker is unable to break the MAC algorithm.

- Basic assumption 2. The attacker is unable to reveal the password.

- Basic ability. The attacker is able to observe all messages, alter messages, insert new messages, delay messages or delete messages transmitted between A and B via normal channels.

- Stronger ability 1. The attacker is able to obtain any previous session keys.

- Stronger ability 2. The attacker is able to compromise the long-term secret keys of A or B.

4.1.3 Security Model

Under the above attack model, a password UECDH-based AKE protocol aims to achieve the following security goals:

- Key authentication under the attack model that the attacker has the basic ability.
- Key confidentiality under the attack model that the attacker has the basic ability.
- Key integrity under the attack model that the attacker has the basic ability.
- Key confirmation under the attack model that the attacker has the basic ability.
- Known-key security (key freshness) under the attack model that the attacker has the basic ability and the stronger ability 1.
- Forward secrecy under the attack model that the attacker has the basic ability and the stronger ability 2.
- Resistance to dictionary attacks under the attack model that the attacker has the basic ability.

4.2 Protocol Description

4.2.1 Protocol I-A

Protocol I-A generates a shared secret for a computationally limited initiator A and a more powerful responder B. It transfers one scalar multiplication from A to B. The protocol is described through the following procedures: initialization, key exchange and session keys computation. It is also illustrated in Figure 4.1.

Initialization

Before the execution of Protocol I-A, the initiator and the responder should obtain their private and public keys respectively. The private keys should be integers in the same finite field. The public keys should be points on the same elliptic curve. In addition, the initiator and the responder should share a password and a one-way function that maps the password to a point on the elliptic curve.

Denote the initiator by A, the responder by B, the finite field by Z_q^*, the elliptic curve by E, the base point of E by G, the private and public keys of A by SK_A and PK_A, the

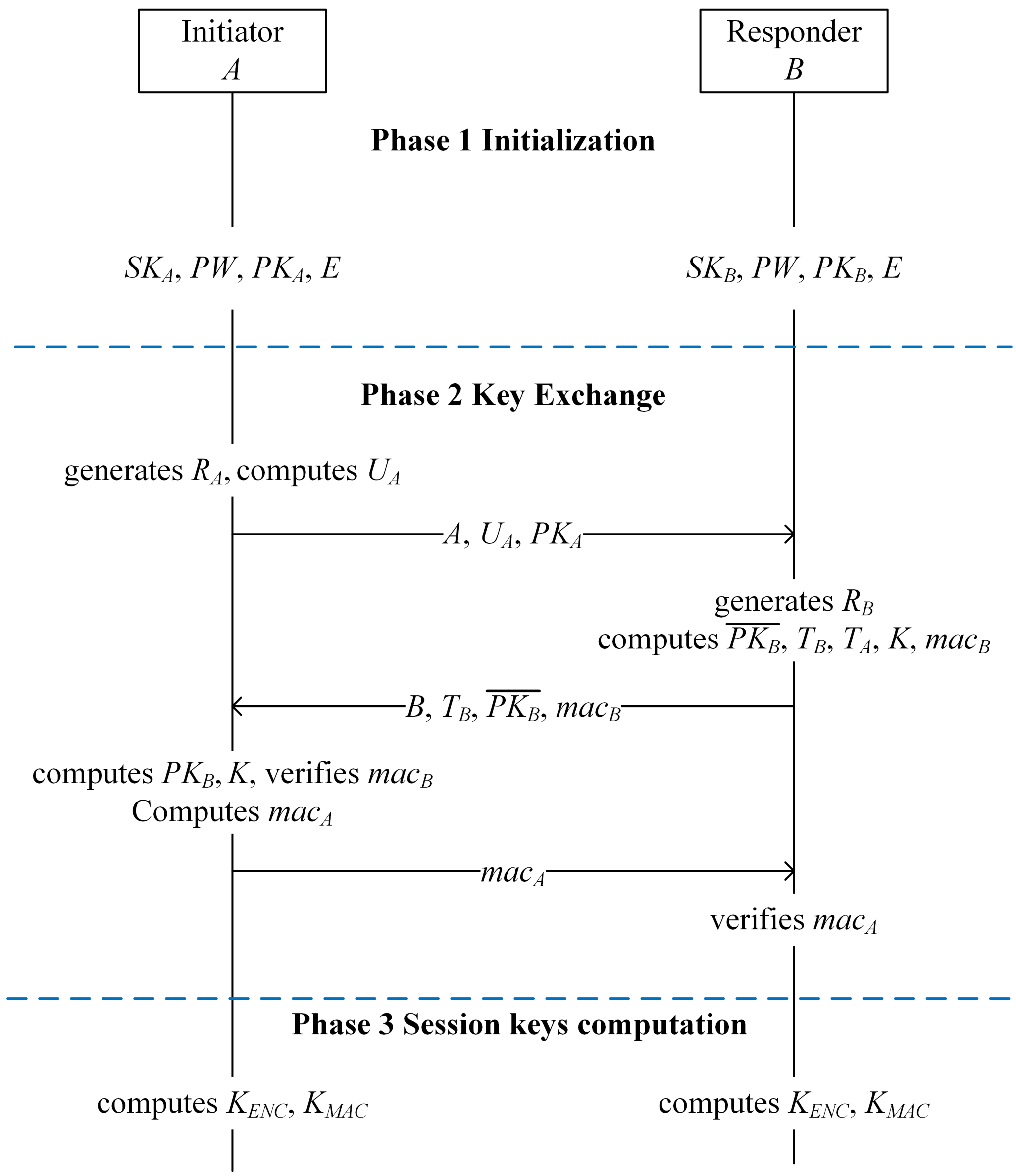

Figure 4.1: Protocol I-A.

private and public keys of B by SK_B and PK_B, the password by PW and the one-way function by Q. Formally, the initialization procedure generates the following values:

- Common parameters shared by A and B: $comm = (Z_q^*, E, G, Q)$.
- Secret information shared by A and B: PW that is kept secret and do not be stored in the device.
- Information held only by A: PK_A and SK_A where SK_A should be securely stored.
- Information held only by B: PK_B and SK_B; and both should be securely stored.

Key Exchange

- A generates a random value $R_A \in Z_q^*$, and computes

$$U_A = R_A + SK_A.$$

Then A sends B M_1 as follows:

$$A \to B : M_1 = (A, U_A, PK_A).$$

- Upon receiving M_1, B firstly generates a random value $R_B \in Z_q^*$ and computes T_B and $\overline{PK_B}$ as follows:

$$U_B = R_B + SK_B,$$

$$T_B = U_B \times G,$$

$$\overline{PK_B} = PK_B - Q(PW).$$

Secondly, B computes the shared secret K_B as follows:

$$T_A = U_A \times G,$$

$$K_B = R_B \times (T_A - PK_A).$$

Thirdly, B computes a message authentication code mac_B as follows:

$$mac_B = \mathrm{MAC}(K_{Bx}, B\|T_B\|\overline{PK_B})$$

where K_{Bx} denotes the x coordinate of K_B.

Finally, B sends A M_2 as follows:

$$B \rightarrow A : M_2 = (B, T_B, \overline{PK_B}, mac_B).$$

- Upon receiving M_2, A firstly computes the shared key K_A as follows:

$$PK_B = \overline{PK_B} + Q(PW)$$

$$K = R_A \times (T_B - PK_B).$$

Secondly, A verifies mac_B as follows:

$$\mathrm{VER}(K_{Ax}, B\|T_B\|\overline{PK_B}, mac_B) = \begin{cases} 1, & valid \\ 0, & invalid \end{cases}$$

where K_{Ax} denotes the x coordinate of K_A.

Thirdly, if mac_B is valid, A computes a message authentication code mac_A as follows:

$$mac_A = \mathrm{MAC}(K_{Ax}, A\|U_A\|PK_A).$$

Finally, A sends B M_3 as follows

$$A \rightarrow B : M_3 = mac_A$$

- Upon receiving mac_A, B verifies mac_A as follows:

$$\mathrm{VER}(K_{Bx}, A\|U_A\|PK_A, mac_A) = \begin{cases} 1, & valid \\ 0, & invalid \end{cases}$$

Session Keys Computation

If mac_B is valid, A derives the session keys from K_{A_y} as follows:

$$K_{ENC} = \mathrm{F}(K_{A_y}, 1),$$

$$K_{MAC} = \mathrm{F}(K_{A_y}, 2),$$

where K_{A_y} denotes the y coordinate of K_A.

If mac_A is valid, B derives the session keys from K_{B_y} as follows:

$$K_{ENC} = \mathrm{F}(K_{B_y}, 1),$$

$$K_{MAC} = \mathrm{F}(K_{B_y}, 2),$$

where K_{B_y} denotes the y coordinate of K_B.

After the AKE process, K_{ENC} will be used by symmetric encryption algorithms; and K_{MAC} will be used by MAC algorithms.

4.2.2 Protocol I-B

Protocol I-B shares a secret between a powerful initiator A and a computationally limited responder B. It transfers one scalar multiplication from B to A. We describe the protocol through the following procedures: initialization, key exchange and session keys computation in detail. In addition, we also illustrate the procedures through Figure 4.2.

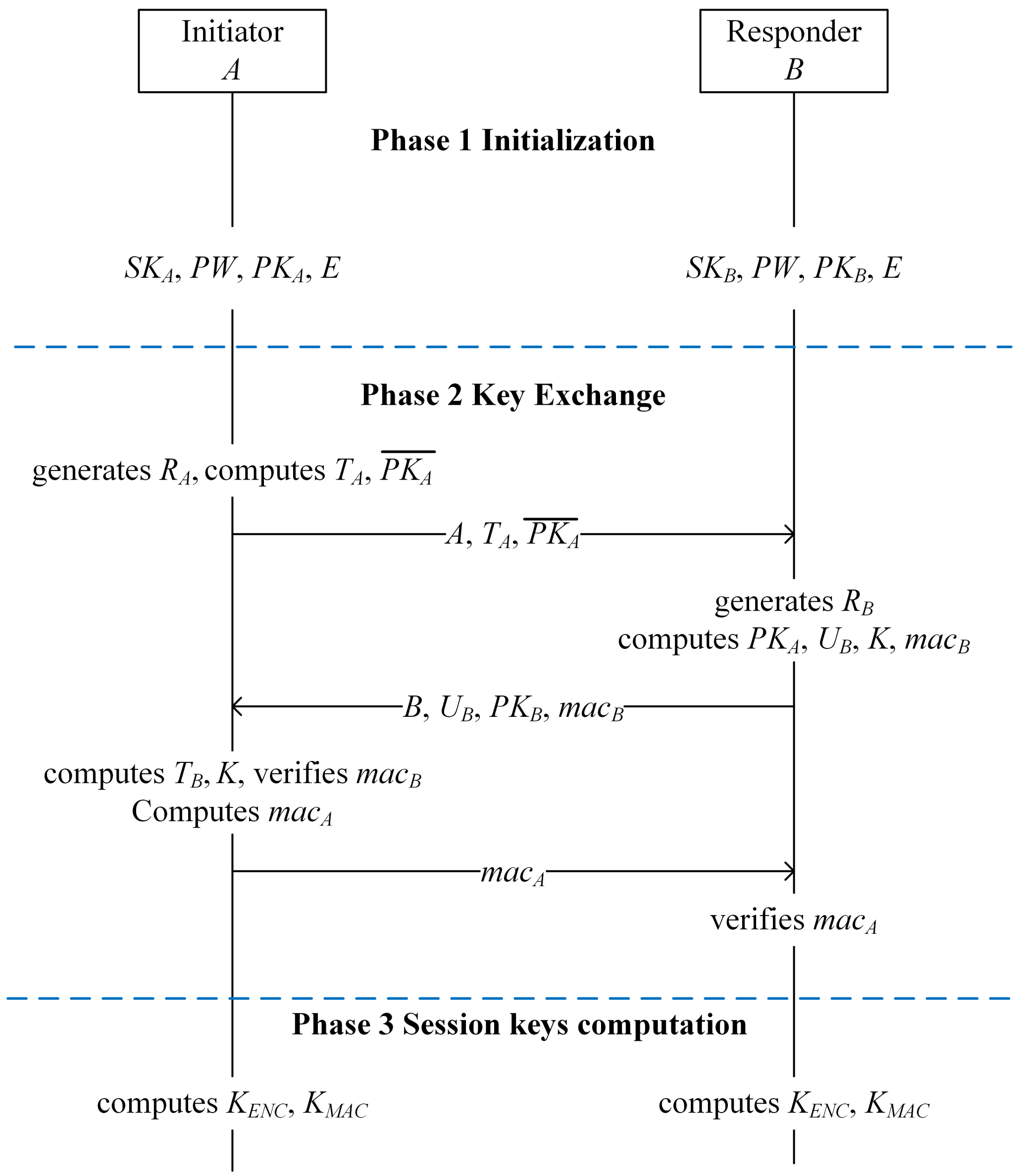

Figure 4.2: Protocol I-B.

Initialization

The initialization here is similar with that of Protocol I-A. Let the notations A, B, Z_q^*, E, G,SK_A, PK_A, SK_B, PK_B PW and Q be the same as we specified in Section 4.2.1. The initialization procedure produces the following values:

- Common parameters shared by A and B: $comm = (Z_q^*, E, G, Q)$.
- Secret information shared by A and B: PW.
- Information held only by A: PK_A and SK_A; and both should be securely stored.
- Information held only by B: PK_B and SK_B where SK_B should be securely stored.

Key Exchange

- A generates a random value $R_A \in Z_q^*$. A computes

$$U_A = R_A + SK_A,$$

$$T_A = U_A \times G.$$

$$\overline{PK_A} = PK_A - Q(PW)$$

Then A sends B M_1 as follows:

$$A \to B : M_1 = (A, T_A, \overline{PK_A}).$$

- Upon receiving M_1, B firstly generates a random value $R_B \in Z_q^*$ and computes

$$U_B = R_B + SK_B.$$

Secondly, B computes the shared secret K_B as follows:

$$PK_A = \overline{PK_A} + Q(PW),$$

$$K_B = R_B \times (T_A - PK_A).$$

Thirdly, B computes a message authentication code mac_B as follows:

$$mac_B = \mathrm{MAC}(K_{Bx}, B\|T_B\|PK_B).$$

Finally, B sends A M_2 as follows:

$$B \to A : M_2 = (B, U_B, PK_B, mac_B).$$

- Upon receiving M_2, A firstly computes the shared secret as follows:

$$T_B = U_B \times G,$$

$$K_A = R_A \times (T_B - PK_B).$$

Secondly, A verifies mac_B as follows:

$$\mathrm{VER}(K_{Ax}, B\|U_B\|PK_B, mac_B) = \begin{cases} 1, & valid \\ 0, & invalid \end{cases}$$

Thirdly, if mac_B is valid, A computes message authentication code mac_A as follows:

$$mac_A = \mathrm{MAC}(K_{Ax}, A\|T_A\|\overline{PK_A}).$$

Finally, A sends B M_3 as follows:

$$A \to B : M_3 = mac_A$$

- Upon receiving mac_A, B verifies mac_A as follows:

$$\mathrm{VER}(K_{Bx}, A\|T_A\|\overline{PK_A}, mac_A) = \begin{cases} 1, & valid \\ 0, & invalid \end{cases}$$

Session Keys Computation

If mac_B is valid, A derives the session keys from K_{Ay} as follows:

$$K_{ENC} = \mathrm{F}(K_{Ay}, 1),$$

$$K_{MAC} = \mathrm{F}(K_{Ay}, 2).$$

If mac_A is valid, B derives the session keys from K_{By} as follows:

$$K_{ENC} = \mathrm{F}(K_{By}, 1),$$

$$K_{MAC} = \mathrm{F}(K_{By}, 2).$$

4.3 Security

This section illustrates that the two password UECDH-based AKE protocols achieve the security goals (Section 4.1.3) under the attack model (Section 4.1.2). For each security goal, we provide a proposition that states a security feature, and prove how the proposition stands. In addition, we also show how the two protocols resist the man-in-the-middle attack and the impersonation attack.

4.3.1 Security Features

Proposition 4.1 (Key authentication of Protocol I-A and I-B)**.** *Assume there is an attacker C who can observe all messages, alter messages, insert new messages, delay messages or delete messages transmitted between A and B via normal channels. After a completed* *run of Protocol I-A (or I-B), A (or B) believes that he (or she) shares a secret with B (or A) other than any other party.*

Proof. (1) A completed run of Protocol I-A is defined by the validation of mac_A and mac_B; and the validation of mac_A and mac_B guarantees the authenticity of U_A, PK_A, T_B and $\overline{PK_B}$.

(2) A computes the shared secret K_A from $R_A \times (T_B - PK_B)$. R_A is the random value generated by A. T_B is authenticated according to (1). PK_B is computed from $\overline{PK_B} + Q(PW)$ where PW is a pre-shared secret stored by A and $\overline{PK_B}$ is authenticated according to (1). Therefore, A believes he or she shares a secret with B other than any other party.

(3) B computes the shared secret K_B from $R_B \times (U_A \times G - PK_A)$. R_B is the random value generated by B. U_A and PK_A are authenticated according to (1). Therefore, B believes he or she shares a secret with A other than any other party.

According to (2) and (3), we have the conclusion that Protocol I-A provides key authentication. Similarly we can prove that Protocol I-B provides key authentication. □

Proposition 4.2 (Key confidentiality of Protocol I-A and I-B)**.** *Assume there is an attacker C who can observe all messages, alter messages, insert new messages, delay messages or delete messages transmitted between A and B via normal channels. After a completed run of Protocol I-A (or I-B), the attacker is unable to derive the shared key of A and B.*

Proof. (1) The shared secret can be computed from any of the following equations:

$$K_A = R_A \times (T_B - PK_B),$$

$$K_B = R_B \times (U_A \times G - PK_A),$$

$$K = R_A \times R_B \times G,$$

Therefore, R_A or R_B is required to compute the shared secret.

(2) R_A is hidden by the following equation:

$$U_A = R_A + SK_A.$$

Therefore, SK_A is required to compute R_A.

R_B is hidden by the following equation:

$$T_B = (R_B + SK_B) \times G.$$

Therefore, SK_B is required to compute R_B.

(3) According to the attack model, C has neither SK_A nor SK_B. C is unable to compute R_A or R_B. Therefore, C is unable to compute $K_A = K_B = K$.

Therefore, we have the conclusion that Protocol I-A provides key confidentiality. Similarly we can prove that Protocol I-B provides key confidentiality. □

Proposition 4.3 (Key integrity of Protocol I-A and I-B)**.** *Assume there is an attacker C who can observe all messages, alter messages, insert new messages, delay messages or delete messages transmitted between A and B via normal channels. After a completed run of Protocol I-A (or I-B), A and B compute the equal secret.*

Proof. (1) As we proved in Theorem 5.1, a completed run of Protocol I-A implies the authenticity of U_A, PK_A, T_B and $\overline{PK_B}$ (or PK_B).

(2) The secret K_A is computed by A from

$$\begin{aligned} K_A &= R_A \times (T_B - PK_B) \\ &= R_A \times (U_B \times G - PK_B) \\ &= R_A \times ((R_B + SK_B) \times G - PK_B) \\ &= R_A \times R_B \times G. \end{aligned}$$

The secret K_B is computed by B from

$$\begin{aligned} K_B &= R_B \times (U_A \times G - PK_A) \\ &= R_B \times ((R_A + SK_A) \times G - PK_A) \\ &= R_B R_A \times G \\ &= R_A \times R_B \times G \\ &= K_A \end{aligned}$$

Therefore, after a completed run of Protocol I-A, A and B compute the equal secret. We have the conclusion that Protocol I-A provides key integrity. Similarly we can prove that Protocol I-B provides key integrity. □

Proposition 4.4 (Key confirmation of Protocol I-A and I-B)**.** *Assume there is an attacker C who can observe all messages, alter messages, insert new messages, delay messages or delete messages transmitted between A and B via normal channels. After a completed run of Protocol I-A (or I-B), both A and B have received evidence confirming that the other party knows the secret.*

Proof. (1) As we proved in Proposition 4.1, a complete run of Protocol I-A is defined the the validation of mac_A and mac_B. Therefore, after a completed run of Protocol I-A, both A and B have received and validated mac_B and mac_A respectively.

(2) mac_A is computed by A and takes the shared secret as one of the inputs. It is the evidence confirming that A knows the shared secret.

(3) mac_B is computed by B and takes the shared secret as one of the inputs. It is the evidence confirming that B knows the shared secret.

According to (1) and (2), after a completed run of Protocol I-A, B has received evidence confirming A knows the shared secret. According to (1) and (3), after a completed run of Protocol I-A, A has received evidence confirming B knows the shared secret. Therefore, Protocol I-A provides key confirmation. Similarly we can prove that Protocol I-B provides key confirmation. □

Proposition 4.5 (Known-key security (key freshness) of Protocol I-A and I-B)**.** *Assume there is an attacker C who can observe all messages, alter messages, insert new messages, delay messages or delete messages transmitted between A and B via normal channels. In addition, C is able to obtain any previous session keys. After a completed run of Protocol I-A (or I-B), C is unable to derive the shared secret from the previous session keys.*

Proof. In Protocol I-A, the computation of the secret takes the R_A and R_B as the inputs. Since R_A and R_B are random values generated by A and B respectively in the key exchange procedure, in each run of Protocol I-A the values are unique. Therefore, the secret is fresh in each run of the protocol. That is, Protocol I-A provides known-key security (key freshness). Similarly we can prove that Protocol I-B provides known-key security. □

Proposition 4.6 (Forward secrecy of Protocol I-A and I-B)**.** *Assume there is an attacker C who can observe all messages, alter messages, insert new messages, delay messages or delete messages transmitted between A and B via normal channels. In addition, C compromises the long-term secrets of A and B. C is unable to derive the previous session keys.*

Proof. (1) According to the attack model, C obtains the following messages transmitted via normal channels:

$$(E, G, Z_q^*, A, U_A, PK_A, B, T_B, \overline{PK_B}).$$

In addition, C compromises the long-term secrets PW and SK_A and SK_B.

(2) The values of R_A and R_B are short-term secrets. In practice, they will be cleared after use. Therefore, for a previous run of the protocol, R_A and R_B are unknown to C. As we proved in Theorem 4.2, R_A or R_B is required to compute the shared secret. Therefore, C is unable to compute the secret of the previous run of the protocol.

Therefore, we have the conclusion that Protocol I-A provides forward secrecy. Similarly we can prove that Protocol I-B provides forward secrecy. □

Proposition 4.7 (Resistance to dictionary attacks of Protocol I-A and I-B)**.** *Assume there is a passive attacker C who can observe all messages, alter messages, insert new messages,* *delay messages or delete messages transmitted between A and B via normal channels. C is unable to eliminate a significant number of possible passwords.*

Proof. In Protocol I-A, the information that is related with the password is $\overline{PK_B} = PK_B + Q(PW)$ which is a point in the elliptic curve. A dictionary attack on the combination of $(\overline{PK_B}, PW)$ (2^{256} or more) is much difficult than directly guessing PW. Therefore, Protocol I-A is resistant to dictionary attacks. Similarly, we can prove that Protocol I-B is resistant to dictionary attacks. $\square$

4.3.2 Resistance to Attacks

The password UECDH-based AKE protocols address the vulnerabilities to the man-in-the-middle attack and the impersonation attack. Below we illustrate how the two attacks fail.

Resistance to the man-in-the-middle attack

Assume C is a man-in-the-middle attacker to Protocol I-A between A and B. To launch the attack, C interacts with A and B as follows.

