A Thesis for the Degree of Master

A Study on Key Agreement Protocol based on Diffie-Hellman Problem

Boyeon Song

School of Engineering

Information and Communications University

2001

A Study on Key Agreement Protocol based on Diffie-Hellman Problem

A Study on Key Agreement Protocol based on Diffie-Hellman Problem

Advisor : Professor Kwangjo Kim

by

Boyeon Song
School of Engineering
Information and Communications University

A thesis submitted to the faculty of Information and Communications University in partial fulfillment of the requirements for the degree of Master of Science in the School of Engineering

Taejon, Korea
Dec. 27. 2000
Approved by

(signed)

Professor Kwangjo Kim
Major Advisor

A Study on Key Agreement Protocol based on Diffie-Hellman Problem

Boyeon Song

I certify that this work has passed the scholastic standards required by the Information and Communications University as a thesis for the degree of Master

Dec. 27. 2000

Approved:

Chairman of the Committee
Kwangjo Kim, Associate Professor
School of Engineering

Committee Member
Chulsoo Lee, Invited Professor
School of Engineering

Committee Member
Junghee Cheon, Assistant Professor
School of Engineering

M.S. Boyeon Song
1998063

A Study on Key Agreement Protocol based on Diffie-Hellman Problem

School of Engineering, 2001, 55p.
Major Advisor : Prof. Kwangjo Kim.
Text in English

Abstract

This paper deals with key establishment protocol needed prior to cryptographic communication. First, the existing key establishment protocols are classified according to key control, cryptographic techniques used, trusted third party, time-variant parameters and the number of passes. And then their properties are compared in four groups; key transport protocols on symmetric techniques, key transport protocols on asymmetric techniques, key agreement protocols on symmetric techniques, and key agreement protocols on asymmetric techniques.

Next, new key agreement protocols are proposed, which are based on Diffie-Hellman problem and is working over elliptic curve group in the setting of asymmetric techniques. Three proposed protocols are the followings; Protocol 4 is an AK (Authenticated Key agreement) protocol, Protocol 5 is an AKC (Authenticated Key agreement with key Confirmation) protocol, and Protocol 6 is an unilateral AKC protocol. And then these protocols are compared with the related work (*i.e.* the MTI/A0, Unified Model, MQV, and Protocol IIA) in security and performance.

Contents

List of Tables

List of Abbreviations

AK Authenticated Key agreement

AKC Authenticated Key agreement with key Confirmation

Cl key Control

CA Certification Authority

EA Entity Authentication

ECDHP Elliptic Curve Diffie-Hellman Problem

ECDHS Elliptic Curve Diffie-Hellman Scheme

ECDLP Elliptic Curve Discrete Logarithm Problem

EKA Explicit Key Authentication

Fr key Freshness

FS Forward Secrecy

IKA Implicit Key Authentication

KA Key Authentication (key integrity and key origin authentication)

KAA Key Agreement protocol on Asymmetric techniques

KAS Key Agreement protocol on Symmetric techniques

KC Key Confirmation

K-CI Key-Compromise Impersonation

KDC Key Distribution Center

kdf key derivation function

K-KS Known-Key Security

KTA Key Transport protocol on Asymmetric techniques

KTC Key Translation Center

KTS Key Transport protocol on Symmetric techniques

MAC Message Authentication Code

NP the Number of Pass

PKO Public Key Operation

Sig Signature

TTP Trusted Third Party

TVP Time Variant Parameter

UK-S Unknown Key-Share

List of Notations

$A \rightarrow B : Z$ A sends Z to B

$A \Rightarrow B : Z$ A sends Z to B over a secure channel

$A : Z$ A executes Z

$a,\ b$ entity A and B's static private keys; $a, b \in_R [1, q-1\ ]$ or two field elements in $\mathbf{F}_q$ defining the equation of the elliptic curve

c cofactor

$Cert_X$ entity X's public key certificate

$D_K(Z)$ decipherment of data Z with a symmetric algorithm over the key K

E an elliptic curve

$E_K(Z)$ encipherment of data Z with a symmetric algorithm over the key K

f key generating function

$\mathbf{F}_q$ finite field of characteristic p

g an element of order q in Z_p^* which is shared public by all the entities

H cryptographic hash function

$\mathcal{H}_1, \mathcal{H}_2$ independent hash function

I_X distinguishing identifier of entity X

ID_X X's identity information

K secret key for a symmetric cryptosystem
or shared secret over elliptic curve group