1. A generates a random value $R_A \in Z_q^*$ and computes

   $$U_A = R_A + SK_A.$$

   Then A sends B M_1 as follows:

   $$A \to B : M_1 = (A, U_A, PK_A).$$

2. C firstly intercepts M_1.

   Secondly, C generates a random value $R_C \in Z_q^*$ and computes

   $$U_C = R_C + SK_C.$$

At last, C sends a forged message (A, U_C, PK_C) to B, i.e.,

$$C \to B : M_1' = (A, U_C, PK_C)$$

3. Upon receiving the forged message M_1', B firstly generates a random value $R_B \in Z_q^*$, and computes

$$U_B = R_B + SK_B,$$

$$T_B = U_B \times G,$$

$$\overline{PK_B} = PK_B - Q(PW).$$

Secondly, B computes the shared secret as follows:

$$T_A' = U_C \times G,$$

$$K_B = R_B \times (T_A - PK_C).$$

Thirdly, B computes a message authentication code mac_B as follows:

$$mac_B = \mathrm{MAC}(K_{Bx}, B \| T_B \| \overline{PK_B}).$$

Then B sends A M_2 as follows:

$$B \to A : M_2 = (B, T_B, \overline{PK_B}, mac_B)$$

4. C firstly intercepts M_2.

Secondly, C computes

$$T_C = U_C \times G.$$

Thirdly, C intends to compute a shared secret $K_{CB} = K_B$ with B as follows:

$$K_{CB} = R_C \times (T_B - PK_B),$$

PK_B is hidden by $Q(PW)$ that is unknown by C. Therefore, C is unable to establish a share secret with B.

Fourthly, C intends to compute a shared secret $K_{CA} = K_A$ with A as follows:

$$K_{CA} = R_C \times (U_A \times G - PK_A).$$

K_A is computed by A from $K_A = R_A \times (T_B - PK_B)$. C can replace T_B with T_C in this step; however, C is unable to replace PK_B without PW. Therefore, C is unable to establish a shared secret with A. The attack fails.

Therefore, Protocol I-A is resistant to the man-in-the-middle attack. Similarly, Protocol I-B is resistant to the man-in-the-middle attack.

Resistance to the impersonation attack

Assume C is an impersonation attacker to Protocol I-A between A and B. To launch the attack, C impersonates B and interacts with A as follows.

1. A generates a random value $R_A \in Z_q^*$ and computes

$$U_A = R_A + SK_A.$$

Then A sends B M_1 as follows:

$$A \to B : M_1 = (A, U_A, PK_A).$$

2. C firstly intercepts and blocks M_1.

Table 4.1: Evaluation of computational costs of Protocol I-A, I-B and IEEE PW.

Computation Cost	Cost on A	Cost on B
Protocol I-A	$2\mathcal{H} + \mathcal{S}$	$2\mathcal{H} + 3\mathcal{S}$
Protocol I-B	$2\mathcal{H} + 3\mathcal{S}$	$2\mathcal{H} + \mathcal{S}$
IEEE PW	$2\mathcal{H} + 2\mathcal{S}$	$2\mathcal{H} + 2\mathcal{S}$

Secondly, C generates a random value $R_C \in Z_q^*$, and computes

$$T_C = R_C \times G + PK_B.$$

However, PK_B is hidden by $Q(PW)$ in the protocol. Without PK_B, C is unable to computes a correct T_C that makes $K_{CA} = R_C \times (U_A \times G - PK_A)$ equals to $K_A = R_A \times (T_C - PK_B)$. As a result, C is unable to compute a valid mac_B. Without the valid mac_B, A terminates the protocol. The attack fails.

Therefore, Protocol I-A is resistant to the impersonation attack. Similarly, Protocol I-B is resistant to the impersonation attack.

4.4 Performance

To study the performance of Protocol I-A and I-B, we choose the IEEE PW protocol in IEEE 802.15.6 as the benchmark. We firstly theoretically evaluate and compare the computational cost. Secondly, we realize prototypes of the protocols and carry out two sets of experiment. The computational time is tested to observe the computational cost.

4.4.1 Evaluation

The computational cost is evaluated through the number of operations and algorithms on A and B. Denote the cost of computing a scalar multiplication by $\mathcal{S}$ and the cost of computing or verifying a MAC by $\mathcal{H}$. Normally, $\mathcal{S} > \mathcal{H}$. The computational cost is evaluated and compared in in Table 4.1. According to Table 4.1 we have the following conclusions:

- Conclusion 1: Protocol I-A reduces the computational cost on A compared with the IEEE PW protocol;

- Conclusion 2: Protocol I-B reduces the computational cost on B compared with the IEEE PW protocol;

- Conclusion 3: When A is a computationally limited device and B is much powerful than A, the overall performance of Protocol I-A is better than that of the IEEE PW protocol since it lets the powerful side undertake computational tasks on behalf of the limited one. Similarly, when B is a limited device and A is a powerful one, the overall performance of Protocol I-B is better than that of the IEEE PW protocol.

4.4.2 Experiments

We realize prototypes of Protocol I-A, I-B and the IEEE PW protocol using Python programming language. The MAC algorithm is realized through HMAC with SHA-256. The communication is realized through socket programming with TCP. Two sets of experiment are carried out. In Experiment I-1, in order to observe how much computational cost that Protocol I-A and I-B reduce on the initiator and the responder respectively, we use two virtual machines with the same configuration to execute the protocols. In Experiment I-2, in order to simulate two parties with different computational powers, we use a Raspberry Pi and a laptop to execute the protocols.

Experiment I-1

The initiator A and the responder B are deployed on two virtual machines with the same configuration (Table 4.2). We firstly run Protocol I-A and I-B with five elliptic curves P-192, P-224, P-256, P-384 and P-521 for ten times. The average computational time is illustrated in Figure 4.3 and 4.4. Secondly, we run Protocol I-A, I-B and the IEEE PW protocol with the elliptic curve P-256 (the curve is recommended in IEEE 802.15.6) for ten times. The average computational time is illustrated in Figure 4.5.

Table 4.2: Experimental environment of Experiment I-1.

Party	Operating System	Base Memory	Storage
A	Ubuntu 16.04.3 (32-bit)	1024 MB	10 GB
B	Ubuntu 16.04.3 (32-bit)	1024 MB	10 GB

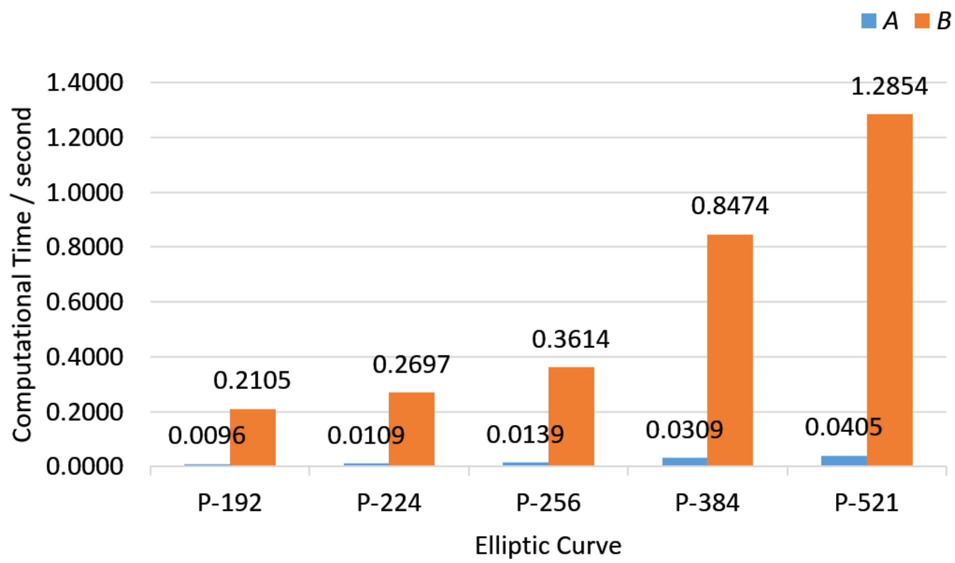

Figure 4.3: Average computational time on A and B of Protocol I-A in Experiment I-1.

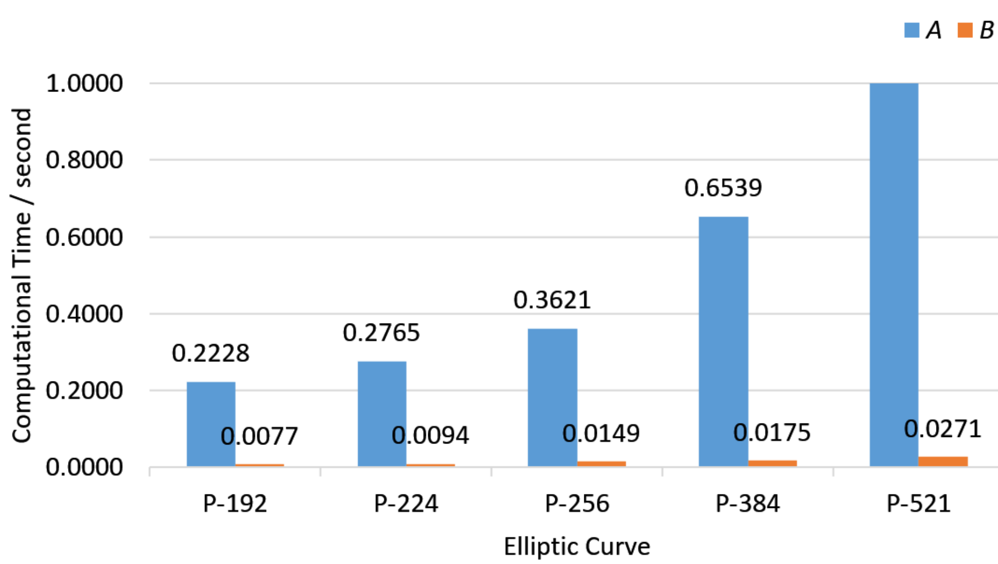

Figure 4.4: Average computational time on A and B of Protocol I-B in Experiment I-1.

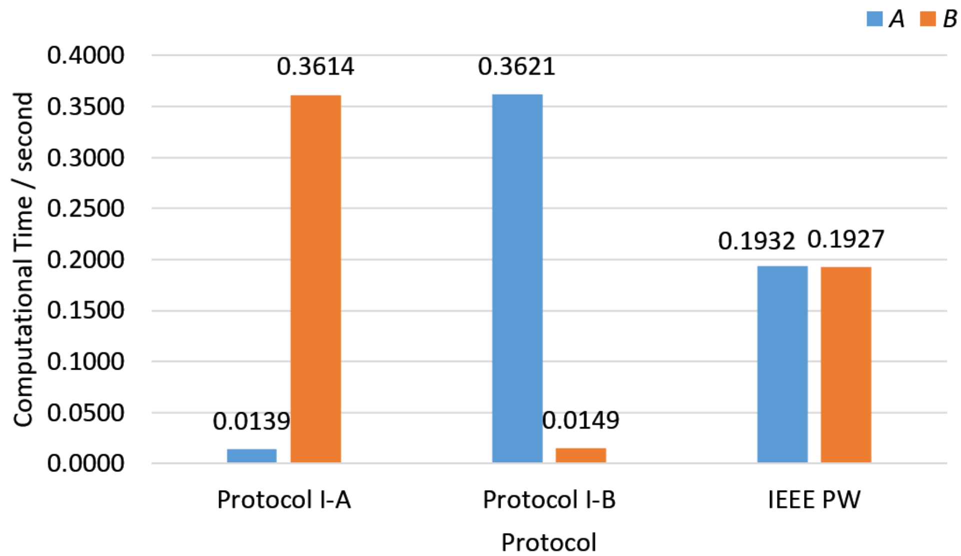

Figure 4.5: Average computational time on A and B of Protocol I-A, I-B and IEEE PW in Experiment I-1.

Figure 4.3 and 4.4 show that for all of the five curves, Protocol I-A has significantly reduced computational time on A; and Protocol I-B has significantly reduced computational time on B. According to Figure 4.5, the average computational time on A of Protocol I-A is less than that of the IEEE PW prtocol; and the average computational time on B of Protocol I-B is less than that of the IEEE PW protocol. This corresponds to the first two conclusions in Section 4.4.1.

Experiment I-2

In Experiment I-2, we use a Raspberry Pi to simulate the computationally limited device, and a laptop to simulate its powerful communicating partner. For Protocol I-A, we deploy the initiator A on the Raspberry Pi and the responder B on the laptop. For Protocol I-B, we deploy the initiator A on the laptop and the responder B on the Raspberry Pi. For the IEEE PW protocol, the initiator is deployed on the Raspberry Pi and the responder is deployed on the laptop. The experimental hardware platform is illustrated in Figure 4.6. Details about the Raspberry Pi and the laptop are listed in Table 4.3.

We run Protocol I-A, I-B and the IEEE PW protocol with the elliptic curves P-256 ten

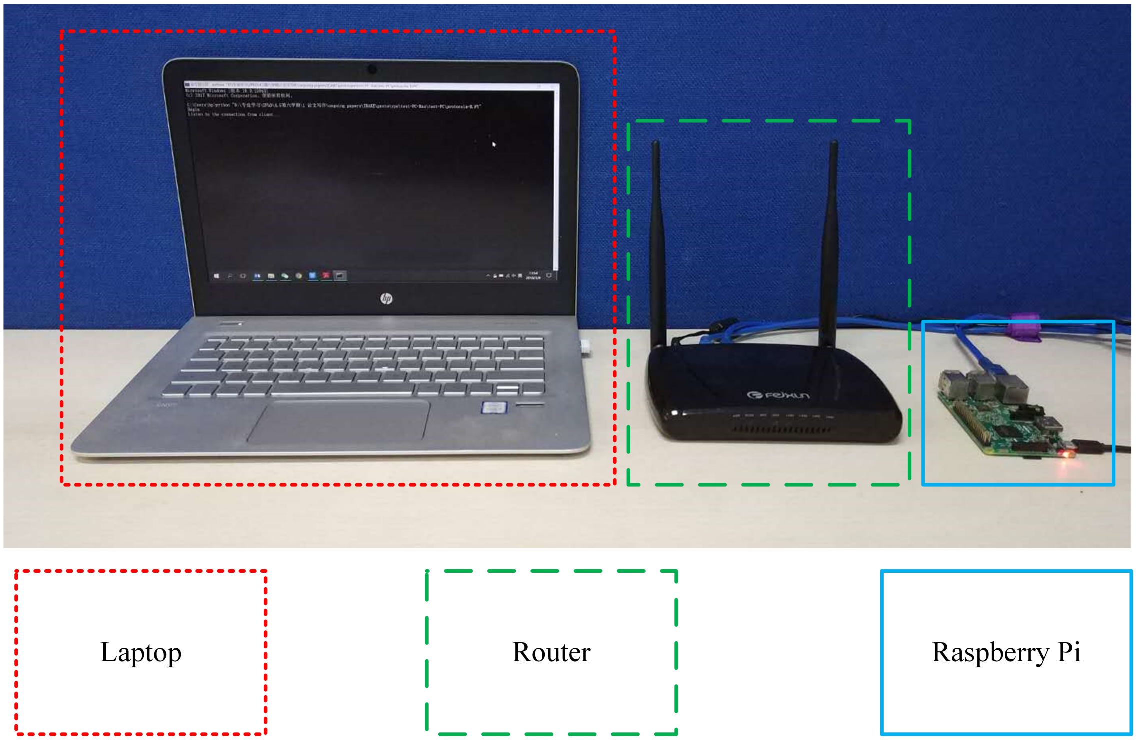

Figure 4.6: Hardware platform of Experiment I-2.

Table 4.3: Experimental environment of Experiment I-2.

Experimental Device	CPU	Memory	Hard Disk
Raspberry Pi	1.2 GMHz ARM	1 GB	32 GB
laptop	2.40 GHz i5-6200U	4 GB	120 GB

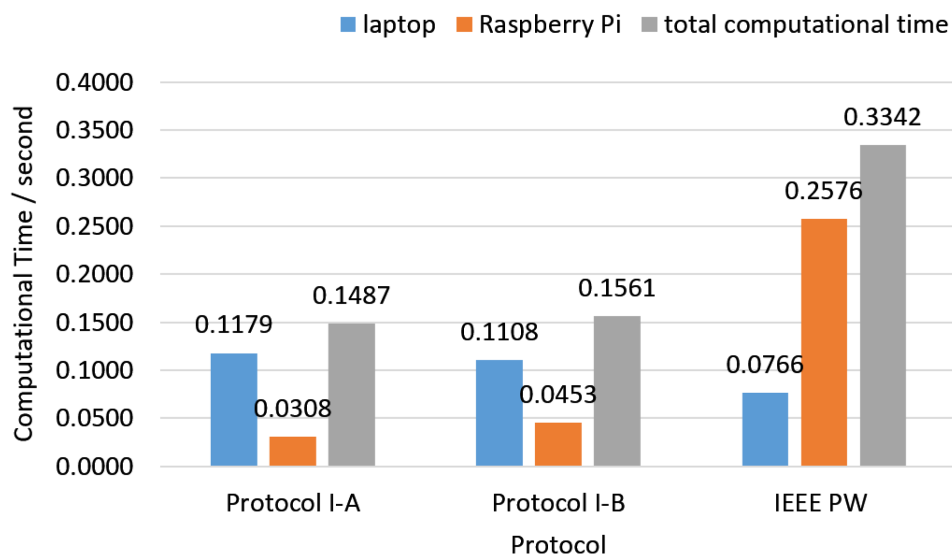

Figure 4.7: Average computational time on A and B of Protocol I-A, I-B and IEEE PW in Experiment I-2.

times. The average computational time is illustrated in Figure 4.7.

According to the Figure 4.7, the overall computational time of Protocol I-A and I-B are less than that of IEEE PW. The experimental results corresponds to the third conclusion in Section 4.4.1.

4.5 Chapter Summary

In this chapter, we presented two password UECDH-based AKE protocols. The two protocols achieve authentication through password and the MAC algorithm. The security of the protocols was analyzed; and the resistance to the man-in-the-middle attack and the impersonation attack was analyzed. To observe the performance of the two protocols, the IEEE PW protocol in IEEE 802.15.6 was set as the benchmark. The performance of the two protocols and the IEEE PW protocol was studied both through theoretical evaluation and through two sets of experiment. The results show that the two password UECDH-based AKE protocols reduce the computational time on the computationally limited device. They are more suitable than the IEEE PW protocol in securing communications between two devices with different computation powers.

The password UECDH-based AKE protocols require the public key of the party undertaking more computations to be hidden; otherwise, attackers can use the public key to impersonate this party. This can lead to security issues in some cases. One example is that the initiator (or the responder) needs to communicate with other parties in addition to the responder (or the initiator). In this case, the other parties can acquire the public key of the initiator (or responder); and the public key is no longer hidden (be only known by two parties); therefore, security issues can occur. To overcome these limitations, in the following chapter we will present two public key authenticated UECDH-based AKE protocols that do not rely on hiding the public key.

Chapter 5

Public Key Authenticated UECDH-based AKE Protocols

In Chapter 4 we explored how to remove man-in-the-middle and impersonation attacks by hiding the public key and using password and MAC algorithms. The MAC algorithm is used to authenticate the exchanged messages. The password is used to authenticate the communicating parities and hide the public keys. The purpose of hiding the public key is to prevent third parties from launching impersonation attacks. In cryptography, a more conventional method for authentication is digital signature. It has wide application in entity authentication and key establishment schemes and standards such as the Secure/Multipurpose Internet Mail Extensions (S/MIME) standard, the Transport Layer Security (TLS) standard and so on.

In this chapter, we present two public key authenticated[1] UECDH-based AKE protocols. By attaching the digital signatures to the exchanged messages, the two protocols address the security issues of UECDH discussed in Chapter 3. Firstly, we provide an overview of the communication model, attack model and security model of the protocols. Secondly, we present the two public key authenticated UECDH-based AKE protocols:

[1]The word "authenticated" here means that the two communication parties possess the authenticate public key of each other. This can be easily realized through Public Key Infrastructure (PKI); therefore, we do not include this procedure in our protocols.

Protocol II-A which requires less scalar multiplications on A than on B, and Protocol II-B which requires less scalar multiplication on B than A. Thirdly, we analyze the security of the protocols according to the attack model and security model, and illustrate how the man-in-the-middle and impersonation attacks to the protocols fail. At last, we study the performance of the two protocols through theoretical evaluation and experimental test.

5.1 Overview

5.1.1 Communication Model

The communication model of a public key authenticated UECDH-based AKE protocol is specified as follows.

- Participants. In each session of the protocol, there are two participants. The participants are denoted by their identities A and B. A is the initiator, and B is the responder. In particular, A and B have significantly different computation powers.
- Channels. The channels between A and B are normal channels.

5.1.2 Attack Model

We define the attack model of a public key authenticated UECDH-based AKE protocol through the following assumptions and ability specifications:

- Basic assumption 1. The attacker is unable to break the digital signature algorithms.
- Basic assumption 2. The attacker is unable to break the MAC algorithms.
- Basic ability. The attacker is able to observe all messages, alter messages, insert new messages, delay messages or delete messages transmitted via normal channels between A and B.
- Stronger ability 1. The attacker is able to obtain any previous session key.
- Stronger ability 2. The attacker is able to compromise the long-term secret keys of A and/or B.

5.1.3 Security Model

Under the above attack model, a public key authenticated UECDH-based AKE protocol desires to achieve the following security goals:

- Key authentication under the attack model that the attacker has the basic ability.
- Key confidentiality under the attack model that the attacker has the basic ability.
- Key integrity under the attack model that the attacker has the basic ability.
- Key confirmation under the attack model that the attacker has the basic ability.
- Known-key security (key freshness) under the attack model that the attacker has the basic ability and the stronger ability 1.
- Forward secrecy under the attack model that the attacker has the basic ability and the stronger ability 2.

5.2 Protocol Description

5.2.1 Protocol II-A

Protocol II-A generates a shared secret for a computationally limited initiator A and a more powerful responder B. It transfers one scalar multiplication from A to B. The protocol is described through the following procedures: initialization, key exchange and session keys computation. It is also illustrated in Figure 5.1.

Initialization

Before the execution of Protocol II-A, the initiator and the responder should obtain their private and public keys respectively. The private keys should be integers in the same finite field. The public keys should be points on the same elliptic curve. In addition, the initiator and the responder should possess the public key of each other.

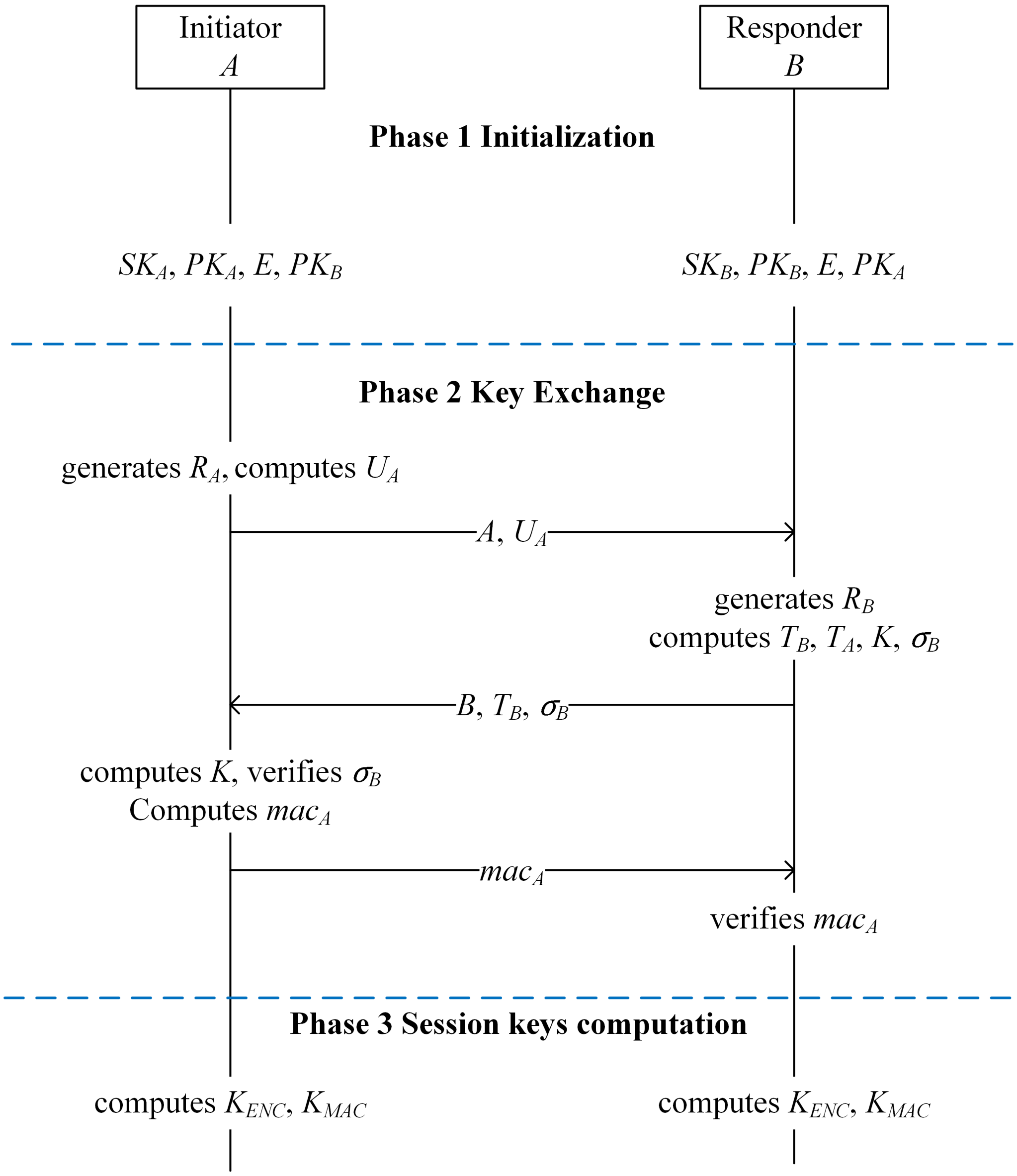

Figure 5.1: Protocol II-A.

Denote the initiator by A, the responder by B, the finite field by Z_q^*, the elliptic curve by E, the base point of E by G, the private and public keys of A by SK_A and PK_A, and the private and public keys of B by SK_B and PK_B. Formally, the initialization procedure generates the following values:

- Common parameters shared by A and B: $comm = (Z_q^*, E, G, PK_A, PK_B)$.
- Secret information held only by A: SK_A.
- Secret information held only by B: SK_B.

Key Exchange

1. A generates a random value $R_A \in Z_q^*$, and computes

$$U_A = R_A + SK_A.$$

Then A sends B M_1 as follows:

$$A \rightarrow B : M_1 = (A, U_A).$$

2. Upon receiving M_1, B firstly generates a random value $R_B \in Z_q^*$ and computes T_B through the following two equations:

$$U_B = R_B + SK_B,$$

$$T_B = U_B \times G.$$

Secondly, B computes the shared secret K_B as follows:

$$T_A = U_A \times G,$$

$$K_B = R_B \times (T_A - PK_A),$$

Thirdly, B computes a digital signature σ_B as follows:

$$\sigma_B = \mathrm{SIGN}(SK_B, B \| T_B \| K_{Bx}).$$

Finally, B sends A M_2 as follows:

$$B \to A : M_2 = (B, T_B, \sigma_B).$$

3. Upon receiving M_2, A firstly computes the shared secret K_A as follows:

$$K_A = R_A \times (T_B - PK_B).$$

Secondly, A verifies σ_B as follows:

$$\mathrm{VERY}(PK_B, B \| T_B \| K_{Ax}, \sigma_B) = \begin{cases} 1, & valid \\ 0, & invalid \end{cases}$$

Thirdly, if σ_B is valid, A computes a message authentication code mac_A as follows:

$$mac_A = \mathrm{MAC}(K_{Ax}, A \| U_A).$$

Finally, A sends B M_3 as follows:

$$A \to B : M_3 = mac_A$$

4. Upon receiving mac_A, B verifies mac_A as follows:

$$\mathrm{VER}(K_{Bx}, A \| U_A, mac_A) = \begin{cases} 1, & valid \\ 0, & invalid \end{cases}$$

Session Keys Computation

If σ_B is valid, A derives the session keys from K_{Ay} as follows:

$$K_{ENC} = \mathrm{F}(K_{Ay}, 1),$$

$$K_{MAC} = \mathrm{F}(K_{Ay}, 2).$$

If mac_A is valid, B derives the session keys from K_{By} as follows:

$$K_{ENC} = \mathrm{F}(K_{By}, 1),$$

$$K_{MAC} = \mathrm{F}(K_{By}, 2).$$

5.2.2 Protocol II-B

Protocol II-B generates a shared secret for a powerful initiator A and a computationally limited responder B. It transfers one scalar multiplication from B to A. We describe the protocol through the following procedures: initialization, key exchange and session keys computation in detail. In addition, we also illustrate the procedures in Figure 5.2.

Initialization

The initialization is the same with that of Protocol II-A. Let the notations A, B, Z_q^*, E, G,SK_A, PK_A, SK_B and PK_B be the same as we specified in Section 5.2.1. The initialization procedure produces the following values:

- Common parameters shared by A and B: $comm = (Z_q^*, E, G, PK_A, PK_B)$.
- Secret information held only by A: SK_A.
- Secret information held only by B: SK_B.

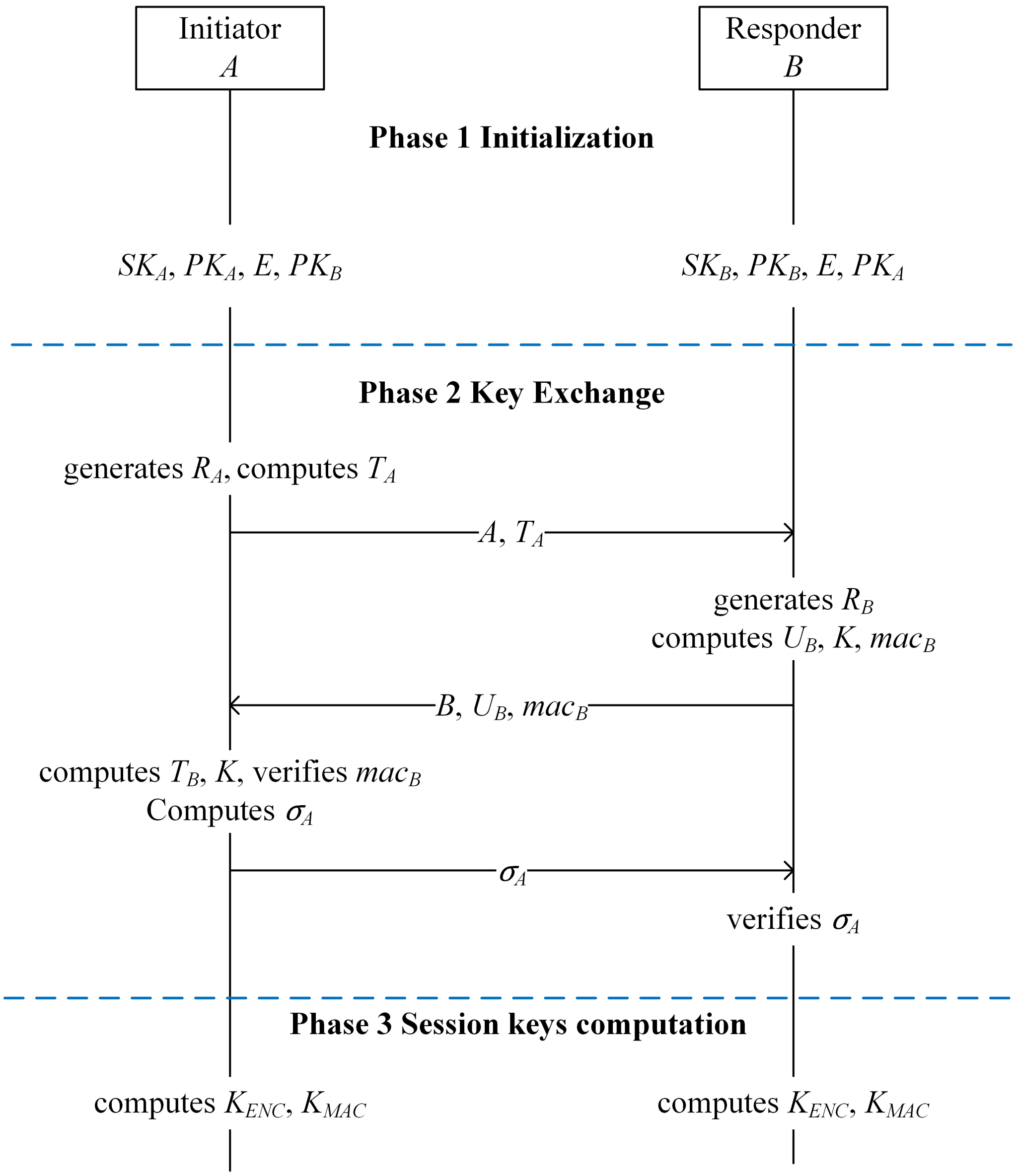

Figure 5.2: Protocol II-B.

Key Exchange

- A generates a random value $R_A \in Z_q^*$. A computes

$$U_A = R_A + SK_A,$$

$$T_A = U_A \times G.$$

Then A sends B M_1 as follows:

$$A \rightarrow B : M_1 = (A, T_A).$$

- Upon receiving M_1, B firstly generates a random value $R_B \in Z_q^*$, and computes

$$U_B = R_B + SK_B.$$

Secondly, B computes the shared secret K_B as follows:

$$K_B = R_B \times (T_A - PK_A),$$

Thirdly, B computes a message authentication code mac_B as follows:

$$mac_B = \mathrm{MAC}(K_{Bx}, B \| T_B).$$

Finally, B sends A M_2 as follows:

$$B \rightarrow A : M_2 = (B, U_B, mac_B).$$

- Upon receiving M_2, A firstly computes the shared secret K_A as follows:

$$T_B = U_B \times G,$$

$$K_A = R_A \times (T_B - PK_B).$$

Secondly, A verifies mac_B as follows:

$$\mathrm{VER}(K_{Ax}, B \| T_B, mac_B) = \begin{cases} 1, & valid \\ 0, & invalid \end{cases}$$

Thirdly, if mac_B is valid, A computes a signature σ_A as follows:

$$\sigma_A = \mathrm{SIGN}(SK_A, K_{Ax} \| A \| T_A).$$

Finally, A sends B M_3 as follows:

$$A \to B : M_3 = \sigma_A$$

- Upon receiving M_3, B verifies σ_A as follows:

$$\mathrm{VERY}(PK_A, K_{Bx} \| A \| T_A, \sigma_A) = \begin{cases} 1, & valid \\ 0, & invalid \end{cases}$$

Session Keys Computation

If mac_B is valid, A derives the session keys from K_{Ay} as follows:

$$K_{ENC} = \mathrm{F}(K_{Ay}, 1),$$

$$K_{MAC} = \mathrm{F}(K_{Ay}, 2).$$

If σ_A is valid, B derives the session keys from K_{By} as follows:

$$K_{ENC} = \mathrm{F}(K_{By}, 1),$$

$$K_{MAC} = \mathrm{F}(K_{By}, 2).$$

5.3 Security

In this section we illustrate that the two public key authenticated UECDH-based AKE protocols achieve the security goals (Section 5.1.3) under the attack model (Section 5.1.2). For each security goal, we provide a proposition that states a security feature, and prove how the proposition stands. In addition, we also show how the two protocols resist the man-in-the-middle attack and the impersonation attack.

5.3.1 Security Features

Proposition 5.1 (Key authentication of Protocol II-A and II-B)**.** *Assume there is an attacker C who can observe all messages, alter messages, insert new messages, delay messages or delete messages transmitted via normal channels between A and B. After a completed run of Protocol II-A (or II-B), A (or B) believes that he (or she) shares a secret with B (or A) other than any other party.*

Proof. (1) A completed run of Protocol II-A is defined by the validation of mac_A and δ_B. The validation of mac_A guarantees the authenticity of U_A; and the validation of δ_B guarantees the authenticity of T_B.

(2) A computes the shared secret from $K_A = R_A \times (T_B - PK_B)$. R_A is secretly generated by A; PK_B is pre-stored by A before the protocol; and T_B is authenticated according to (1). Therefore, after a completed run of Protocol II-A, A believes that he (or she) shares a secret with B other than any other party.

(3) B computes the shared secret from $K_B = R_B \times (U_A \times G - PK_A)$. R_B is secretly generated by B; PK_A is pre-stored by B before the protocol; and U_A is authenticated according to (1). Therefore, after a completed run of Protocol II-A, B believes that he (or she) shares a secret with A other than any other party.

According to (2) and (3), we have the conclusion that Protocol II-A provides key authentication. Similarly we can prove that Protocol II-B provides key authentication. □

Proposition 5.2 (Key confidentiality of Protocol II-A and II-B)**.** *Assume there is an* *attacker C who can observe all messages, alter messages, insert new messages, delay messages or delete messages transmitted via normal channels between A and B. After a completed run of Protocol II-A (or II-B), the attacker is unable to derive the shared key of A and B.*

Proof. (1) The shared secret can be computed from any of the following equations:

$$K_A = R_A \times (T_B - PK_B),$$

$$K_B = R_B \times (U_A \times G - PK_A),$$

$$K = R_A \times R_B \times G.$$

Therefore, R_A or R_B is required to compute the shared secret.

(2) R_A is hidden by the following equation:

$$U_A = R_A + SK_A.$$

Therefore, SK_A is required to compute R_A.

R_B is hidden by the following equation:

$$T_B = (R_B + SK_B) \times G.$$

Therefore, SK_B is required to compute R_B.

(3) According to the attack model, C has neither SK_A nor SK_B. C is unable to compute R_A or R_B. Therefore, C is unable to compute $K_A = K_B = K$.

Therefore, we have the conclusion that Protocol II-A provides key confidentiality. Similarly we can prove that Protocol II-B provides key confidentiality. □

Proposition 5.3 (Key integrity of Protocol II-A and II-B). *Assume there is an attacker C who can observe all messages, alter messages, insert new messages, delay messages or delete messages transmitted via normal channels between A and B. After a completed run* *of Protocol II-A (or II-B), A and B computes the equal secret.*

Proof. (1) As we proved in Theorem 5.1, a completed run of Protocol II-A implies the authenticity of U_A and T_B.

(2) The secret K_A is computed by A from

$$\begin{aligned} K_A &= R_A \times (T_B - PK_B) \\ &= R_A \times (U_B \times G - PK_B) \\ &= R_A \times ((R_B + SK_B) \times G - PK_B) \\ &= R_A \times R_B \times G \end{aligned}$$

The secret K_B is computed by B from

$$\begin{aligned} K_B &= R_B \times (U_A \times G - PK_A) \\ &= R_B \times ((R_A + SK_A) \times G - PK_A) \\ &= R_B R_A \times G \\ &= R_A \times R_B \times G \\ &= K_B. \end{aligned}$$

Therefore, after a completed run of Protocol II-A, A and B compute the equal secret. We have the conclusion that Protocol II-A provides key integrity. Similarly we can prove that Protocol II-B provides key integrity. □

Proposition 5.4 (Key confirmation of Protocol II-A and II-B)**.** *Assume there is an attacker C who can observe all messages, alter messages, insert new messages, delay messages or delete messages transmitted via normal channels between A and B. After a completed run of Protocol II-A (or II-B), both A and B have receive evidence confirming that the other party knows the secret.*

Proof. (1) As we proved in Proposition 5.1, a completed run of Protocol II-A is defined the validation of mac_A and δ_B. That is, A has received and validated δ_B; and B has received and validated mac_B.

(2) δ_B is generated by B and takes the shared secret as part of the inputs. It is the evidence confirming that B knows the secret.

(3) mac_A is generated by A and take the shared secret as part of the inputs. It is the evidence confirming that A knows the secret.

According to (1) and (2), after a completed run of Protocol II-A, A has received evidence confirming that B knows the secret. According to (1) and (3), after a completed run of Protocol II-A, B has received evidence confirming that A knows the secret. Therefore, we have the conclusion that Protocol II-A provides key confirmation. Similarly we can prove that Protocol II-B provides key confirmation. □

Proposition 5.5 (Known-key security (key freshness) of Protocol II-A and II-B). *Assume there is an attacker C who can observe all messages, alter messages, insert new messages, delay messages or delete messages transmitted via normal channels between A and B. In addition, C is able to obtain any previous session keys. After a completed run of Protocol II-A (or II-B), C is unable to derive the shared secret from the previous session keys.*

Proof. In Protocol II-A, the computation of the secret takes the R_A and R_B as the inputs. Since R_A and R_B are random values generated by A and B respectively in the key exchange procedure, in each run of Protocol II-A the values are unique. Therefore, the secret is fresh in each run of the protocol. That is, Protocol II-A provides known-key security (key freshness). Similarly we can prove that Protocol II-B provides known-key security. □

Proposition 5.6 (Forward secrecy of Protocol II-A and II-B). *Assume there is an attacker C who can observe all messages, alter messages, insert new messages, delay messages or delete messages transmitted via normal channels between A and B. In addition, C compromises the long-term secrets (i.e. the private keys) of A and B. C is unable to derive the previous session keys of Protocol II-A or Protocol II-B.*

Proof. As we proved in Theorem 5.2, R_A or R_B is required to compute the shared secret. For a previous run of the protocol, the values of R_A and R_B are short-term secrets that are unknown to C. Therefore, C is unable to compute the secret of the previous run of the protocol.

Therefore, we have the conclusion that Protocol II-A provides forward secrecy. Similarly we can prove that Protocol II-B provides forward secrecy. □

5.3.2 Resistance to Attacks

The public key authentication UECDH-based AKE protocols address the vulnerabilities to the man-in-the-middle attack and the impersonation attack. Bellow we illustrate how the two attacks fail.

Resistance to the man-in-the-middle attack

Assume C is a man-in-the-middle attacker to Protocol II-A between A and B. To launch the attack, C interacts with A and B as follows.

1. A generates a random value $R_A \in Z_q^*$ and computes

$$U_A = R_A + SK_A.$$

   Then A sends B M_1 as follows:

$$A \to B : M_1 = (A, U_A).$$

2. C firstly intercepts M_1.

   Secondly, C generates a random value $R_C \in Z_q^*$ and computes

$$U_C = R_C + SK_C.$$

   At last, C sends a forged message (A, U_C) to B, i.e.,

$$C \to B : M_1' = (A, U_C).$$

3. Upon receiving the forge message M_1', B firstly generates a random value $R_B \in Z_q^*$, and computes

$$U_B = R_B + SK_B,$$

$$T_B = U_B \times G.$$

Secondly, B computes the shared secret K_B as follows:

$$T_A' = U_C \times G,$$

$$K_B' = R_B \times (T_A' - PK_A).$$

Thirdly, B computes a digital signature σ_B as follows:

$$\sigma_B = \mathrm{SIGN}(SK_{Bx}, B \| T_B \| K_{Bx}).$$

Then B sends A with M_2 as follows:

$$B \to A : M_2 = (B, T_B, \sigma_B)$$

4. C firstly intercepts M_2.

Secondly, C computes

$$T_C = U_C \times G.$$

Thirdly, C computes the shared secrets K_{CB} and K_{CA} as follows:

$$K_{CB} = R_C \times (T_B - PK_B),$$

$$K_{CA} = R_C \times (U_A \times G - PK_A).$$

- C is unable to forge a valid digital signature of B according to the basic assumption 1.

– C is unable to compute equal keys with B or A since $K_{CB} \neq K_B$ and $K_{CA} \neq K_A$.

Without the valid mac_A and σ_B, both B and A terminate the protocol. The attack fails.

Therefore, Protocol II-A is resistant to the man-in-the-middle attack. Similarly, Protocol II-B is resistant to the man-in-the-middle attack.

Resistance to the impersonation attack

Assume C is an impersonation attacker to Protocol II-A between A and B. To launch the attack, C impersonates B and interacts with A as follows.

1. A generates a random value $R_A \in Z_q^*$ and computes

$$U_A = R_A + SK_A.$$

Then A sends B M_1 as follows:

$$A \rightarrow B : M_1 = (A, U_A).$$

2. C firstly intercepts and blocks M_1.

Secondly, C generates a random value $R_C \in Z_q^*$, and computes

$$T_C = R_C \times G + PK_B.$$

Thirdly, C computes the shared secret K_{CA} as follows:

$$T_A = U_A \times G,$$

$$K'_{CA} = R_C \times (T_A - PK_A).$$

Table 5.1: Comparison of computational cost.

Computational cost	Cost on A	Cost on B
Protocol II-A	$\mathcal{H}+\mathcal{D}+\mathcal{S}$	$\mathcal{H}+\mathcal{D}+3\mathcal{S}$
Protocol II-B	$\mathcal{H}+\mathcal{D}+3\mathcal{S}$	$\mathcal{H}+\mathcal{D}+\mathcal{S}$
TLS PK Authenticated	$2\mathcal{D}+2\mathcal{S}$	$2\mathcal{D}+2\mathcal{S}$

At this stage, C fails since he (or she) is unable to forge a valid digital signature of B. Without the valid σ_B, A terminates the protocol. The attack fails.

Therefore, Protocol II-A is resistant to the impersonation attack. Similarly, Protocol II-B is resistant to the impersonation attack.

5.4 Performance

To study the performance of Protocol II-A and Protocol II-B, we choose the TLS PK Authenticated protocol as the benchmark. We firstly theoretically evaluate and compare the computational cost. Secondly, we realize prototypes of the protocols and carry out two sets of experiment. The computational time is tested to observe the computational cost.

5.4.1 Evaluation

Denote the cost of computing a scalar multiplication by $\mathcal{S}$, the cost of computing or verifying a MAC by $\mathcal{H}$, the cost of computing or verifying a digital signature by $\mathcal{D}$. The computational cost is evaluated and compared in Table 5.1.

According to Table 5.1 we have the following conclusions:

- Conclusion 1: Protocol II-A reduces the computational cost on A compared with the TLS PK Authenticated protocol;

- Conclusion 2: Protocol II-B reduces the computational cost on B compared with the TLS PK Authenticated protocol;

- Conclusion 3: When A is a computationally limited device, and B is much powerful than A, the overall performance of Protocol II-A is better than that of the TLS PK Authenticated protocol since it lets the powerful side undertake computational tasks on behalf of the limited one. Similarly, when B is a limited device and A is a powerful one, the overall performance of Protocol II-B is better than that of the TLS PK Authenticated protocol.

5.4.2 Experiments

We realize prototypes of Protocol II-A, Protocol II-B and the TLS PK Authenticated protocol using Python programming language. The MAC algorithm is realized through HMAC with SHA-256. The digital signature is realized through ECDSA. The communication is realized through socket programming with TCP. Two sets of experiment are carried out. In Experiment II-1, in order to observe how much computational cost that Protocol II-A and Protocol II-B reduce on the initiator and the responder respectively, we use two virtual machines with the same configuration to execute the protocols. In Experiment II-2, in order to simulate two parties with different computational powers, we use a Raspberry Pi and a laptop to execute the protocols.

Experiment II-1

The initiator A and the responder B are deployed on two virtual machines with the same configuration (Table 5.2). We firstly run Protocol II-A and II-B with five elliptic curves P-192, P-224, P-256, P-384 and P-521 for ten times. The average computational time is illustrated in Figure 5.3 and 5.4. Secondly, we run Protocol II-A, II-B and the TLS PK Authenticated protocol with the elliptic curve P-256 and P-384 (recommended in TLS 3.0) for ten times. The average computational time is illustrated in Figure 5.5.

Figure 5.3 and 5.4 show that for all of the five curves, Protocol II-A has less computational time on A than on B; and Protocol II-B has less computational time on B than on A. According to Figure 5.5, the average computational time on A of Protocol II-A is less than that of the TLS PK Authenticated protocol; and the average computational

Table 5.2: Experimental environment of Experiment II-1.

Party	Operating System	Base Memory	Storage
A	Ubuntu 16.04.3 (32-bit)	1024 MB	10 GB
B	Ubuntu 16.04.3 (32-bit)	1024 MB	10 GB

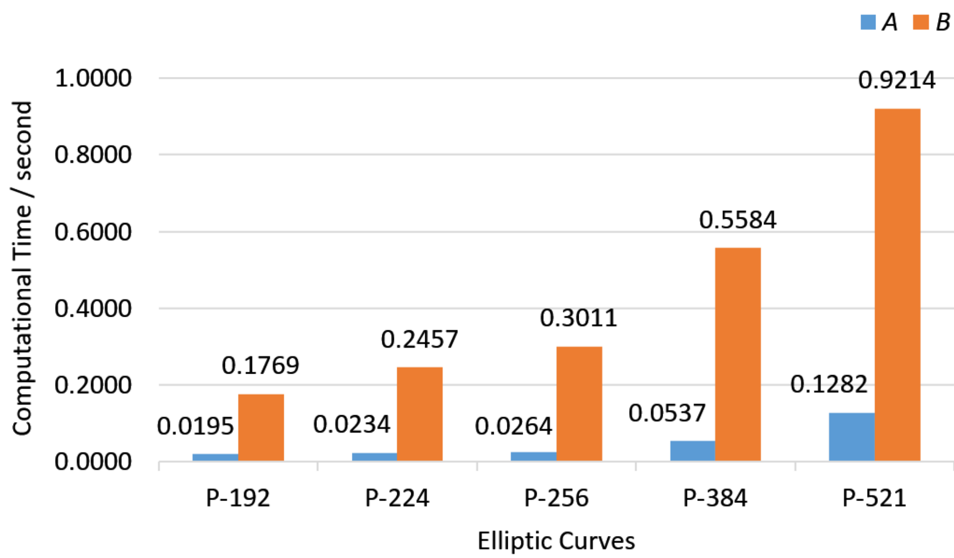

Figure 5.3: Average computational time on A and B of Protocol II-A in Experiment II-1.

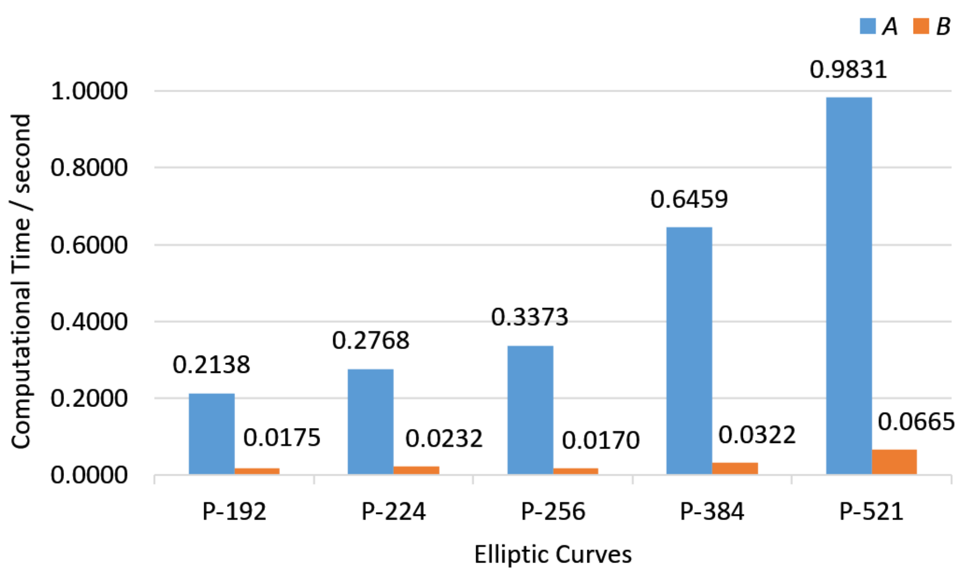

Figure 5.4: Average computational time on A and B of Protocol II-B in Experiment II-1.

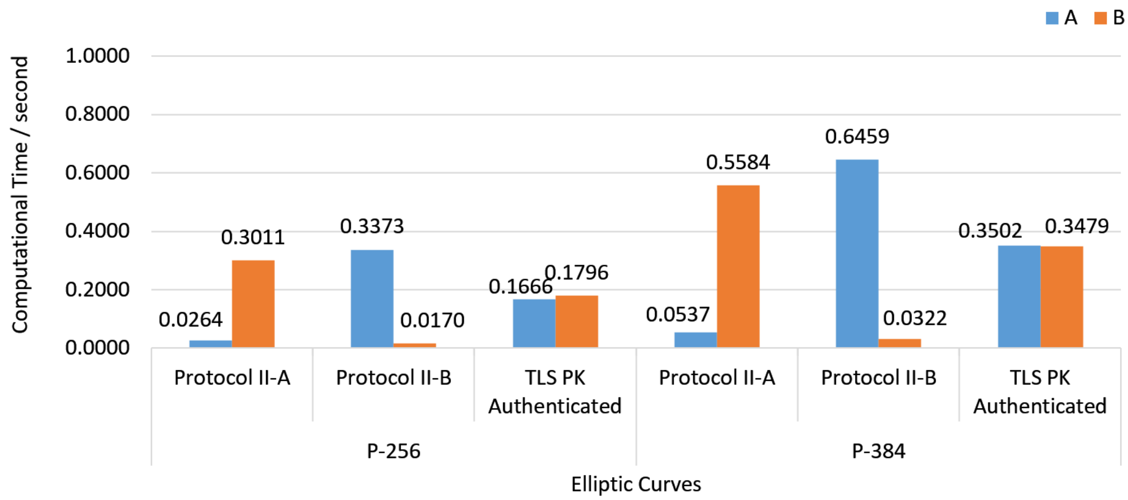

Figure 5.5: Average computational time of A and B of Protocol II-A, Protocol II-B and TLS PK Authenticated in Experiment II-1.

Table 5.3: Experimental environment of Experiment II-2.

Experimental Device	CPU	Memory	Hard Disk
Raspberry Pi	1.2 GMHz ARM	1 GB	32 GB
laptop	2.40 GHz i5-6200U	4 GB	120 GB

time on B of Protocol II-B is less than that of the TLS PK Authenticated protocol. It is corresponding to the first two conclusions in Section 5.4.1.

Experiment II-2

In Experiment II-2, we use a Raspberry Pi as the computationally limited device, and a laptop as its powerful communicating partner. For Protocol II-A, we deploy the initiator A on the Raspberry Pi and the responder B on the laptop. For Protocol II-B, we deploy the initiator A on the laptop and the responder B on the Raspberry Pi. Details about the Raspberry Pi and the laptop are listed in Table 5.3.

We run Protocol II-A, II-B and the TLS PK Authenticated protocol with the elliptic curve P-256 for ten times. The average computational time is illustrated in Figure 5.6.

According to Figure 5.6, Protocol II-A and II-B are more friendly to the limited device

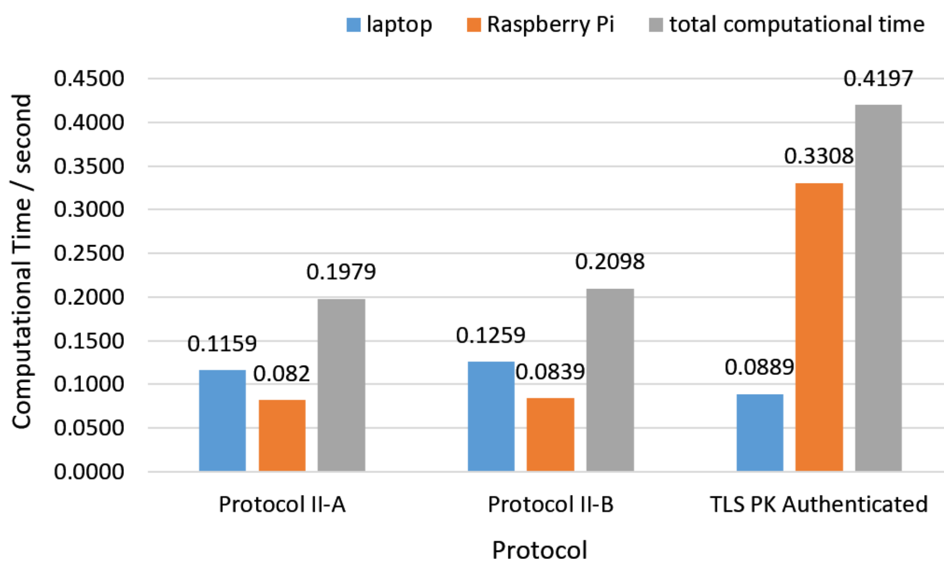

Figure 5.6: Average computational time on A and B of Protocol II-A, Protocol II-B and TLS PK Authenticated in Experiment II-2.

(Raspberry Pi); and the overall computational time of Protocol II-A and II-B are less than that of the TLS PK Authenticated protocol. The experimental results are corresponding to the third conclusion in Section 5.4.1.

5.5 Chapter Summary

This chapter presented two public key authenticated UECDH-based AKE protocols. Digital signature algorithms are used to achieve authentication. The protocols assume the communicating parties have the authenticated public key of each other. Security of the protocols was analyzed; and resistance to the man-in-the-middle and impersonation attacks was illustrated. Prototypes of the protocols and similar protocol in TLS were realized. Based on the prototypes, two sets of of experiment were carried out to observe the performance. The results show that the proposed protocols have successfully reduced the computational cost on the limited party; and have lower over computational time than similar protocol in TLS.

To use the digital signature algorithms, authenticated public keys between the communicating parties are required. This is often realized through Public Key Infrastructure

(PKI). In PKI, a trusted center named Certificate Authority (CA) issues public keys for the users. However, maintaining the public key certificates is complicated; and may overburden the limited devices. In the following chapter, we will introduce another authentication measure that does not rely on PKI.

Chapter 6

High Bandwidth OOB UECDH-based AKE Protocols

In previous chapters we illustrated two different ways to remove attacks and achieve authentication: 1) using a short pre-shared short secret, i.e., a password; and 2) issuing authenticated public keys through trusted parties. Each of them has an applicable scenario. However, there are scenarios where all the above methods are unsuitable. For example, it is infeasible for two unacquainted devices to have a pre-shared secret; and moreover, if the devices are computationally limited sensors, it is too expensive to apply PKI to issue and maintain authenticated public keys. In this and the following chapter, we introduce a different measure that neither requires pre-shared secret nor relies on PKI. That is, OOB channels. The OOB channel is used to transmit authenticated messages in security protocols since it is not vulnerable to attacks. Such protocols are popular in recent years. International standards such as Bluetooth 5.0 and IEEE 802.15.6 include OOB channel based security protocols.

In this chapter, we introduce UECDH-based AKE protocols with high bandwidth OOB channels that are capable of transmiting long strings. Examples of such channels are emails, QR codes, human body channels and so on. We firstly provide an overview of the protocols in terms of the communication model, attack model and security model. Secondly, we describe the protocols: Protocol III-A which requires less scalar multiplications on A than on B, and Protocol III-B which requires less scalar multiplications on B than A. Thirdly, we analyze the security of the protocols according to the attack and security models; in particular, we discuss how the protocols resist the man-in-the-middle attack and the impersonation attack. Finally, prototypes of the protocols are realized. We evaluate the performance of the protocols through theoretical evaluation and experiments.

6.1 Overview

6.1.1 Communication Model

The communication model of a high bandwidth OOB UECDH-based AKE protocol is specified as follows.

- Participants. In each session of the protocol, there are two participants. The participants are denoted by their identities A and B. A is the initiator, and B is the responder. In particular, A and B have significantly different computational capabilities.

- Channels. The channels between A and B include normal channels and high bandwidth OOB channels.

6.1.2 Attack Model

The following assumptions and attack model specify what an attacker to a high bandwidth OOB UECDH-based AKE protocol is able and unable to do.

- Basic assumption 1. The attacker is unable to alter, insert, delay or delete messages transmitted in the OOB channel.

- Basic assumption 2. The attacker is unable to break the MAC algorithms.

- Basic ability. The attacker is able to observe all messages, alter messages, insert new messages, delay messages or delete messages transmitted between A and B.

- Stronger ability 1. The attacker is able to obtain any previous session key.
- Stronger ability 2. The attacker is able to compromise the long-term secret keys of A and/or B.

6.1.3 Security Model

Under the above attack model, a high bandwidth OOB UECDH-based AKE protocol aims to achieve the following security goals:

- Key authentication under the attack model that the attacker has the basic ability.
- Key confidentiality under the attack model that the attacker has the basic ability.
- Key integrity under the attack model that the attacker has the basic ability.
- Key confirmation under the attack model that the attacker has the basic ability.
- Known-key security (key freshness) under the attack model that the attacker has the basic ability and the stronger ability 1.
- Forward secrecy under the attack model that the attacker has the basic ability and the stronger ability 2.

6.2 Protocol Description

6.2.1 Protocol III-A

Protocol III-A shares a secret between a computationally limited initiator A and a more powerful responder B. It transfers one scalar multiplication from A to B. The protocol is described through the following procedures: initialization, key exchange and session keys computation. It is also illustrated in Figure 6.1.

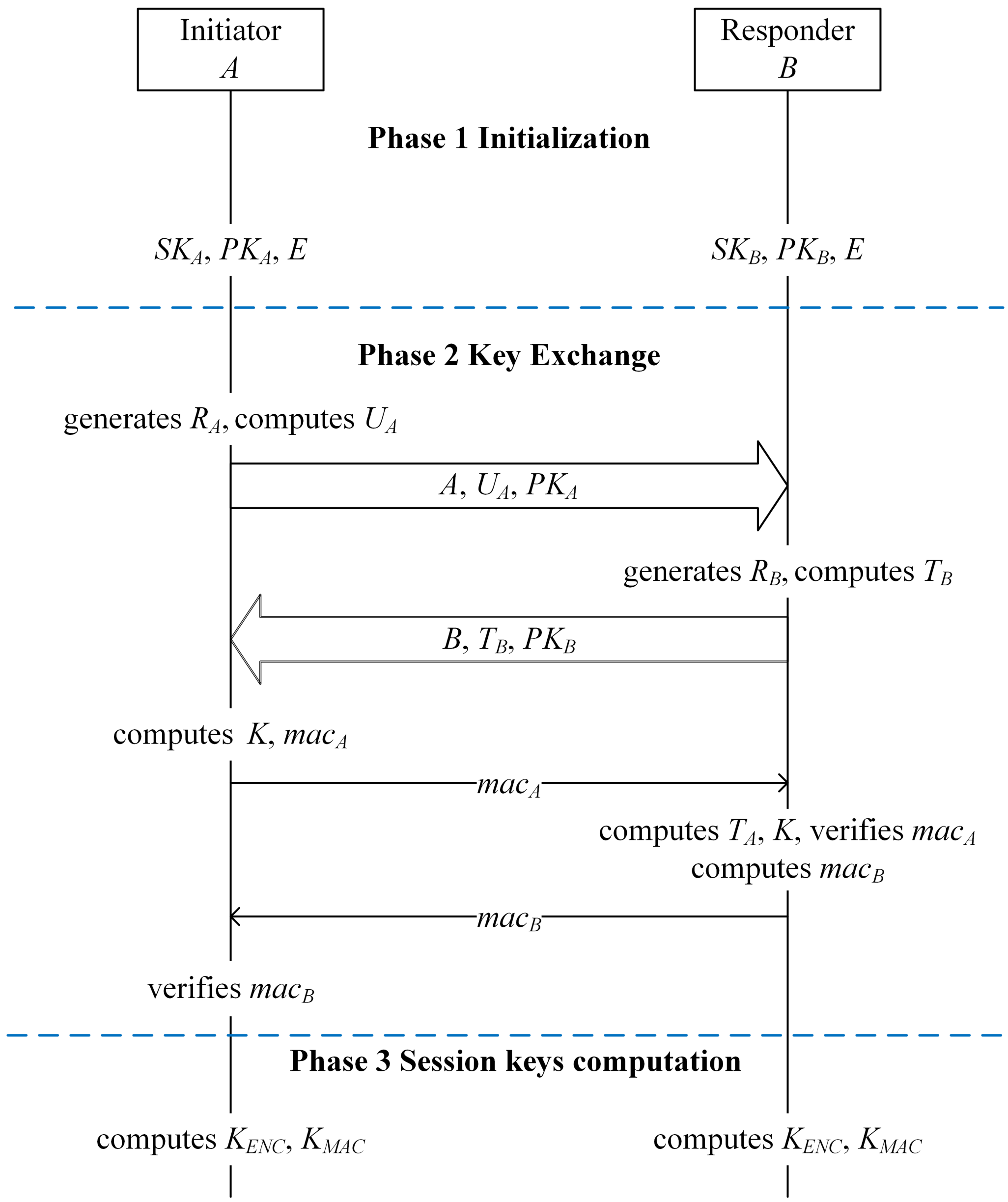

Figure 6.1: Protocol III-A.

Initialization

Before the execution of Protocol III-A, the initiator and the responder shall obtain their private and public keys respectively. The private keys should be integers in the same finite field. The public keys should be points on the same elliptic curve.

Denote the initiator by A, the responder by B, the finite field by Z_q^*, the elliptic curve by E, the base point of E by G, the private and public keys of A by SK_A and PK_A, and the private and public keys of B by SK_B and PK_B. Formally, the initialization procedure generates the following values:

- Common parameters shared by A and B: $comm = (Z_q^*, E, G)$.
- Information held only by A: SK_A and PK_A where SK_A should be securely stored.
- Information held only by B: SK_B, PK_B where SK_B should be securely stored.

Key Exchange

1. A generates a random value $R_A \in Z_q^*$, and computes

$$U_A = R_A + SK_A.$$

   Then A sends B M_1 through a high-bandwidth OOB channel as follows:

$$A \Rightarrow_h B : M_1 = (A, PK_A, U_A).$$

2. Upon receiving M_1, B firstly generates a random value $R_B \in Z_q^*$ and computes T_B through the following two equations:

$$U_B = R_B + SK_B,$$

$$T_B = U_B \times G.$$

Then B sends A with M_2 through a high-bandwidth OOB channel as follows:

$$B \Rightarrow_h B : M_2 = (B, PK_B, T_B).$$

3. Upon receiving M_2, A firstly computes the shared secret K_A as follows:

$$K_A = R_A \times (T_B - PK_B).$$

Secondly, A computes a message authentication code mac_A as follows:

$$mac_A = \mathrm{MAC}(K_{Ax}, A \| PK_A \| U_A).$$

Finally, A sends B M_3 through a normal channel as follows:

$$A \to B : M_3 = mac_A.$$

4. Upon receiving M_3, B firstly computes the shared secret K_B as follows:

$$T_A = U_A \times G,$$

$$K_B = R_B \times (T_A - PK_A).$$

Secondly, B verifies mac_A as follows:

$$\mathrm{VER}(K_{Bx}, A \| PK_A \| U_A, mac_A) = \begin{cases} 1, & valid \\ 0, & invalid \end{cases}$$

Thirdly, if mac_A is valid, B computes mac_B as follows:

$$mac_B = \mathrm{MAC}(K_{Bx}, B \| PK_B \| T_B).$$

Finally, B sends A M_4 through a normal channel as follows:

$$B \rightarrow A : M_4 = mac_B.$$

5. Upon receiving M_4, A verifies mac_B as follows:

$$\mathrm{VER}(K_{Ax}, B \| PK_B \| T_B, mac_B) = \begin{cases} 1, & valid \\ 0, & invalid \end{cases}$$

Session Keys Computation

If mac_B is valid, A derives the session keys from K_{Ay} as follows:

$$K_{ENC} = \mathrm{F}(K_{Ay}, 1),$$

$$K_{MAC} = \mathrm{F}(K_{Ay}, 2).$$

If mac_A is valid, B derives the session keys from K_{By} as follows:

$$K_{ENC} = \mathrm{F}(K_{By}, 1),$$

$$K_{MAC} = \mathrm{F}(K_{By}, 2).$$

6.2.2 Protocol III-B

Protocol III-B shares a secret between a powerful initiator A and a computationally limited responder B. It transfers one scalar multiplication from B to A. We describe the protocol through the following procedures: initialization, key exchange and session keys computation in detail. In addition, we also illustrate the procedures through Figure 6.2.

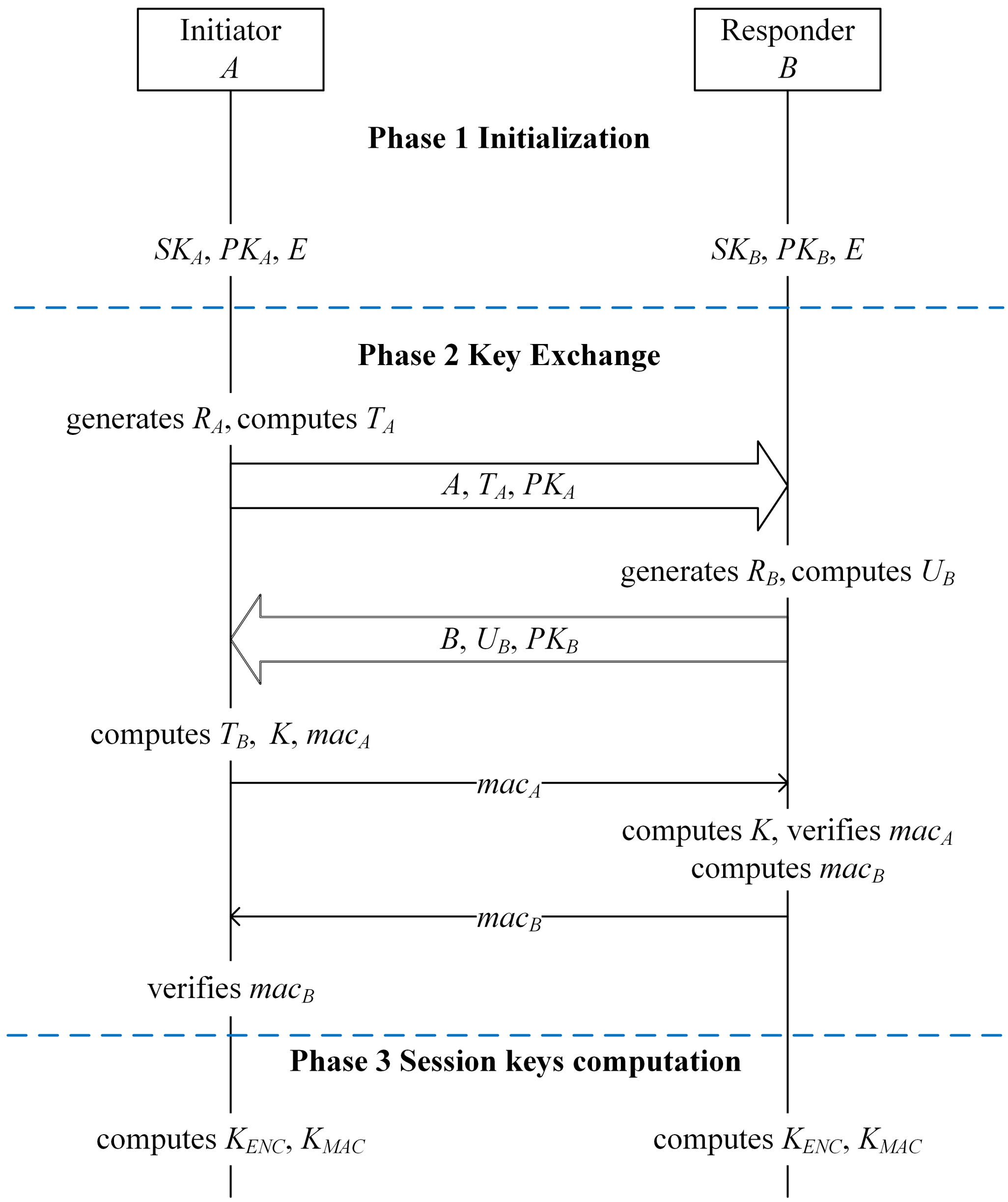

Figure 6.2: Protocol III-B.

Initialization

The initialization here is similar with that of Protocol III-A. Let the notations A, B, Z_q^*, E, G, SK_A, PK_A, SK_B and PK_B be the same as we specified in Section 6.2.1. The initialization procedure produces the following values:

- Common parameters shared by A and B: $comm = (Z_q^*, E, G)$.
- Information held only by A: SK_A and PK_A where SK_A is securely stored.
- Information held only by B: SK_B and PK_B where SK_B is securely stored.

Key Exchange

1. A generates a random value $R_A \in Z_q^*$ and computes

$$U_A = R_A + SK_A,$$

$$T_A = U_A \times G.$$

   Then A sends B M_1 through a high-bandwidth OOB channel as follows:

$$A \Rightarrow_h B : M_1 = (A, PK_A, T_A).$$

2. Upon receiving M_1, B firstly generates a random value $R_B \in Z_q^*$, and computes

$$U_B = R_B + SK_B.$$

   Then B sends A M_2 through a high-bandwidth OOB channel as follows:

$$B \Rightarrow_h: A : M_2 = (B, PK_B, U_B).$$

3. Upon receiving M_2, A firstly computes the shared secret K_A as follows:

$$T_B = U_B \times G,$$

$$K_A = R_A \times (T_B - PK_B).$$

Secondly, A computes a message authentication code mac_A as follows:

$$mac_A = \mathrm{MAC}(K_A x, A \| PK_A \| T_A).$$

Finally, A sends B M_3 through a normal channel as follows:

$$A \rightarrow B : M_3 = mac_A.$$

4. Upon receiving M_3, B firstly computes the shared secret K_B as follows:

$$K_B = R_B \times (T_A - PK_A).$$

Secondly, B verifies mac_A as follows:

$$\mathrm{VER}(K_{Bx}, A \| PK_A \| T_A, mac_A) = \begin{cases} 1, & valid \\ 0, & invalid \end{cases}$$

Thirdly, if mac_B is valid, B computes a message authentication code mac_B as follows:

$$mac_B = \mathrm{MAC}(K_{Bx}, B \| PK_B \| U_B).$$

Finally, B sends A M_4 through a normal channel as follows:

$$B \rightarrow A : M_4 = mac_B.$$

5. Upon receiving M_4, A verifies mac_B as follows:

$$\mathrm{VER}(K_{Ax}, B\|PK_B\|U_B, mac_B) = \begin{cases} 1, & valid \\ 0, & invalid \end{cases}$$

Session Keys Computation

If mac_B is valid, A derives the session keys from K_{Ay} as follows:

$$K_{ENC} = \mathrm{F}(K_{Ay}, 1),$$

$$K_{MAC} = \mathrm{F}(K_{Ay}, 2).$$

If mac_A is valid, B derives the session keys from K_{By} as follows:

$$K_{ENC} = \mathrm{F}(K_{By}, 1),$$

$$K_{MAC} = \mathrm{F}(K_{By}, 2).$$

6.3 Security

This section illustrates that the two high bandwidth OOB UECDH-based AKE protocols achieve the security goals (Section 6.1.3) under the attack model (Section 6.1.2). For each security goal, we provide a proposition that states a security feature, and prove how the proposition stands. In addition, we also show how the two protocols resist the man-in-the-middle attack and the impersonation attack.

6.3.1 Security Features

Proposition 6.1 (Key authentication of Protocol III-A and III-B)**.** *Assume there is an attacker C who can observe all messages, alter messages, insert new messages, delay messages or delete messages transmitted via the normal channels between A and B. After a* *completed run of Protocol III-A (or III-B), A (or B) believes that he (or she) shares a secret with B (or A) other than any other party.*

Proof. (1) According to the basic assumption 1, messages transmitted via the OOB channels are authenticated. Therefore, in Protocol III-A, PK_A, U_A, PK_B and T_B are authenticated.

(2) A computes the secret K_A from the following equation:

$$K_A = R_A \times (T_B - PK_B).$$

R_A is generated by A; and T_B and PK_B are authenticated according to (1). Therefore, A believes that he (or she) shares a secret with B.

(3) B computes the secret K_B from the following equation:

$$K_B = R_B \times (U_A \times G - PK_A).$$

R_B is generated by B; U_A and PK_A are authenticated according to (1). Therefore, B believes that he (or she) shares a secret with A.

According to (2) and (3), Protocol III-A provides key authentication under the basic assumptions and the attack model that C has the basic ability. Similarly, we can prove that Protocol III-B provides key authentication under the basic assumptions and the attack model that C has the basic ability. □

Proposition 6.2 (Key confidentiality of Protocol III-A and III-B). *Assume there is an attacker C who can observe all messages, alter messages, insert new messages, delay messages or delete messages transmitted via normal channels between A and B. After a completed run of Protocol III-A (or III-B), the attacker is unable to derive the shared key of A and B.*

Proof. (1) The shared secret can be computed from any of the following equations:

$$K_A = R_A \times (T_B - PK_B),$$

$$K_B = R_B \times (U_A \times G - PK_A),$$

$$K = R_A \times R_B \times G.$$

Therefore, R_A or R_B is required to compute the shared secret.

(2) R_A is hidden by the following equation:

$$U_A = R_A + SK_A.$$

Therefore, SK_A is required to compute R_A.

R_B is hidden by the following equation:

$$T_B = (R_B + SK_B) \times G.$$

Therefore, SK_B is required to compute R_B.

According to the attack model, C has neither SK_A nor SK_B. C is unable to compute R_A or R_B. As a result, C is unable to compute $K_A = K_B = K$. Therefore, we have the conclusion that Protocol III-A provides key confidentiality. Similarly we can prove that Protocol III-B provides key confidentiality. □

Proposition 6.3 (Key integrity of Protocol III-A and III-B)**.** *Assume there is an attacker C who can observe all messages, alter messages, insert new messages, delay messages or delete messages transmitted via normal channels between A and B. After a completed run of Protocol III-A (or III-B), A and B compute the equal secret.*

Proof. (1) As we proved in Theorem 6.1, PK_A, U_A, PK_B and U_B are authenticated.

(2) The secret K_A is computed by A from

$$\begin{aligned} K_A &= R_A \times (T_B - PK_B) \\ &= R_A \times (U_B \times G - PK_B) \\ &= R_A \times ((R_B + SK_B) \times G - PK_B) \\ &= R_A \times R_B \times G. \end{aligned}$$

The secret K_B is computed by B from

$$\begin{aligned} K_B &= R_B \times (U_A \times G - PK_A) \\ &= R_B \times ((R_A + SK_A) \times G - PK_A) \\ &= R_B R_A \times G \\ &= R_A \times R_B \times G \\ &= K_A \end{aligned}$$

Therefore, we have the conclusion that Protocol III-A provides key integrity. Similarly we can prove that Protocol III-B provides key integrity. □

Proposition 6.4 (Key confirmation of Protocol III-A and III-B). *Assume there is an attacker C who can observe all messages, alter messages, insert new messages, delay messages or delete messages transmitted via normal channel between A and B. After a completed run of Protocol III-A (or III-B), both A and B have received evidence confirming that the other party knows the secret.*

Proof. (1) A completed run of Protocol III-A is defined by the validation of mac_A and mac_B. Therefore, after a completed run of Protocol III-A, both A and B have received and validated mac_A and mac_B.

(2) mac_A is computed by A and takes the shared secret as one of the inputs. It is the evidence confirming that A knows the secret.

(3) mac_B is computed by B and takes the shared secret as one of the inputs. It is the

evidence confirming that B knows the secret.

According to (1) and (2), after a completed run of Protocol III-A, B has received the evidence confirming that A knows the secret. According to (1) and (3), after a completed run of Protocol III-A, A has received the evidence confirming that B knows the secret. Therefore, we have the conclusion that Protocol III-A provides key confirmation. Similarly we can prove that Protocol III-B provides key confirmation. □

Proposition 6.5 (Known-key security (key freshness) of Protocol III-A and III-B)**.** *Assume there is an attacker C who can observe all messages, alter messages, insert new messages, delay messages or delete messages transmitted via normal channels between A and B. In addition, C is able to obtain any previous session keys. After a completed run of Protocol III-A (or III-B), C is unable to derive the shared secret from the previous session keys.*

Proof. In Protocol III-A, the computation of the secret takes the R_A and R_B as the inputs. Since R_A and R_B are random values generated by A and B respectively in the key exchange procedure, in each run of Protocol III-A the values are unique. Therefore, the secret is fresh in each run of the protocol. That is, Protocol III-A provides known-key security (key freshness). Similarly we can prove that Protocol III-B provides known-key security. □

Proposition 6.6 (Forward secrecy of Protocol III-A and III-B)**.** *Assume there is an attacker C who can observe all messages, alter messages, insert new messages, delay messages or delete messages transmitted via normal channels between A and B. In addition, C compromises the long-term secrets of A and B. C is unable to derive the previous session keys.*

Proof. (1) In Protocol III-A, C obtains the following information:

$$(E, G, Z_q^*, A, B).$$

In addition, C compromises the SK_A and SK_B.

(2) The values of R_A and R_B are short-term secrets. In practice, they are cleared after use. For a previous run of the protocol, R_A and R_B are are unknown to C. As we proved in Theorem 6.2, R_A or R_B is required to compute the shared secret. Therefore, C is unable to compute the secret of the previous run of the protocol.

Therefore, we have the conclusion that Protocol III-A provides forward secrecy. Similarly we can prove that Protocol III-B provides forward secrecy. □

6.3.2 Resistance to Attacks

The high bandwidth OOB UECDH-based AKE protocols address the vulnerabilities to the man-in-the-middle attack and the impersonation attack. Bellow we illustrate how the two attacks fail.

Resistance to the man-in-the-middle attack

Assume C is a man-in-the-middle attacker to Protocol III-A between A and B. To launch the attack, C interacts with A and B as follows.

1. A generates a random value $R_A \in Z_q^*$ and computes

$$U_A = R_A + SK_A.$$

   Then A sends B M_1 via a high bandwidth OOB channel as follows:

$$A \Rightarrow B : M_1 = (A, PK_A, U_A).$$

2. To launch a man-in-the-middle attack, C intends to intercept and replace M_1. However, since M_1 is transmitted through OOB channel, C is unable to block M_1 and insert his (or her) messages. The man-in-the-middle attack fails.

Therefore, Protocol III-A is resistant to the man-in-the-middle attack. Similarly, Protocol III-B is resistant to the man-in-the-middle attack.

Resistance to the impersonation attack

Assume C is an impersonation attacker to Protocol III-A between A and B. To launch the attack, C impersonates B and interacts with A as follows.

1. A generates a random value $R_A \in Z_q^*$ and computes

$$U_A = R_A + SK_A.$$

   Then A sends B M_1 as follows:

$$A \rightarrow B : M_1 = (A, U_A).$$

2. C intends to block and replace M_1. However, as M_1 is transmitted via OOB channels between A and B, C is unable to block and replace the message. The impersonation attack fails.

Therefore, Protocol III-A is resistant to the impersonation attack. Similarly, Protocol III-B is resistant to the impersonation attack.

6.4 Performance

To study the performance of Protocol III-A and III-B, we choose the Bluetooth OOB protocol as the benchmark. We firstly theoretically evaluate and compare the computational cost. Secondly, we realize prototypes of the protocols and carry out two sets of experiment. The computational time is tested to observe the computational cost.

6.4.1 Evaluation

Denote the cost of computing a scalar multiplication by $\mathcal{S}$ and the cost of computing or verifying a MAC by $\mathcal{H}$. The computational cost is evaluated in Table 6.1. According to Table 6.1 we have the following conclusions:

Table 6.1: Evaluation of computational costs of Protocol III-A, III-B and Bluetooth OOB.

Computation Cost	Cost on A	Cost on B
Protocol III-A	$2\mathcal{H} + \mathcal{S}$	$2\mathcal{H} + 3\mathcal{S}$
Protocol III-B	$2\mathcal{H} + 3\mathcal{S}$	$2\mathcal{H} + \mathcal{S}$
Bluetooth OOB	$4\mathcal{H} + 2\mathcal{S}$	$4\mathcal{H} + 2\mathcal{S}$

- Conclusion 1: Protocol III-A reduces the computational cost on A compared with the Bluetooth OOB protocol;

- Conclusion 2: Protocol III-B reduces the computational cost on B compared with the Bluetooth OOB protocol;

- Conclusion 3: When A is a computationally limited device and B is much powerful than A, the overall performance of Protocol III-A is better than that of the Bluetooth OOB protocol since it lets the powerful side undertake computational tasks on behalf of the limited one. Similarly, when B is a limited device and A is a powerful one, the overall performance of Protocol III-B is better than that of the Bluetooth OOB protocol.

6.4.2 Experiments

We realize prototypes of Protocol III-A, III-B and the Bluetooth OOB protocol using Python programming language. The MAC algorithm is realized through HMAC with SHA-256. The communication is realized through socket programming with TCP. Two sets of experiment are carried out. In Experiment III-1, in order to observe how much computational cost that Protocol III-A and III-B reduce on the initiator and the responder respectively, we use two virtual machines with the same configuration to execute the protocols. In Experiment III-2, in order to simulate two parties with different computational powers, we use a Raspberry Pi and a laptop to execute the protocols.

Table 6.2: Experimental environment of Experiment III-1.

Party	Operating System	Base Memory	Storage
A	Ubuntu 16.04.3 (32-bit)	1024 MB	10 GB
B	Ubuntu 16.04.3 (32-bit)	1024 MB	10 GB

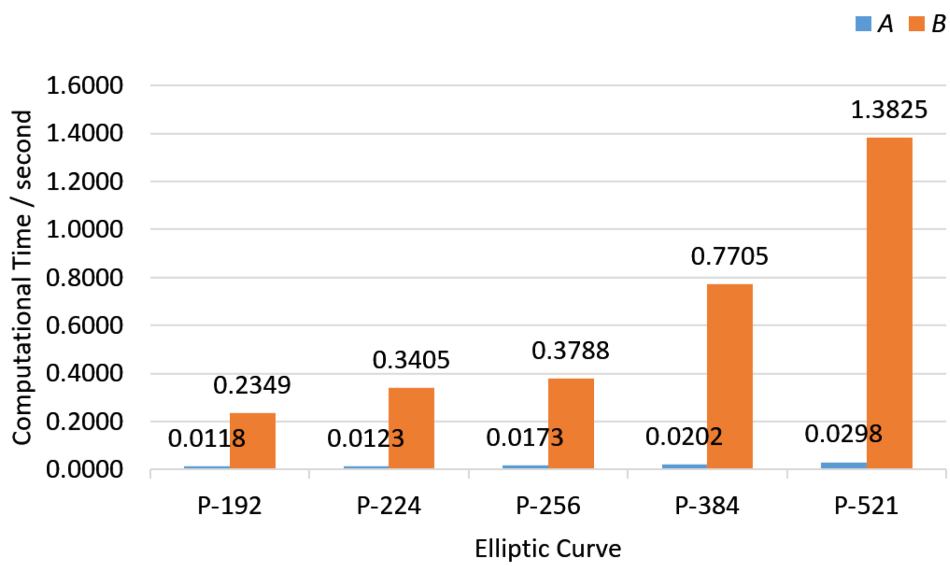

Figure 6.3: Average computational time on A and B of Protocol III-A in Experiment III-1.

Experiment III-1

The initiator A and the responder B are deployed on two virtual machines with the same configuration (Table 6.2). We firstly run Protocol III-A and III-B with five elliptic curves P-192, P-224, P-256, P-384 and P-521 for ten times. The average computational time is illustrated in Figure 6.3 and 6.4. Secondly, we run Protocol III-A, III-B and Bluetooth OOB with the elliptic curve P-256 for ten times. The average computational time is illustrated in Figure 6.6.

Figure 6.3 and 6.4 show that for all of the five curves, Protocol III-A requires less computational time on A than on B; and Protocol III-B requires less computational time on B than on A. According to Figure 6.5, the average computational time on A of Protocol III-A is less than that of the Bluetooth OOB protocol; and the average computational time on B of Protocol III-B is less than that of the Bluetooth OOB protocol. It is corresponding

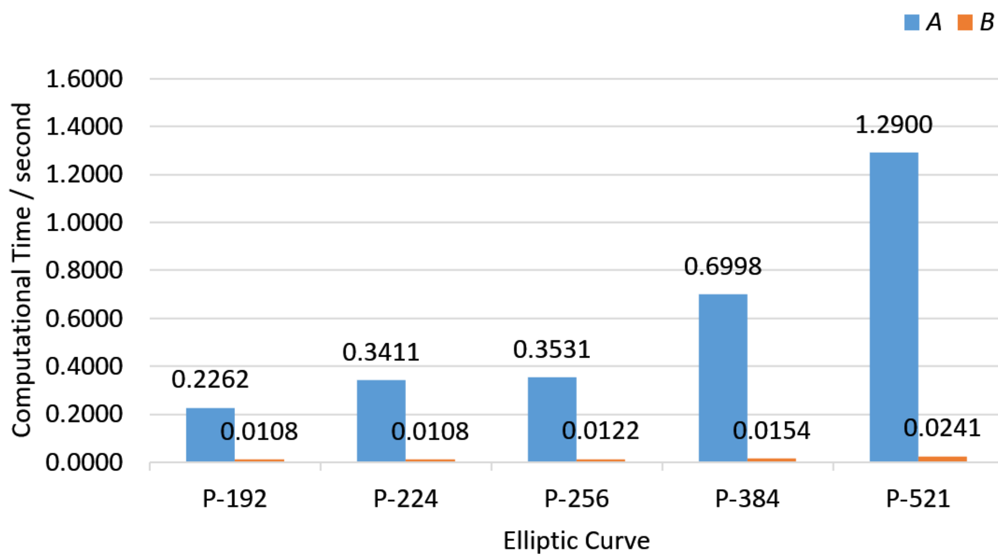

Figure 6.4: Average computational time on A and B of Protocol III-B in Experiment III-1.

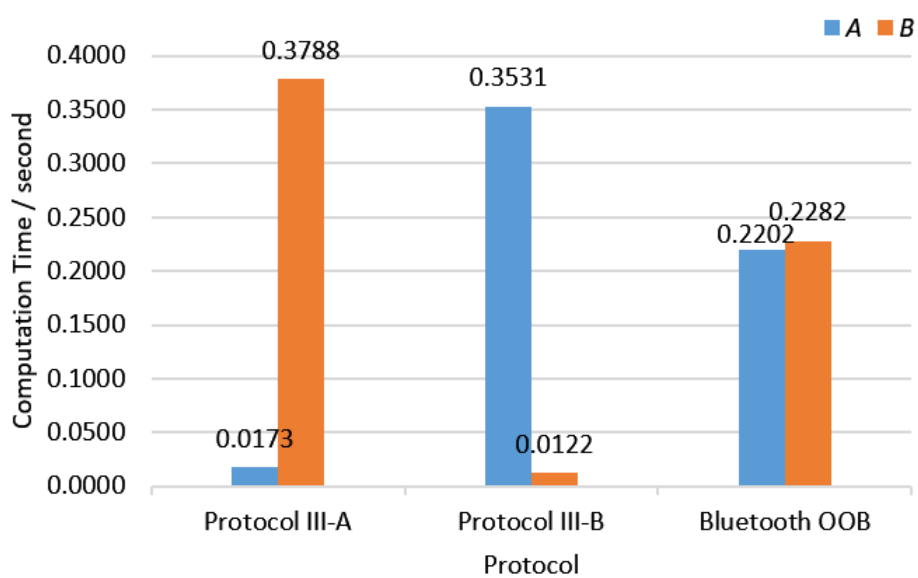

Figure 6.5: Average computational time on A and B of Protocol III-A, III-B and the Bluethooth OOB protocol in Experiment III-1.

Table 6.3: Experimental environment of Experiment III-2.

Experimental Device	CPU	Memory	Hard Disk
Raspberry Pi	1.2 GMHz ARM	1 GB	32 GB
laptop	2.40 GHz i5-6200U	4 GB	120 GB

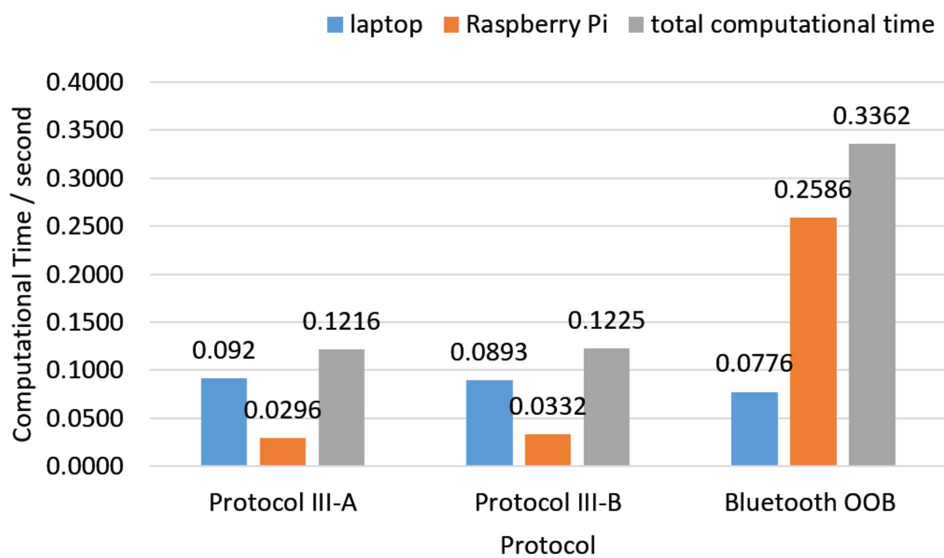

Figure 6.6: Average computational time of A and B of Protocol III-A, III-B and the Bluetooth OOB protocol in Experiment III-2.

to the first two conclusions in Section 6.4.1.

Experiment III-2

In Experiment III-2, we use a Raspberry Pi as the computationally limited device, and a laptop as its powerful communicating partner. For Protocol III-A, we deploy the initiator A on the Raspberry Pi and the responder B on the laptop. For Protocol III-B, we deploy the initiator A on the laptop and the responder B on the Raspberry Pi. Details about the Raspberry Pi and the laptop are listed in Table 6.3.

We run Protocol III-A, III-B and the Bluetooth OOB protocol with the elliptic curve P-256 for ten times. The average computational time is illustrated in Figure 6.6.

According to the Figure 6.6, Protocol III-A and III-B are more friendly to the limited device (Raspberry Pi); and the overall computational time of Protocol III-A and III-B are less than that of Bluetooth OOB. The experimental results are corresponding the third conclusion in Section 6.4.1.

6.5 Chapter Summary

This chapter presented two UECDH-based AKE protocols that use high bandwidth OOB channels. The two protocols remove attacks to the UECDH key exchange scheme through transmitting authenticated messages via high bandwidth OOB channels. The security of the protocols was analyzed; and the resistance to the man-in-the-middle attack and the impersonation attack was analyzed. To observe the performance of the two protocols, the OOB channel based authentication protocol in Bluetooth 5.0 standard was set as the benchmark. The performance of the two protocols and the Bluetooth OOB protocol was studied both through theoretical evaluation and through two sets of experiments. The results show that the two high bandwidth OOB UECDH-based protocols reduce the computational time on the computationally limited device. They are more suitable than the Bluetooth OOB protocol in securing communications between two devices with different computational capabilities.

The proposed protocols transmit long strings (including the public keys) via OOB channels. This requires the two communicating devices can establish high bandwidth OOB channels. In some cases, the communicating devices can only establish low bandwidth OOB channels that are incapable of transmitting long strings. For example, both devices have a small display. In the following chapter, we will illustrate how to design UECDH-based AKE protocols with low bandwidth OOB channels, in particular, the display OOB channels.

Chapter 7

Low Bandwidth OOB UECDH-based AKE Protocols

In Chapter 6 we proposed UECDH-based AKE protocols that utilize high bandwidth OOB channels to transmit authenticated messages. In practice, there are a number of devices that are incapable of establishing high bandwidth OOB channels. In this chapter, we will introduce protocols using short bandwidth OOB channels, for example, the display OOB channels. Low bandwidth OOB channels are used by the user to compare an authentication number computed by the communicating devices. The number is usually a five-digit number which is the positive decimal integer converted from a digest string (i.e., the 16 bit output of MAC). Short hash functions such as the a MAC with 16 bit output usually do not resist combinational attacks. Therefore, the commitment mechanism is utilized in designing low bandwidth OOB UECHD-based protocols. It forces the parities to be (jointly) committed to the digest before knowing what it is until they reveal their respective shares. As a result, combinatorial search attacks such as the birthday attacks fail. Low bandwidth OOB channels are popular in recent years, especially in designing authentication protocols between IoT devices. The international standards IEEE 802.15.6 includes a display authenticated association protocol; and the Bluetooth 5.0 also includes a numeric comparison authentication protocol that is based on display OOB channels.

In the remaining parts of this chapter, we firstly provide an overview of the protocols in terms of the communication model, attack model and security model. Secondly, we describe the low bandwidth OOB UECDH-based AKE protocols: Protocol IV-A which requires less scalar multiplications on A than on B, and Protocol IV-B which requires less scalar multiplications on B than A. Thirdly, we analyze the security of the protocols according the attack and security models; in particular, we discuss how the protocols resist the man-in-the-middle attack and the impersonation attack. Finally, the IEEE Display protocol and the Bluetooth Display protocol are chosen as the benchmarks; prototypes of Protocol IV-A, IV-B and the benchmark protocols are realized. We evaluate and compare the performance of the protocols through theoretical evaluation and experiments.

7.1 Overview

7.1.1 Communication Model

The communication model of a low bandwidth OOB UECDH-based protocol is specified as follows.

- Participants. In each session of the protocol, there are two participants. The participants are denoted by their identities A and B. A is the initiator, and B is the responder. In particular, A and B have significantly different computational capabilities.

- Channels. The channels between A and B include normal channels and low bandwidth OOB channels, for example, the display OOB channels.

7.1.2 Attack Model

The following assumptions and attack model specify what an attacker to a low bandwidth OOB UECDH-based AKE protocol is able and unable to do.

- Basic assumption 1. The attacker is unable to alter, insert, delay or delete messages transmitted in the OOB channel.

- Basic assumption 2. The attacker is unable to break the MAC algorithms.
- Basic ability. The attacker is able to observe all messages, alter messages, insert new messages, delay messages or delete messages transmitted via normal channels between A and B.
- Stronger ability 1. The attacker is able to obtain any previous session key.
- Stronger ability 2. The attacker is able to compromise the long-term secret keys of A or B.

7.1.3 Security Model

Under the above attack model, a low bandwidth OOB UECDH-based AKE protocol aims to achieve the following security goals:

- Key authentication under the attack model that the attacker has the basic ability.
- Key confidentiality under the attack model that the attacker has the basic ability.
- Key integrity under the attack model that the attacker has the basic ability.
- Key confirmation under the attack model that the attacker has the basic ability.
- Known-key security (key freshness) under the attack model that the attacker has the basic ability and the stronger ability 1.
- Forward secrecy under the attack model that the attacker has the basic ability and the stronger ability 2.
- Resistance to combinatorial attacks under the attack model that the attacker has the basic ability.

7.2 Protocol Description

7.2.1 Protocol IV-A

Protocol IV-A shares a secret between a computationally limited initiator A and more powerful responder B. It transfers one scalar multiplication from A to B. The protocol is described through the following procedures: initialization, key exchange and session keys computation. It is also illustrated in Figure 7.1.

Initialization

Before the execution of Protocol IV-A, the initiator and the responder shall obtain their private and public keys respectively. The private keys should be integers in the same finite field. The public keys should be points on the same elliptic curve.

Denote the initiator by A, the responder by B, the finite field by Z_q^*, the elliptic curve by E, the base point of E by G, the private and public keys of A by SK_A and PK_A, and the private and public keys of B by SK_B and PK_B. Formally, the initialization procedure generates the following values:

- Common parameters shared by A and B: $comm = (Z_q^*, E, G)$.
- Information held only by A: PK_A and SK_A where SK_A should be securely stored.
- Information held only by B: PK_B and SK_B where SK_B should be securely stored.

Key Exchange

1. A generates a random value $R_A \in Z_q^*$ and computes

$$U_A = R_A + SK_A,$$

$$commit_A = \mathrm{MAC}(U_A, A \| PK_A)$$

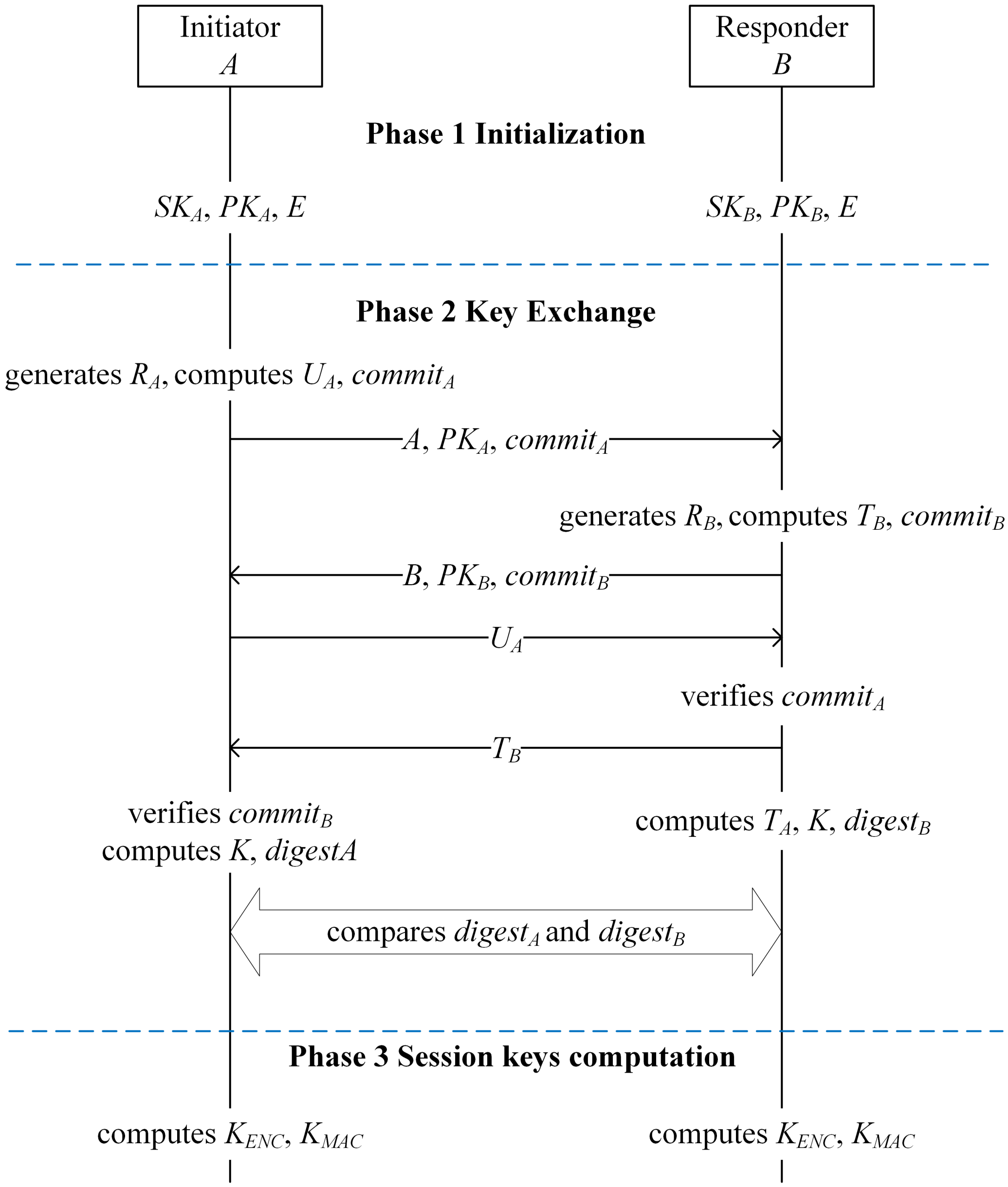

Figure 7.1: Protocol IV-A.

Then A sends B M_1 via a normal channel as follows:

$$A \rightarrow B : M_1 = (A, PK_A, commit_A).$$

2. Upon receiving M_1, B generates a random value $R_B \in Z_q^*$, and computes

$$U_B = R_B + SK_B,$$

$$T_B = U_B \times G,$$

$$commit_B = \mathrm{MAC}(T_B, B \| PK_B).$$

Then B sends A M_2 via a normal channel as follows:

$$B \rightarrow A : M_2 = (B, PK_B, commit_B)$$

3. Upon receiving M_2, A sends B M_3 via a normal channel as follows

$$A \rightarrow B : M_3 = U_A.$$

4. Upon receiving M_3, B firstly verifies $commit_A$.

$$\mathrm{Equal}(\mathrm{MAC}(U_A, A \| PK_A), commit_A) = \begin{cases} 1, & valid \\ 0, & invalid \end{cases}$$

Secondly, if $commit_A$ is valid, B sends A M_4 via a normal channel as follows:

$$B \rightarrow A : M_4 = T_B.$$

Thirdly, B computes the shared secret K_B as follows:

$$T_A = U_A \times G,$$

$$K_B = R_B \times (T_A - PK_A).$$

Finally, B computes a digest as follows and shows it on the display:

$$digest_B = \mathrm{MAC}_{16}(K_{Bx}, A\|B\|PK_A\|PK_B\|U_A\|T_B).$$

5. Upon receiving M_4, A firstly verifies $commit_B$ as follows:

$$\mathrm{Equal}(\mathrm{MAC}(T_B, B\|PK_B), commit_B) = \begin{cases} 1, & valid \\ 0, & invalid \end{cases}$$

Secondly, if $commit_B$ is valid, A computes the shared secret K_A as follows:

$$K_A = R_A \times (T_B - PK_B).$$

Finally, A computes a digest *digest* as follows and shows it on the display:

$$digest_A = \mathrm{MAC}_{16}(K_{Ax}, A\|B\|PK_A\|PK_B\|U_A\|T_B).$$

6. A and B compares the digests.

Session Keys Computation

If $digest_A = digest_B$, A derives the session keys from K_{Ay} as follows:

$$K_{ENC} = \mathrm{F}(K_{Ay}, 1),$$

$$K_{MAC} = \mathrm{F}(K_{Ay}, 2).$$

B derives the session keys from K_{By} as follows:

$$K_{ENC} = \mathrm{F}(K_{By}, 1),$$

$$K_{MAC} = \mathrm{F}(K_{By}, 2).$$

7.2.2 Protocol IV-B

Protocol IV-B shares a secret between a powerful initiator A and a computationally limited responder B. It transfers one scalar multiplication from B to A. We describe the protocol through the following procedures: initialization, key exchange and session keys computation in detail. In addition, we also illustrate the procedures through Figure 7.2.

Initialization

The initialization here is similar with that of Protocol IV-A. Let the notations A, B, Z_q^*, E, G, SK_A, PK_A, SK_B and PK_B be the same as we specified in Section 6.2.1. The initialization procedure produces the following values:

- Common parameters shared by A and B: $comm = (Z_q^*, E, G)$.
- Information held only by A: (PK_A, SK_A) where SK_A should be secretly stored.
- Information held only by B: (PK_B, SK_B) where SK_B should be secretly stored.

Key Exchange

1. A generates a random value $R_A \in Z_q^*$ and computes

$$U_A = R_A + SK_A,$$

$$T_A = U_A \times G,$$

$$commit_A = \mathrm{MAC}(T_A, A \| PK_A).$$

Then A sends B M_1 via a normal channel as follows:

$$A \to B : M_1 = (A, PK_A, commit_A).$$

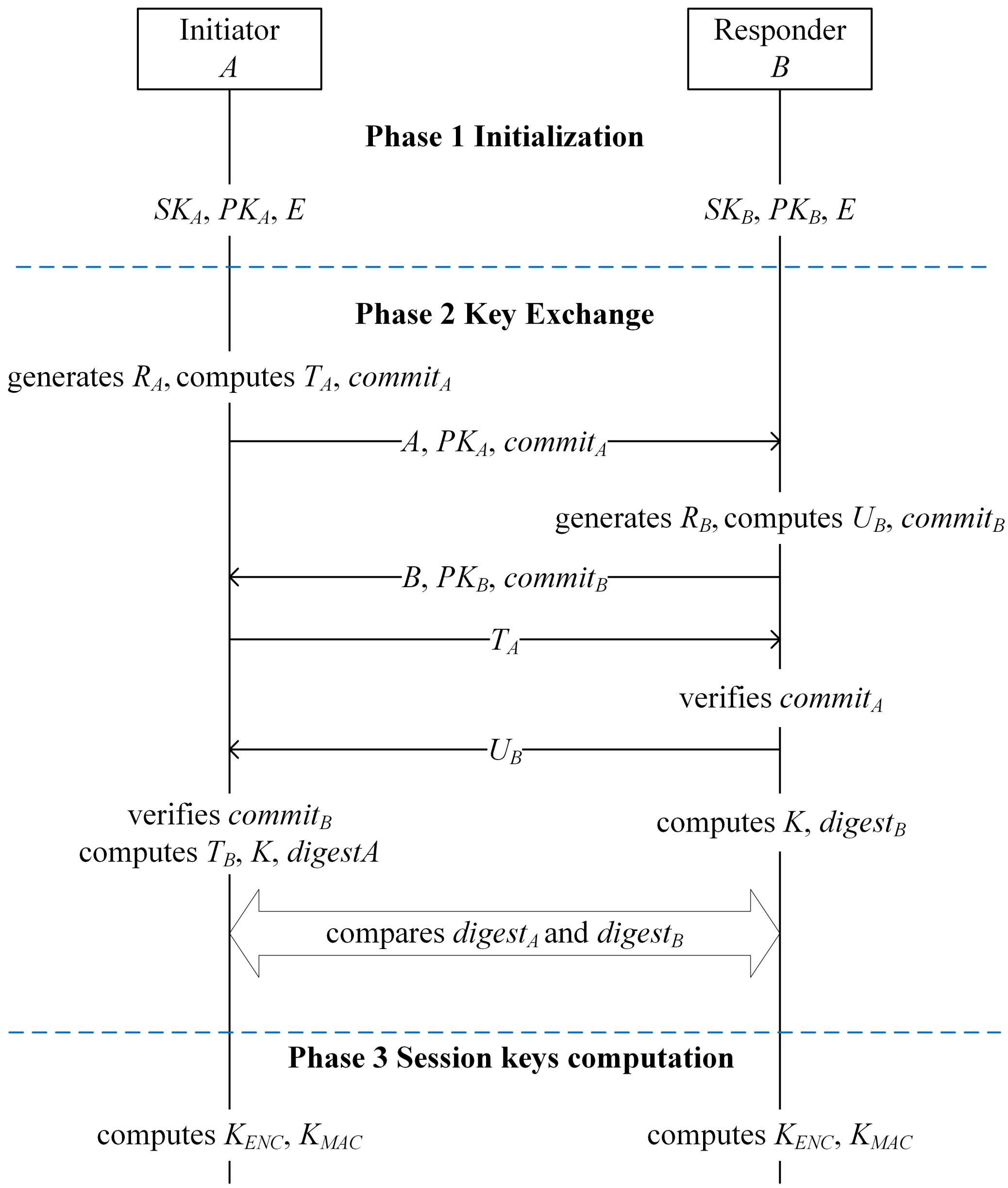

Figure 7.2: Protocol IV-B.

2. Upon receiving M_1, B generates a random value $R_B \in Z_q^*$, and computes

$$U_B = R_B + SK_B.$$

$$commit_B = \text{MAC}(U_B, B \| PK_B).$$

Then B sends A M_2 via a normal channel as follows:

$$B \to A : M_2 = (B, PK_B, commit_B).$$

3. Upon receiving M_2, A sends B M_3, i.e. T_A via a normal channel.

$$A \to B : M_2 = T_A.$$

4. Upon receiving M_3, B verifies $commit_A$ as follows:

$$\text{Equal}(\text{MAC}(U_A, A \| PK_A), commit_A) = \begin{cases} 1, & valid \\ 0, & invalid \end{cases}$$

Secondly, if $commit_A$ is valid, B sends A M_4, i.e. U_B via a normal channel.

$$B \to A : M_4 = U_B.$$

Thirdly, B computes the shared secret K_B as follows

$$K_B = R_B \times (T_A - PK_A).$$

Finally, B computes a digest as follows and shows it on the display:

$$digest_B = \text{MAC}_{16}(K_{Bx}, A \| B \| PK_A \| PK_B \| T_A \| U_B).$$

5. Upon receiving M_4, A firstly verifies $commit_B$ as follows

$$\text{Equal}(\text{MAC}(U_B, B \| PK_B), commit_B) = \begin{cases} 1, & valid \\ 0, & invalid \end{cases}$$

Secondly, if $commit_B$ is valid, A computes the shared secret K_A as follows

$$T_B = U_B \times G,$$

$$K_A = R_A \times (T_B - PK_B).$$

Finally, A computes a digest $digest_A$ as follows and shows it on the display:

$$digest_A = \text{MAC}_{16}(K_{Ax}, A \| B \| PK_A \| PK_B \| T_A \| U_B)$$

6. A and B compares the digests.

Session Keys Computation

If $digest_A = digest_B$, A derives the session keys from K_{Ay} as follows:

$$K_{ENC} = \text{F}(K_{Ay}, 1),$$

$$K_{MAC} = \text{F}(K_{Ay}, 2).$$

B derives the session keys from K_{By} as follows:

$$K_{ENC} = \text{F}(K_{By}, 1),$$

$$K_{MAC} = \text{F}(K_{By}, 2).$$

7.3 Security

This section illustrates that the two low bandwidth OOB UECDH-based AKE protocols achieve the security goals (Section 7.1.3) under the attack model (Section 7.1.2). For each security goal, we provide a proposition that states a security feature, and prove how the proposition stands. In addition, we also show how the two protocols resist the man-in-the-middle attack and the impersonation attack.

7.3.1 Security Features

Proposition 7.1 (Key authentication of Protocol IV-A and IV-B)**.** *Assume there is an attacker C who can observe all messages, alter messages, insert new messages, delay messages or delete messages transmitted via the normal channels between A and B. After a completed run of Protocol IV-A (or IV-B), A (or B) believes that he (or she) shares a secret B (or A) other than any other party.*

Proof. (1) A completed run of Protocol IV-A is defined by the equality of $digest_A$ and $digest_B$; and the validation of $commit_A$ and $commit_B$ is the preconditions to comparing $digest_A$ and $digest_B$. Therefore, after a completed run of Protocol IV-A, $commit_A$ and $commit_B$ are validated.

(2) Since $commit_A = \mathrm{MAC}(A, PK_A, U_A)$, the validation of $commit_A$ guarantees the authenticity of PK_A and U_A. Similarly, the validation of $commit_B$ guarantees the authenticity of PK_B and T_B. Therefore, after a completed run of Protocol IV-A, PK_A, U_A, PK_B and T_B are authenticated.

(3) Since $K_A = R_A \times (T_B - PK_B)$, R_A is generated by A and T_B and PK_B are authenticated according to (2), A believes that K_A is the shared secret B other than any other party.

(4) Since $K_B = R_B \times (U_A \times G - PK_A)$, R_B is generated by B and U_A and PK_A are authenticated according to (2), B believes that K_B is the shared secret A other than any other party.

According to (3) and (4), Protocol IV-A provides key authentication under the basic assumptions and the attack model that C has the basic ability. Similarly we can prove that Protocol IV-B provides key authentication under the basic assumptions and the attack model that C has the basic ability. □

Proposition 7.2 (Key confidentiality of Protocol IV-A and IV-B). *Assume there is an attacker C who can observe all messages, alter messages, insert new messages, delay messages or delete messages transmitted via normal channels between A and B. After a completed run of Protocol IV-A (or IV-B), the attacker is unable to derive the shared key of A and B.*

Proof. (1) The shared secret can be computed from any of the following equations:

$$K_A = R_A \times (T_B - PK_B),$$

$$K_B = R_B \times (U_A \times G - PK_A),$$

$$K = R_A R_B \times G.$$

Therefore, R_A or R_B is required to compute the shared secret.

(2) Since R_A is hidden by the following equation:

$$U_A = R_A + SK_A,$$

SK_A is required to compute R_A.

Since R_B is hidden by the following equation:

$$T_B = (R_B + SK_B) \times G,$$

SK_B is required to compute R_B.

(3) According to the attack model, C has neither SK_A nor SK_B. C is unable to compute R_A or R_B. Therefore, C is unable to compute $K_A = K_B = K$.

According to (3), we have the conclusion that Protocol IV-A provides key confidentiality under the basic assumptions and the attack model that C has the basic ability. Similarly we can prove that Protocol IV-B provides key confidentiality under the basic assumptions and the attack model that C has the basic ability. □

Proposition 7.3 (Key integrity of Protocol IV-A and IV-B)**.** *Assume there is an attacker C who can observe all messages, alter messages, insert new messages, delay messages or delete messages transmitted via normal channel between A and B. After a completed run of Protocol IV-A (or IV-B), A and B computes the equal secret.*

Proof. (1) As we proved in Proposition 7.1, a completed run of Protocol IV-A (or IV-B) implies authenticity of T_B, PK_B, U_A and PK_A.

(2) The secret K_A is computed by A from

$$\begin{aligned} K_A &= R_A \times (T_B - PK_B) \\ &= R_A \times (U_B \times G - PK_B) \\ &= R_A \times ((R_B + SK_B) \times G - PK_B) \\ &= R_A \times R_B \times G. \end{aligned}$$

The secret K_B is computed by B from

$$\begin{aligned} K_B &= R_B \times (U_A \times G - PK_A) \\ &= R_B \times ((R_A + SK_A) \times G - PK_A) \\ &= R_B R_A \times G \\ &= R_A \times R_B \times G \\ &= K_A \end{aligned}$$

Therefore, we have the conclusion that Protocol IV-A provides key integrity under the basic assumptions and the attack model that C has the basic ability. Similarly we can prove that Protocol IV-B provides key integrity under the basic assumptions and the attack model that C has the basic ability. □

Proposition 7.4 (Key confirmation of Protocol IV-A and IV-B). *Assume there is an attacker C who can observe all messages, alter messages, insert new messages, delay messages or delete messages transmitted via normal channel between A and B. After a completed run of Protocol IV-A (or IV-B), both A and B have received evidence confirming that the other party knows the secret.*

Proof. (1) A completed run of Protocol IV-A is defined by the equality of $digest_A$ and $digest_B$.

(2) $digest_A$ is computed by A and takes K_A as part of the inputs. It is the evidence confirming that A knows the secret.

(3) $digest_B$ is computed by B and takes K_B as part of the inputs. It is the evidence confirming that B knows the secret.

According to (1) and (2), after a completed run of Protocol IV-A, B has received evidence confirming that A knows the secret; and according to (1) and (3), after a completed run of Protocol IV-A, A has received evidence confirming that B knows the secret. Therefore, we have the conclusion that Protocol IV-A provides key confirmation under the basic assumptions and the attack model that C has the basic ability. Similarly we can prove that Protocol IV-B provides key confirmation under the basic assumptions and the attack model that C has the basic ability. □

Proposition 7.5 (Known-key security (key freshness) of Protocol IV-A and IV-B). *Assume there is an attacker C who can observe all messages, alter messages, insert new messages, delay messages or delete messages transmitted via normal channels between A and B. In addition, C is able to obtain any previous session keys. After a completed run of Protocol IV-A (or IV-B), C is unable to derive the shared secret from the previous session keys.*

Proof. In Protocol IV-A, the computation of the shared secret takes the R_A and R_B as the inputs. Since R_A and R_B are random values generated by A and B respectively in the key exchange procedure, in each run of Protocol IV-A the values are unique. Therefore, the secret is fresh in each run of the protocol. That is, Protocol IV-A provides known-key security (key freshness) under the basic assumptions and the attack model that C has the basic ability and the stronger ability 1. Similarly we can prove that Protocol IV-B provides known-key security under the basic assumptions and the attack model that C has the basic ability and the stronger ability 1. □

Proposition 7.6 (Forward secrecy of Protocol IV-A and IV-B)**.** *Assume there is an attacker C who can observe all messages, alter messages, insert new messages, delay messages or delete messages transmitted via normal channels between A and B. In addition, C compromises the long-term secrets of A and B. C is unable to derive the previous session keys.*

Proof. (1) In Protocol V-A, C obtains the following information:

$$(E, G, Z_q^*, A, B, PK_A, PK_B, U_A, T_B).$$

In addition, C compromises SK_A and SK_B.

(2) As we proved in Theorem 7.2, for a previous session of the protocol, R_A or R_B in that session is required to compute the shared secret. According to (1), C does not obtain R_A or R_B (which is short-term secret). Therefore, C is unable to compute the secret of the previous run of the protocol.

According to (2), we have the conclusion that Protocol IV-A provides forward secrecy under the basic assumptions and the attack model that C has the basic ability and the stronger ability 2. Similarly we can prove that Protocol IV-B provides forward secrecy under the basic assumptions and the attack model that C has the basic ability and the stronger ability 2. □

Proposition 7.7 (Resistance to combinatorial attacks of Protocol IV-A and IV-B)**.** *Assume there is an attacker C who can observe all messages, alter messages, insert new messages, delay messages or delete messages transmitted via normal channels between A and B. C is unable to break the short MAC outputs (i.e., the digests) through combinatorial attacks.*

Proof. In Protocol IV-A, the value of $digest_A = disgest_B$ is determined in step 1; the value U_A to compute $digest_A$ is kept secret until step 3; and the value T_B to compute $digest_B$ is kept secret until step 4. As a result, C is unable to break $digest_A = digest_B$ through combinatorial attacks before it is displayed in step 4. Therefore, we have the conclusion that Protocol IV-A is resistant to combinatorial attacks. Similarly, we can prove that Protocol IV-B is resistant to combinatorial attacks. □

7.3.2 Resistance to Attacks

The low bandwidth OOB UECDH-based AKE protocols address the vulnerabilities to the man-in-the-middle attack and the impersonation attack. Bellow we illustrate how these attacks fail.

Resistance to the man-in-the-middle attack

Assume C is a man-in-the-middle attacker to Protocol IV-A between A and B. To launch the attack, C interacts with A and B as follows.

1. A generates a random value $R_A \in Z_q^*$ and computes

$$U_A = R_A + SK_A,$$

$$commit_A = \mathrm{MAC}(U_A, A \| PK_A).$$

   Then A sends B M_1 via a normal channel as follows:

$$A \to B : M_1 = (A, PK_A, commit_A).$$

2. C firstly intercepts M_1.

   Secondly, C generates a random value $R_C \in Z_q^*$ and computes

$$U_C = R_C + SK_C,$$

$$commit_{CA} = \mathrm{MAC}(U_C, A \| PK_C).$$

At last, C sends a forged message $(A, PK_C, commit_{CA})$ to B, i.e.,

$$C \rightarrow B : M_1' = (A, PK_C, commit_C)$$

3. Upon receiving M_1', B generates a random value $R_B \in Z_q^*$ and computes

$$U_B = R_B + SK_B,$$

$$T_B = U_B \times G,$$

$$commit_B = \mathrm{MAC}(T_B, B \| PK_B).$$

Then B sends A M_2 via a normal channel as follows:

$$B \rightarrow A : M_2 = (B, PK_B, commit_B)$$

4. C firstly intercepts M_2.

Secondly, C computes

$$T_C = U_C \times G,$$

$$commit_{CB} = \mathrm{MAC}(T_C, B \| PK_C).$$

Thirdly, C sends a forged message $(B, PK_C, commit_{CB})$ to A, i.e.,

$$C \rightarrow A : M_2' = (B, PK_C, commit_{CB}).$$

4. Upon receiving M_2', A sends B M_3 as follows:

$$A \rightarrow B : M_3 = U_A$$

5. C intercepts M_3 and replaces U_A with U_C, i.e.,

$$C \to B : M_3' = U_C$$

6. Upon receiving M_3', B firstly verifies $commit_{CA}$. The verification will succeed.

   Second, B sends A with the following message

$$B \to A : M_4 = T_B$$

   Third, B computes the shared secret (with C) as follows

$$T_A' = U_C \times G,$$

$$K_B' = R_B \times (T_A' - PK_C) = R_C R_B \times G.$$

   Finally, B computes a digest as follows and shows it on the display:

$$digest_B = \mathrm{MAC}_{16}(K_{Bx}', A\|B\|PK_C\|PK_B\|U_C\|T_B).$$

7. C intercepts M_4 and replace T_B with T_C, i.e.,

$$C \to A : M_4' = T_C.$$

8. Upon receiving M_4', A first verifies $commit_{CB}$. The verification will succeed.

   Second, A computes the shared secret (with C) as follows

$$K_A' = R_A \times (T_C - PK_C) = R_C R_A \times G.$$

Finally, A computes a digest as follows and shows it on the display:

$$digest_A = \mathrm{MAC}_{16}(K'_{Ax}, A\|B\|PK_A\|PK_C\|U_A\|T_C).$$

However, $digest_A \neq digest_B$ since $K'_A \neq K'_B$ and $A\|B\|PK_A\|PK_C\|U_A\|T_C \neq A\|B\|PK_C\|PK_B\|U_C\|T_B$. The attack fails at this stage.

Therefore, Protocol IV-A is resistent to the man-in-the-middle attack. Similarly, Protocol IV-B is resistent to the man-in-the-middle attack.

Resistance to the impersonation attack

Assume C is an impersonation attacker to Protocol IV-A between A and B. To launch the attack, C impersonates B and interacts with A as follows.

1. A generates a random value $R_A \in Z_q^*$ and computes

$$U_A = R_A + SK_A,$$

$$commit_A = \mathrm{MAC}(U_A, A\|PK_A)$$

   Then A sends B M_1 as follow:

$$A \rightarrow B : M_1 = (A, PK_A, commit_A).$$

2. C firstly intercepts and blocks M_1.

   Secondly, C generates a random value $R_C \in Z_q^*$ and computes

$$U_C = R_C + SK_C,$$

$$T_C = U_C \times G$$

$$commit_{CA} = \mathrm{MAC}(T_C, B\|PK_C)$$

Thirdly, C sends A M_2' as follows

$$C \rightarrow A : M_2' = (B, PK_C, commit_C).$$

3. Upon receiving M_2', A sends B with the following message

$$A \rightarrow B : M_3 = U_A.$$

4. C firstly intercepts and blocks M_2.

   Secondly, C sends the following message to A.

$$C \rightarrow A : M_4' = T_C$$

   At this stage, C needs to computes a digest and shows it on B's display. However, C is unable to show a value on B's display. The attack fails.

Therefore, Protocol IV-A is resistent to the impersonation attack. Similarly, Protocol IV-B is resistent to the impersonation attack.

7.4 Performance

To study the performance of Protocol IV-A and Protocol IV-B, we choose the Bluetooth Display protocol and the IEEE Display protocol as the benchmarks. We firstly theoretically evaluate and compare the computational cost. Secondly, we realize prototypes of the protocols and carry out two sets of experiment. The computational time is tested to observe the computational cost.

7.4.1 Evaluation

Denote the cost of computing a scalar multiplication by $\mathcal{S}$ and the cost of computing or verifying a MAC by $\mathcal{H}$. The computational cost is evaluated in Table 7.1. According to

Table 7.1: Evaluation of computational costs of Protocol IV-A, IV-B, the Bluetooth Display protocol and the IEEE Display protocol.

Computation Cost	Cost on A	Cost on B
Protocol IV-A	$3\mathcal{H} + \mathcal{S}$	$3\mathcal{H} + 3\mathcal{S}$
Protocol IV-B	$3\mathcal{H} + 3\mathcal{S}$	$3\mathcal{H} + \mathcal{S}$
Bluetooth Display	$2\mathcal{H} + 2\mathcal{S}$	$2\mathcal{H} + 2\mathcal{S}$
IEEE Display	$3\mathcal{H} + 2\mathcal{S}$	$3\mathcal{H} + 2\mathcal{S}$

Table 7.1 we have the following conclusions:

- Conclusion 1: Protocol IV-A reduces the computational cost on A compared with the Bluetooth Display protocol and the IEEE Display protocol;
- Conclusion 2: Protocol IV-B reduces the computational cost on B compared with the Bluetooth Display protocol and the IEEE Display protocol;
- Conclusion 3: When A is a computationally limited device and B is much powerful than A, the overall performance of Protocol IV-A is better than that of the Bluetooth Display protocol and the IEEE Display protocol since it let the powerful side undertake computational tasks on behalf of the limited one. Similarly, when B is a limited device and A is a powerful one, the overall performance of Protocol IV-B is better than that of the Bluetooth Display protocol and the IEEE Display protocol.

7.4.2 Experiments

We realize prototypes of Protocol IV-A, IV-B, the Bluetooth Display protocol and the IEEE Display protocol using Python programming language. The MAC algorithm is realized through HMAC with SHA-256. The communication is realized through socket programming with TCP. Two sets of experiment are carried out. In Experiment IV-1, in order to observe how much computational cost that Protocol IV-A and IV-B reduce on the initiator and the responder respectively, we use two virtual machines with the same configuration to execute the protocols. In Experiment IV-2, in order to simulate two parties

Table 7.2: Experimental environment of Experiment IV-1.

Party	Operating System	Base Memory	Storage
A	Ubuntu 16.04.3 (32-bit)	1024 MB	10 GB
B	Ubuntu 16.04.3 (32-bit)	1024 MB	10 GB

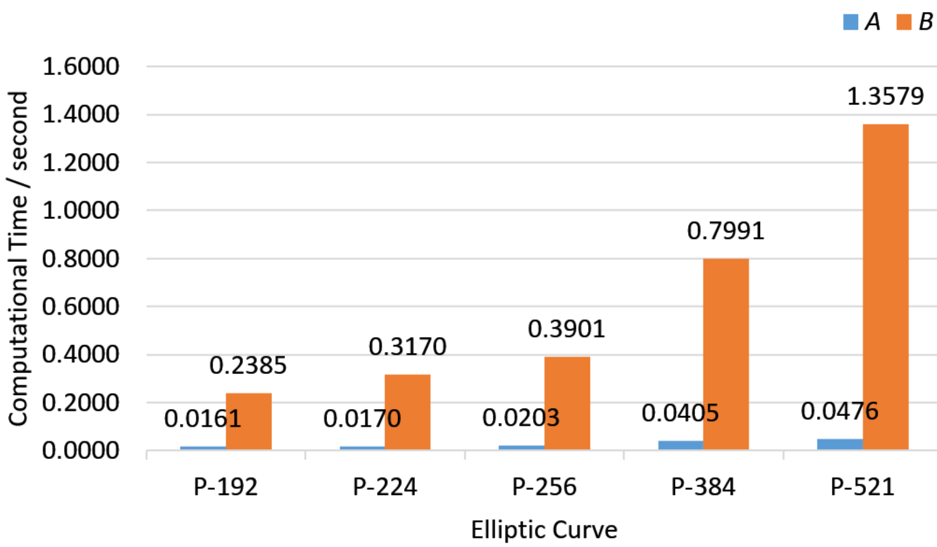

Figure 7.3: Average computational time on A and B of Protocol IV-A in Experiment IV-1.

with different computational powers, we use a Raspberry Pi and a laptop to execute the protocols.

Experiment IV-1

The initiator A and the responder B are deployed on two virtual machines with the same configuration (Table 7.2). We firstly run Protocol IV-A and IV-B with five elliptic curves P-192, P-224, P-256, P-384 and P-521 for ten times. The average computational time is illustrated in Figure 7.3 and 7.4. Secondly, we run IV-A, IV-B, Bluetooth Display and IEEE Display with the elliptic curve P-256 for ten times. The average computational time is illustrated in Figure 7.5.

Figure 7.3 and 7.4 show that for all of the five curves, Protocol IV-A has less computational time on A than on B; and Protocol IV-B has less computational time on B than on

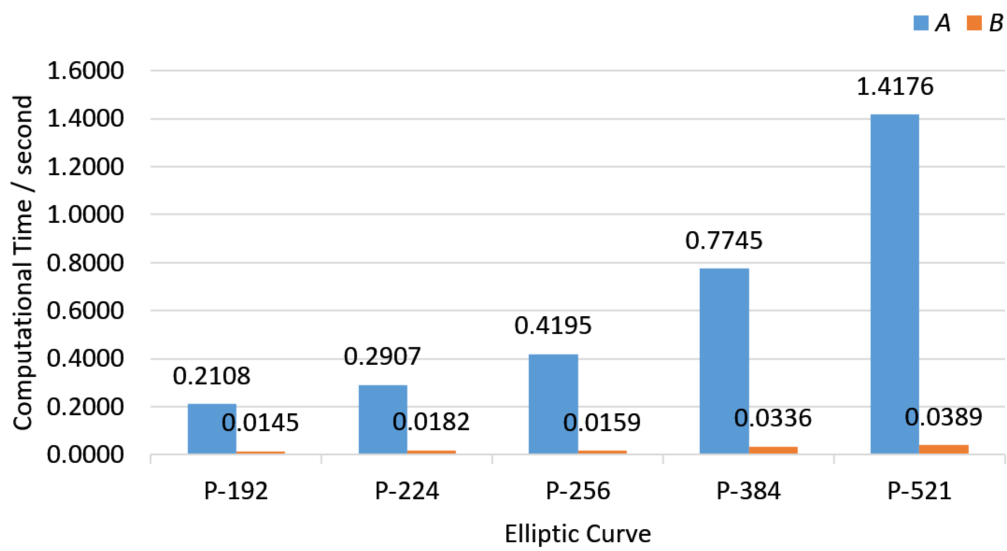

Figure 7.4: Average computational time on A and B of Protocol IV-B in Experiment IV-1.

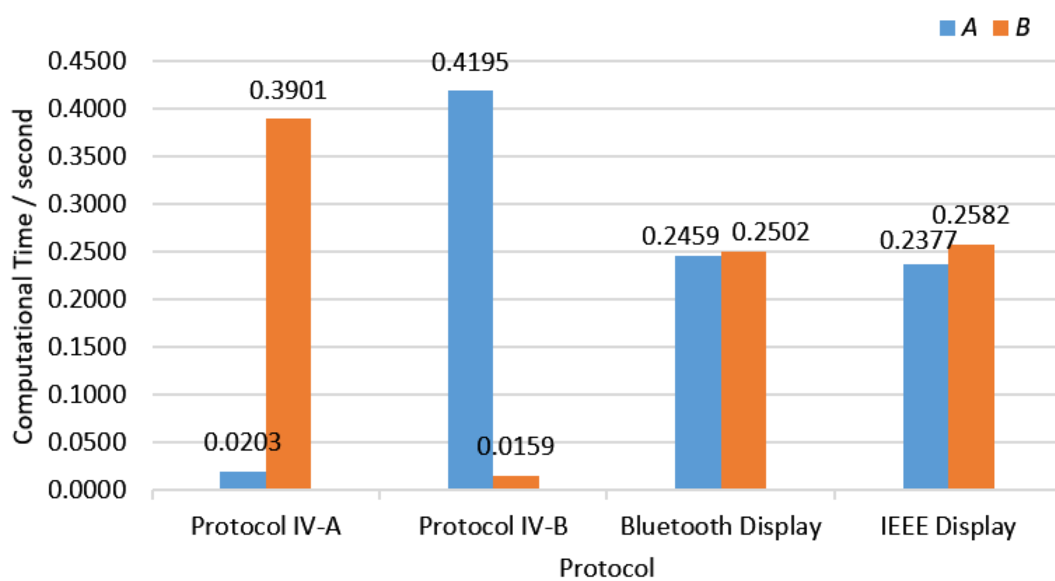

Figure 7.5: Average computational time on A and B of Protocol IV-A, IV-B, Bluetooth Display and IEEE Display in Experiment IV-1.

Table 7.3: Experimental environment of Experiment IV-2.

Experimental Device	CPU	Memory	Hard Disk
Raspberry Pi	1.2 GMHz ARM	1 GB	32 GB
laptop	2.40 GHz i5-6200U	4 GB	120 GB

A. According to Figure 7.5, the average computational time on A of Protocol IV-A is less than that of the Bluetooth Display protocol and the IEEE Display protocol; and the average computational time on B of Protocol IV-B is less than that of the Bluetooth Display protocol and the IEEE Display protocol. It is corresponding to the first two conclusions in Section 7.4.1.

Experiment IV-2

In Experiment IV-2, we use a Raspberry Pi as the computationally limited device, and a laptop as its powerful communicating partner. For Protocol IV-A, we deploy the initiator A on the Raspberry Pi and the responder B on the laptop. For Protocol IV-B, we deploy the initiator A on the laptop and the responder B on the Raspberry Pi. Details about the Raspberry Pi and the laptop are listed in Table 7.3.

We run Protocol IV-A, IV-B, the Bluetooth Display protocol and the IEEE Display protocol with the elliptic curve P-256 for ten times. The average computational time is illustrated in Figure 7.6.

According to Figure 7.6, Protocol IV-A and IV-B are more friendly to the limited device (Raspberry Pi); and the overall computational time of Protocol IV-A and IV-B are less than that of the Bluetooth Display protocol and the IEEE Display protocol. The experimental results are corresponding to the third conclusion in Section 7.4.1.

7.5 Chapter Summary

This chapter presented two low bandwidth OOB UECDH-based AKE protocols. The two protocols remove attacks to UECDH through the commitment mechanism and comparing

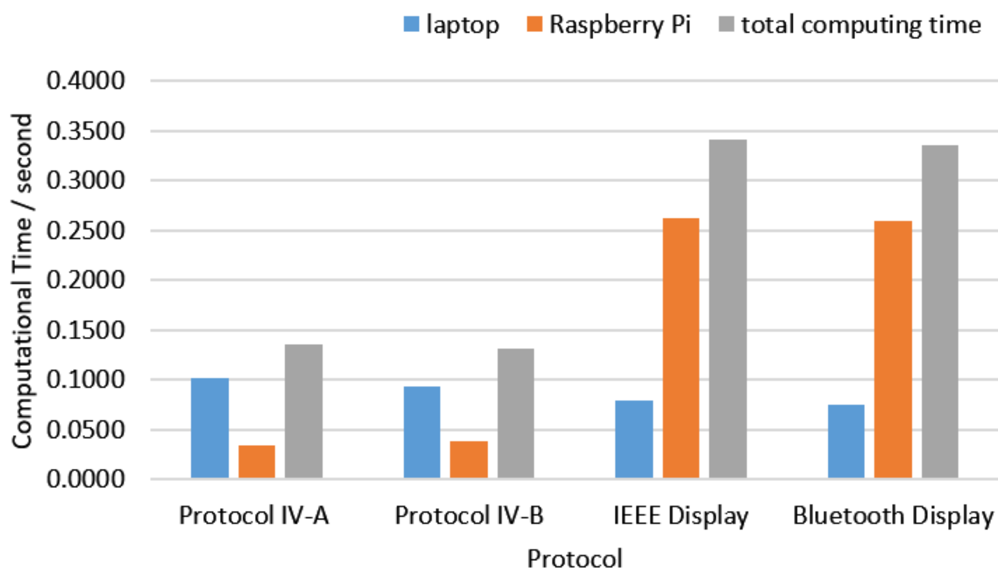

Figure 7.6: Average computational time on A and B of Protocol IV-A, IV-B, the Bluetooth Display protocol and the IEEE Display protocol in Experiment IV-2.

digests via low bandwidth OOB channel, i.e., the displays. The security of the protocols was analyzed; and the resistance to the man-in-the-middle attack and the impersonation attack was analyzed. To observe the performance of the two protocols, the display association protocol in IEEE 802.15.6 and the numeric comparison protocol in Bluetooth 5.0 were set as the benchmarks. The prototypes of the two protocols and the benchmark protocols were realized. The performance was studied and compared both through theoretical evaluation and two sets of experiments. The results show that the low bandwidth UECDH-based AKE protocols reduce the computational time on the computationally limited device; and have better overall performance than the benchmarks. They are more suitable than the benchmark protocols in securing communications between two devices with different computational capabilities.

Chapter 8

Conclusion and Future Work

In this thesis, we studied AKE protocols with unbalanced computational requirements. In particular, we proposed UECDH key exchange scheme by transferring one scalar multiplication from one party to its communicating partner in the ECDH key exchange scheme; and utilizing different authentication measures, we presented four sets of UECDH-based AKE schemes that are suitable for a variety of use cases. Similar protocols from international standards including IEEE 802.15.6, TLS and Bluetooth were set as benchmarks. We realized prototypes for the proposed protocols and benchmark protocols. Performance of these protocols was evaluated and tested in terms of computational time. The experimental results show that the proposed protocols have better performance than the benchmark protocols. This chapter gives a brief review for the four sets of UECDH-based AKE protocols and presents the future work.

8.1 Conclusion

The four sets of UECDH-based AKE protocol are concluded as follows. In addition, we also compare them in Table 8.1 in terms of the authentication measure, benchmark, advantage and limitation.

Table 8.1: Comparison of UEDCH-based AKE protocols.

Protocol	Benchmark	Authentication Measure	Advantage	Limitation
Protocol I-A, I-B	IEEE PW	Securely pre-sharing password and hiding the public key of the party undertaking more computations	Low storage cost	Security vulnerabilities in one-to-many communicating scenarios
Protocol II-A, II-B	TLS PK Authenticated	Issuing authenticated public keys through trusted third party	Removing security vulnerability in one-to-many communicating scenarios	Complexity and expensive cost on issuing and maintaining public key certificates
Protocol III-A, III-B	Bluetooth OOB	High bandwidth OOB channels	Removing the relying on pre-shard secrets and trusted third party	Requiring high bandwidth OOB channels
Protocol IV-A, IV-B	Bluetooth Display and IEEE Display	Display OOB channels and commitment mechanism	Removing the relying on pre-shard secret and trusted third party	Requiring both parties have a display

8.1.1 Password UECDH-based AKE Protocols

Passwords are short secrets that are easily remembered by humans. Therefore, instead of being securely stored in the device, they can be input by the human users during the run of a protocol. AKE protocols using passwords have been widely studied. The IEEE Standard 802.15.6 also includes a password authenticated association protocol. In this thesis, we proposed two password UECDH-based AKE protocols: Protocol I-A and I-B. The password in the protocols is used for both authentication and hiding the public key (of the party undertaking more computations).

The advantage of password UECDH-based AKE protocols is the lower storage cost on both parties. The limitation is the unsuitability for one-to-many communicating scenarios. When the party undertaking more computations needs to execute the protocol with more than one parties, the public key will be stored by more parties and no longer kept hidden. This may lead to security vulnerabilities.

8.1.2 Public Key Authenticated UECDH-based AKE Protocols

In practice, communicating parties can acquire authenticated public key of each other through the PKI. In the PKI, there is a CA issuing public key certificates for the parties. The PKI-based measure is a more conventional and widely used way to exchange public keys for secure handshakes. For example, the most widely used security scheme, TLS, involves several AKE protocols that are based on PKI. In this thesis, we proposed two public key authenticated UECDH-based AKE protocols: Protocol II-A and II-B. The protocols assume both parties have the authenticated public key of each other through PKI prior to the key exchange procedure. In addition, the digital signature algorithm is used to guarantee authentication.

The public key authenticated UECDH-based AKE protocols do not require hiding the public keys. Therefore, compared with Protocol I-A and I-B, the advantage is that they remove security vulnerabilities in one-to-many communicating scenarios. However, issuing and maintaining the public key certificates are complicated and expensive.

8.1.3 High Bandwidth OOB UECDH-based AKE Protocols

The OOB channels are resistant to a number of attacks; therefore, they are often used to transmit authenticated messages in security protocols. Both the Bluetooth Specification and the IEEE Standard 802.15.6 include AKE protocols that use OOB channels. In this thesis, we designed two UECDH-based AKE protocols that use high bandwidth OOB channels: Protocol III-A and Protocol III-B. The protocols transmit the public keys and core values to compute the shared secret through OOB channels. As a result, the man-in-the-middle attacks and impersonation attacks to the protocols fail.

The advantage of high bandwidth OOB UECDH-based AKE protocols is that they do not rely on any pre-shared secret or any trusted third party. Therefore, they are much useful in pervasive computing and communication where the parties are unacquainted with each other. The limitation is that they require the communicating parties can establish high bandwidth OOB channels between them.

8.1.4 Low Bandwidth OOB UECDH-based AKE Protocols

Low bandwidth OOB channels, such as Display OOB channels, are easily to establish between two devices that both of them have a display for five-digit hash output. Therefore, to resist combinatorial attacks to short hash strings, commitment mechanisms are applied. Low bandwidth OOB UECDH-based AKE protocols are also included in the Bluetooth Specification and the IEEE Standard 802.15.6. In this thesis, we presented two low bandwidth OOB UECDH-based AKE protocols: Protocol IV-A and IV-B. To guarantee authentication, the two protocols use the displays to compare digests.

Similarly with the high bandwidth OOB UECDH-based AKE protocols, the advantage of low bandwidth OOB UECDH-based AKE protocols is removing reliance on pre-shared secrets or trusted third parties. The limitation is that they require both communicating devices have a display for five-digit number.

8.2 Future Work

8.2.1 Formal Verification of Protocols

Formal verification of security protocols is a highly active topic in the research community. It proves the correctness of protocols and avoids faults in designing security protocols. A number of formal methods and automatic tools have been developed by now, such as the Failures Divergences Refinement (FDR) [39, 38] which is a model checker for the process algebra Communicating Sequential Processes (CSP) [49], Burrows-Abadi-Needham (BAN) logic [19], Gong-Needham-Yahalom (GNY) logic [44] and so on.

Available formal methods cannot verify security protocols using OOB channels; and they are unavailable to verify some new features of security protocols such as the unbalanced computations. Therefore, it is valuable to extend current formal methods and the automatic tools.

8.2.2 Unbalancing Other AKE Protocols

In addition to the ECDH key exchange scheme, it is also valuable to unbalance computations in the identity (ID)-based AKE schemes. The ID-based cryptography is first introduced by Sharmir [91]. It guarantees authenticity by linking the public keys to the entities identities; therefore, secure communication can be established without a pre-shared secret or a PKI.

Since Boneh and Franklin [17] introduced the first ID-based encryption scheme from bilinear pairings, a number of ID-based AKE protocols [96, 95, 27, 104, 47] from bilinear pairings are proposed. The majority of available ID-based AKE protocols require the two parties execute equivalent bilinear pairing computation which is a time-consuming operation [25]. Therefore, transferring the bilinear parings from the limited party to the powerful one will significantly improve the overall performance of ID-based AKE protocols.

8.2.3 Applications

The AKE protocols with unbalanced computational requirements are suitable for a variety of applications. An emerging application in recent years is the blockchain-based IoT data management. Blockchain is an innovative technique for distributed computing introduced by the cryptographic currency bitcoin. It is anticipated to have enormous potential to manage numerous distributed data in IoT.

Currently, the security of blockchain-based IoT data management relies on TLS. However, devices in IoT have fairly different computational capabilities; and communications in IoT often take place between a limited end device and a more powerful gateway or server. Therefore, it is valuable to replace the TLS handshake protocols with the unbalanced AKE protocols. A meaningful future work is designing a more suitable security layer for blockchain-based IoT data management.

Appendix A

List of Acronyms

AKE Authenticated Key Exchange

BAN BurrowsCAbadiCNeedham

CA Certificate Authority

CSP Communicating Sequential Processes

DH Diffie-Hellman

ECC Elliptic Curve Cryptography

ECDH Elliptic Curve Diffie-Hellman

ECDHP Elliptic Curve Diffie-Hellman Problem

ECDLP Elliptic Curve Discrete Logarithm Problem

ECDSA Elliptic Curve Digital Signature Algorithm

FDR Failures Divergences Refinement

FIPS Federal Information Processing Standards

GNY Gong-Needham-Yahalom

HMAC Hash-Based Message Authentication Code

ID Identity

IEC International Electrotechnical Commission

IEEE Institute of Electrical and Electronics Engineers

IOS International Organization for Standardization

IoT Internet of Things

MAC Message Authentication Code

NIST National Institute of Standards and Technology

OOB Out of Band

PK Public Key

PKI Public Key Infrastructure

PW Password

QR Quick Response

RSA RivestCShamirCAdleman

S/MIME Secure/Multipurpose Internet Mail Extensions

SHA Secure Hash Algorithms

TCP Transmission Control Protocol

TLS Transport Layer Security

UECDH Unbalanced Elliptic Curve Diffie-Hellman

WBAN Wireless Body Area Network

Bibliography

[1] Fadel Adib and Dina Katabi. See through walls with wifi! *Proceedings of the ACM SIGCOMM*, pages 75–86, 2013.

[2] David Adrian, Karthikeyan Bhargavan, Zakir Durumeric, Pierrick Gaudry, Matthew Green, J. Alex Halderman, Nadia Heninger, Drew Springall, Emmanuel Thomé, Luke Valenta, Benjamin VanderSloot, Eric Wustrow, Santiago Zanella-Béguelin, and Paul Zimmermann. Imperfect forward secrecy: How Diffie-Hellman fails in practice. *In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 5–17, 2015.

[3] Bijan Ansari and M. Anwar Hasan. High-performance architecture of elliptic curve scalar multiplication. *IEEE Transactions on Computers*, 57(11):1443–1453, 2008.

[4] Mohamad Badra and Sherali Zeadally. Lightweight and efficient privacy-preserving data aggregation approach for the smart grid. *Ad Hoc Networks*, 64:32–40, 2017.

[5] P. Balasubramaniam and E. Karthikeyan. Elliptic curve scalar multiplication algorithm using complementary recoding. *Applied Mathematics and Computation*, 190(1):51–56, 2007.

[6] Dirk Balfanz, Diana K. Smetters, Paul Stewart, and H. Chi Wong. Talking to strangers: Authentication in ad-hoc wireless networks. *NDSS*, 2002.

[7] Paolo Baronti, Prashant Pillai, Vince WC Chook, Stefano Chessa, Alberto Gotta, and Y. Fun Hu. Wireless sensor networks: A survey on the state of the art and the 802.15. 4 and zigbee standards. *Computer communications*, 30(7):1655–1695, 2007.

[8] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. *Annual International Cryptology Conference*, pages 11–15, 1996.

[9] Mihir Bellare and Tadayoshi Kohno. Hash function balance and its impact on birthday attacks. *In International Conference on the Theory and Applications of Cryptographic Techniques*, pages 401–418, 2004.

[10] Michael J. Beller, Li-Fung Chang, and Yacov Yacobi. Privacy and authentication on a portable communications system. *In GLOBECOM' 91*, IEEE Press:1922–1927, 1991.

[11] Steven M Bellovin and Michael Merritt. Augmented encrypted key exchange: a password-based protocol secure against dictionary attacks and password file compromise. *In Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 244–250, 1993.

[12] Mehdi Bennis, Meryem Simsek, Andreas Czylwik, Walid Saad, Stefan Valentin, and Merouane Debbah. When cellular meets WiFi in wireless small cell networks. *IEEE communications magazine*, 51(6):44–50, 2013.

[13] Muhammad Nasir Mumtaz Bhutta, Haitham S. Cruickshank, and Zhili Sun. An efficient, scalable key transport scheme (ESKTS) for delay/disruption tolerant networks. *Wireless networks*, 20(6):1597–1609, 2014.

[14] Simon Blake-Wilson and Alfred Menezes. Entity authentication and authenticated key transport protocols employing asymmetric techniques. *International Workshop on Security Protocols*, pages 137–158, 1997.

[15] Simon Blake-Wilson and Alfred Menezes. Entity authentication and authenticated key transport protocols employing asymmetric techniques. *International Workshop on Security Protocols*, pages 137–158, 1997.

[16] Simon Blake-Wilson and Alfred Menezes. Authenticated Diffie-Hellman key agreement protocols. *International Workshop on Selected Areas in Cryptography*, pages 339–361, 1998.

[17] Dan Boneh and Matt Franklin. Identity-based encryption from the Weil pairing. *In Annual international cryptology conference, Springer*, pages 213–229, 2001.

[18] Colin Boyd and Juan González Nieto. On forward secrecy in one-round key exchange. *In IMA International Conference on Cryptography and Coding*, pages 451–468, 2011.

[19] Michael Burrows, Martin Abadi, and Roger Needham. A logic of authentication. *Proceedings of the Royal Society A*, pages 233–271, 1989.

[20] Mario Cagalj, Srdjan Capkun, and J-P. Hubaux. Key agreement in peer-to-peer wireless networks. *Proceedings of the IEEE*, 94(2):467–478, 2006.

[21] Jiaren Cai, Xin Huang, Jie Zhang, Jiawei Zhao, Yaxi Lei, Dawei Liu, and Xiaofeng Ma. A handshake protocol with unbalanced cost for wireless updating. *IEEE Access*, 6:18570–18581, 2018.

[22] Franco Callegati, Walter Cerroni, and Marco Ramilli. Man-in-the-middle attack to the https protocol. *IEEE Security & Privacy*, 7(1):78–81, 2009.

[23] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. *Advances in Cryptology-EUROCRYPT 2001*, pages 453–474, 2001.

[24] Huasong Cao, Victor Leung, Cupid Chow, and Henry Chan. Enabling technologies for wireless body area networks: A survey and outlook. *IEEE Communications Magazine*, 27(12), 2009.

[25] Xuefei Cao, Weidong Kou, and Xiaoni Du. A pairing-free identity-based authenticated key agreement protocol with minimal message exchanges. *Information Sciences*, 180(15):2895–2903, 2010.

[26] Kun-Long Chen, Yan-Ru Chen, Yuan-Pin Tsai, and Nanming Chen. A novel wireless multifunctional electronic current transformer based on zigbee-based communication. *IEEE Transactions on Smart Grid*, 8(4):1888–1897, 2017.

[27] Liqun Chen and Caroline Kudla. Identity based authenticated key agreement protocols from pairings. *In Computer Security Foundations Workshop, IEEE*, pages 219–233, 2003.

[28] Mario Collotta and Giovanni Pau. Bluetooth for internet of things: A fuzzy approach to improve power management in smart homes. *Computers & Electrical Engineering*, 44:137–152, 2015.

[29] Mauro Conti, Ali Dehghantanha, Katrin Franke, and Steve Watson. Internet of Things security and forensics: Challenges and opportunities. *Future Generation Computer Systems*, 78(2):544–546, 2018.

[30] Don Coppersmith. Another birthday attack. *In Conference on the Theory and Application of Cryptographic Techniques*, pages 14–17, 1985.

[31] Sadie Creese, Michael Goldsmith, Richard Harrison, Bill Roscoe, Paul Whittaker, and Irfan Zakiuddin. Exploiting empirical engagement in authentication protocol design. *International Conference on Security in Pervasive Computing*, pages 119–133, 2005.

[32] Cas Cremers and Michele Feltz. Beyond eCK: Perfect forward secrecy under actor compromise and ephemeral-key reveal. *In European Symposium on Research in Computer Security*, pages 734–751, 2012.

[33] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

[34] Whitfield Diffie, Paul C. van Oorschot, and Michael J. Wiener. Authentication and authenticated key exchange. *Designs, Codes and Cryptography*, 2:107–125, 1992.

[35] Danny Dolev and Andrew Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1993.

[36] Shahin Farahani. Zigbee wireless networks and transceivers. *Newnes*, 2011.

[37] Marc Fischlin, Felix Günther, Benedikt Schmidt, and Bogdan Warinschi. Key confirmation in key exchange: A formal treatment and implications for tls 1.3. *In Security and Privacy (SP)*, IEEE Symposium on:452–469, 2016.

[38] Thomas Gibson-Robinson, Philip Armstrong, Alexandre Boulgakov, and A.W. Roscoe. Failures divergences refinement (FDR) version 3. https://www.cs.ox.ac.uk/projects/fdr/, 2013.

[39] Thomas Gibson-Robinson, Philip Armstrong, Alexandre Boulgakov, and A.W. Roscoe. FDR3: A modern refinement checker for CSP. *Tools and Algorithms for the Construction and Analysis of Systems*, 8413:187–201, 2014.

[40] Marc Girault, Robert Cohen, and Mireille Campana. A generalized birthday attack. *In Workshop on the Theory and Application of of Cryptographic Techniques*, pages 129–156, 1998.

[41] Dieter Gollmann. What do we mean by entity authentication? *In IEEE Symposium on Security and Privacy*, IEEE Computer Society Press:46–54, 1996.

[42] Li Gong. A security risk of depending on synchronized clocks. *ACM Operating Systems Review*, 26(1):49–53, 1992.

[43] Li Gong. Variations on the themes of message freshness and replay. *In 6th IEEE Computer Security Foundations Workshop*, pages 131–136, 1993.

[44] Li Gong, Roger Needham, and Raphael Yahalom. Reasoning about belief in cryptographic protocols. *Research in Security and Privacy, 1990 IEEE Computer Society Symposium on*, 1990.

[45] Jaap C Haartsen. The bluetooth radio system. *IEEE personal communications*, 7(1):28–36, 2000.

[46] Darrel Hankerson, Alfred J. Menezes, and Scott Vanstone. Guide to elliptic curve cryptography. *Springer Science & Business Media*, 2006.

[47] Debiao He, Sherali Zeadally, Baowen Xu, and Xinyi Huang. An efficient identity-based conditional privacy-preserving authentication scheme for vehicular ad hoc networks. *IEEE Transactions on Information Forensics and Security*, 10(12):2681–2691, 2015.

[48] Robin Heydon. Bluetooth low energy: the developer's handbook. *Upper Saddle River: Prentice Hall*, 1, 2013.

[49] Charles Antony Richard Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–677, 1978.

[50] Russell Housley, Warwick Ford, William Polk, and David Solo. Internet x. 509 public key infrastructure certificate and crl profile. *RFC*, 2459, 1998.

[51] Xin Huang. Multi-channel security protocols in personal networks. *Doctoral dissertation, University of Oxford*, 2014.

[52] ISO. Information technology - security techniques-key management - part 2: Mechanisms using symmetric techniques. *ISO/IEC 11770-2*, 1996.

[53] Qi Jiang, Sherali Zeadally, Jianfeng Ma, and Debiao He. Lightweight three-factor authentication and key agreement protocol for internet-integrated wireless sensor networks. *IEEE Access*, 5:3376–3392, 2017.

[54] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ECDSA). *International journal of information security*, 1(1):36–63, 2001.

[55] Jonathan Katz and Yehuda Lindell. Introduction to modern cryptography. *CRC press*, 2014.

[56] Peter Kieseberg, Manuel Leithner, Martin Mulazzani, Lindsay Munroe, Sebastian Schrittwieser, Mayank Sinha, and Edgar Weippl. QR code security. *In Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia*, ACM:430–435, 2010.

[57] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987.

[58] Hugo Krawczyk, Ran Canetti, and Mihir Bellare. Hmac: Keyed-hashing for message authentication. *rfc2104*, 1997.

[59] Brian LaMacchia, Kristin Lauter, and Anton Mityagin. Stronger security of authenticated key exchange. *In International conference on provable security*, pages 1–16, 2007.

[60] Benoît Latré, Bart Braem, Ingrid Moerman, Chris Blondia, and Piet Demeester. A survey on wireless body area networks. *Wireless Networks*, 17(1):1–18, 2011.

[61] Kristin Lauter. The advantages of elliptic curve cryptography for wireless security. *IEEE Wireless communications*, 11(1):62–67, 2004.

[62] Huan-Bang Li and Kamya Yekeh Yazdandoost. Wireless body area network. *River publishers*, 2010.

[63] Xiong Li, Jianwei Niu, Md Zakirul Alam Bhuiyan, Fan Wu, Marimuthu Karuppiah, and Saru Kumari. A robust ecc based provable secure authentication arotocol with privacy protection for industrial internet of things. *IEEE Transactions on Industrial Informatics*, 2017.

[64] Jie Lin, Wei Yu, Nan Zhang, Xinyu Yang, Hanlin Zhang, and Wei Zhao. A survey on Internet of Things: Architecture, enabling technologies, security and privacy, and applications. *IEEE Internet of Things Journal*, 4(5):1125–1142, 2017.

[65] Zhen Ling, Junzhou Luo, Yiling Xu, Chao Gao, Kui Wu, and Xinwen Fu. Security vulnerabilities of Internet of Things: A case study of the Smart Plug system. *IEEE Internet of Things Journal*, 4(6):1899–1909, 2017.

[66] Gavin Lowe. Breaking and fixing the Needham-Schroeder public key protocol using FDR. *Tools and Algorithms for the Construction and Analysis of Systems*, pages 147–166, 1996.

[67] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. Handbook of applied cryptography. *CRC Press*, 1997.

[68] Victor S Miller. Use of elliptic curves in cryptography. *Conference on the theory and application of cryptographic techniques*, 1985.

[69] Dheerendra Mishra, Saru Kumari, Muhammad Khurram Khan, and Sourav Mukhopadhyay. An anonymous biometric based remote user authenticated key agreement scheme for multimedia systems. *International Journal of Communication Systems*, 30(1), 2017.

[70] Zeyad Mohammad, Yaw-Chung Chen, Chien-Lung Hsu, and Chi-Chun Lo. Cryptanalysis and enhancement of two-pass authenticated key agreement with key confirmation protocols. *IETE Technical Review*, 27(3):252–265, 2010.

[71] Kathleen Moriarty. The transport layer security (TLS) protocol version 1.3. 2018.

[72] Arsalan Mosenia and Niraj K. Jha. A comprehensive study of security of Internet-of-Things. *IEEE Transactions on Emerging Topics in Computing*, 5(4):586–602, 2017.

[73] Junghyun Nam, Juryon Paik, H-K. Kang, Ung Mo Kim, and Dongho Won. An off-line dictionary attack on a simple three-party key exchange protocol. *IEEE Communications Letters*, 13(3):205–207, 2009.

[74] Arvind Narayanan and Vitaly Shmatikov. Fast dictionary attacks on passwords using time-space tradeoff. *In Proceedings of the 12th ACM conference on Computer and communications security*, pages 364–372, 2005.

[75] Vankamamidi S. Naresh and Nistala VES Murthy. Provably secure group key agreement protocol based on ECDH with integrated signature. *Security and Communication Networks*, 9(10):1085–1102, 2016.

[76] Roger Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1999.

[77] B. Clifford Neuman and Theodore Ts'o. Kerberos: An authentication service for computer networks. *IEEE Communications magazine*, 32(9):33–38, 1994.

[78] Long Hoang Nguyen and A. William Roscoe. Authentication protocols based on low-bandwidth unspoofable channels: a comparative survey. *Journal of Computer Security*, 19(1):139–201, 2011.

[79] National Institute of Standards and Technology (NIST) Federal Information Processing Standards (FIPS). Recommended elliptic curves for federal government use. 1999.

[80] National Institute of Standards and Technology (NIST) Federal Information Processing Standards (FIPS). Digital signature standard (DSS), FIPS PUB 186-4. 2013.

[81] Dong-Sik Oh, Bong-Han Kim, and Jae-Kwang Lee. A study on authentication system using QR code for mobile cloud computing environment. *In Future Information Technology*, Springer:500–507, 2011.

[82] Chang-Seop Park. On certificate-based security protocols for wireless mobile communication systems. *IEEE Network*, 11(5):50–55, 1997.

[83] DongGook Park, Colin Boyd, and Sang-Jae Moon. Forward secrecy and its application to future mobile communications security. *In International Workshop on Public Key Cryptography*, pages 433–445, 2000.

[84] Philips. Philips, 2015 annual report. http://www.philips.com/corporate/resources/annualresults/2015/PhilipsFullAnnualReport2015English.pdf, 2016.

[85] Bluetooth SIG Proprietary. Bluetooth core specification version 5.0. 2016.

[86] Eyal Ronen, Adi Shamir, Achi-Or Weingarten, and Colin O'Flynn. IoT goes nuclear: Creating a zigbee chain reaction. *Security and Privacy (SP)*, 2017 IEEE Symposium:195–212, 2017.

[87] Marwa Salayma, Ahmed Al-Dubai, Imed Romdhani, and Youssef Nasser. Wireless body area network (WBAN): A survey on reliability, fault tolerance, and technologies coexistence. *ACM Computing Surveys (CSUR)*, 50(1):3, 2017.

[88] Arto Salomaa. Public-key cryptography. *Springer Science & Business Media*, 2013.

[89] Mahadev Satyanarayanan. Integrating security in a large distributed system. *ACM Transactions on Computer Systems (TOCS)*, 7(3):247–280, 1989.

[90] Savio Sciancalepore, Giuseppe Piro, Gennaro Boggia, and Giuseppe Bianchi. Public key authentication and key agreement in IoT devices with minimal airtime consumption. *IEEE Embedded Systems Letters*, 9(1):1–4, 2017.

[91] Adi Shamir. Identity-based cryptosystems and signature schemes. *In Workshop on the theory and application of cryptographic techniques, Springer*, pages 47–53, 1984.

[92] Jian Shen, Shaohua Chang, Jun Shen, Qi Liu, and Xingming Sun. A lightweight multi-layer authentication protocol for wireless body area networks. *Future Generation Computer Systems*, 78:956–963, 2018.

[93] Jian Shen, Tianqi Zhou, Debiao He, Yuexin Zhang, Xingming Sun, and Yang Xiang. Block design-based key agreement for group data sharing in cloud computing. *IEEE Transactions on Dependable and Secure Computing*, 2017.

[94] Jian Shen, Tianqi Zhou, Fushan Wei, Xingming Sun, and Yang Xiang. Privacy-preserving and lightweight key agreement protocol for V2G in the social Internet of Things. *IEEE Internet of Things Journal*, 2017.

[95] Kyungah Shim. Id-based authenticated key agreement protocol based on Weil pairing. *Electronics Letters*, 39(8):653–654, 2003.

[96] Nigel P Smart. Identity-based authenticated key agreement protocol based on weil pairing. *Electronics letters*, 38(13):630–632, 2002.

[97] IEEE Computer Society. IEEE standard 802.15.6: Wireless body area networks. 2012.

[98] Daniel J. Solove and Paul Schwartz. Information privacy law. *Wolters Kluwer Law & Business*, 2014.

[99] Paul Syverson and Paul C. van Oorschot. On unifying some cryptographic protocol logics. *In IEEE Symposium on Research in Security and Privacy*, IEEE Computer Society Press:14–28, 1994.

[100] Ding Wang and Ping Wang. Offline dictionary attack on password authentication schemes using smart cards. *In Information Security*, pages 221–237, 2015.

[101] Hao Wang, Daqing Zhang, Yasha Wang, Junyi Ma, Yuxiang Wang, and Shengjie Li. Rt-fall: A real-time and contactless fall detection system with commodity WiFi devices. *IEEE Transactions on Mobile Computing*, 16(2):511–526, 2017.

[102] Ford Long Wong and Frank Stajano. Multichannel security protocols. *IEEE Pervasive Computing*, 6(4), 2007.

[103] Kwok-Wo Wong, Edward CW Lee, L. M. Cheng, and Xiaofeng Liao. Fast elliptic scalar multiplication using new double-base chain and point halving. *Applied mathematics and computation*, 183(2):1000–1007, 2006.

[104] Libing Wu, Yubo Zhang, Yong Xie, Abdulhameed Alelaiw, and Jian Shen. An efficient and secure identity-based authentication and key agreement protocol with user anonymity for mobile devices. *Wireless Personal Communications*, 94(4):3371–3387, 2017.

[105] Nian Xue, Xin Huang, and Jie Zhang. Poster: a framework for iot reprogramming. *International Conference on Security and Privacy in Communication Systems*, pages 751–754, 2016.

[106] Nian Xue, Xin Huang, and Jie Zhang. S^2Net: a security framework for software defined intelligent building networks. *Trustcom/BigDataSE/ISPA,2016 IEEE*, pages 654–661, 2016.

[107] Zheng Yuan, Wei Wang, Keting Jia, Guangwu Xu, and Xiaoyun Wang. New birthday attacks on some MACs based on block ciphers. *In Advances in Cryptology-CRYPTO 2009*, pages 209–230, 2009.

[108] Sofia Zebboudj, Feriel Cherifi, Mohamed Mohammedi, and Mawloud Omar. Secure and efficient ECG-based authentication scheme for medical body area sensor networks. *Smart Health*, 3:75–84, 2017.

[109] Jie Zhang, Xin Huang, Paul Craig, Alan Marshall, and Dawei Liu. An improved protocol for the password authenticated association of IEEE 802.15. 6 standard that alleviates computational burden on the node. *Symmetry*, 8(11):131, 2016.

[110] Jie Zhang, Nian Xue, and Xin Huang. A secure system for pervasive social network-based healthcare. *IEEE Access*, 4:9239–9250, 2016.