K_X secret symmetric session key chosen by entity X

K_{XY} secret key associated with the entities X and Y

L validity period (life time)

$MAC_K(Z)$ message authentication code on data Z over the key K

n the order of the point P

$\mathcal{O}$ a point at infinity

P a base point of the elliptic curve E

p a large prime or characteristic

$P_X(Z)$ entity X's public encipherment transformation of data Z

q a prime divisor of $p-1$ or a field size

r, m random number

R_X X's ephemeral public key

r_X random number issued by entity X or X's ephemeral private key

(R_X, r_X) X's ephemeral key pair

$S_X(Z)$ entity X's private signature transformation on data Z

T distinguishing identifier of TTP

t_X/n_X time stamp or a sequence number issued by entity X

TVP_X time variant parameter issued by entity X

W_X X's static public key

w_X X's static private key

(W_X, w_X) X's static key pair

X' something which has different value from X but the same property as X

X, Y concatenation of data items X and Y in that order

$x,\ y$ entity A and B's ephemeral private keys; $x, y \in_R [1, q-1]$

$x_p,\ y_p$ two field elements in $\mathbf{F}_q$ defining a base point $P = (x_p, y_p)$

$z_A,\ z_B$ entity A and B's static public keys; $z_A = g^a\ mod\ p,\ z_B = g^b\ mod\ p$

I. Introduction

1.1 Overview

Cryptography is the one of the most important methods for transmitting information secret and for protecting to be tampered with information via unsecured network. Such secure communication needs some cryptographic algorithms, which require any key with information being transmitted as its inputs. The key must be established commonly between communicating entities prior to communications for security services on cryptographic mechanisms like data encipherment, message authentication, and digital signature. *Key establishment* protocol is a process whereby a shared secret key becomes available to two or more entities participating in the protocol, for subsequent cryptographic use[20]. The shared secret key may subsequently be used to achieve some cryptographic goal such as confidentiality or data integrity. Key establishment protocol is broadly divided into key transport and key agreement protocols[20]. In *key transport* protocols, a key is created by one entity and securely transmitted to the other entity. In *key agreement* protocols, both entities contribute information to generate the shared secret key[4]. Key establishment protocol employs symmetric or asymmetric key cryptography. A *symmetric* cryptographic system is a system involving two transformations - one for the initiator and one for the responder - both of which make use of the same secret key[20]. In this system the two entities previously possess common secret information, so the key management problem is a crucial issue. An *asymmetric* cryptographic system is a system involving two related transformations - one defined by a public key (the public transformation), and another defined by a private key (the private transformation) - with the property that it is computationally infeasible to determine the private transformation from the public transformation[20]. In this system the two entities share only public information that has been authenticated[5].

Time-variant parameters serve to distinguish one protocol instance from another. The values are used no more than once for the same purpose, which typically serve to prevent replay. Three main classes of time-variant parameters are timestamps, sequence numbers, and random numbers. Timestamp-based protocols offer the advantage of fewer message, but they require that the entities' time clocks are both synchronized and secured. Timestamps in protocols may be replaced by a random number challenge, then a return pass is needed in the protocols[20].

1.2 Research Objectives

This paper first classifies the existing key establishment protocols surveyed so far, describes the briefs of the protocols, and compares them according to some requirements needed for secure key establishment. And then two-entity authenticated key agreement protocols are focused upon. They are based on Diffie-Hellman key agreement working over an elliptic curve group in the asymmetric (public-key) setting. The new three protocols are proposed. Firstly, the proposed protocol is an authenticated key agreement (AK) protocol, which provides more desirable security attributes than other AK protocols discussed. The second protocol is an authenticated key agreement with key confirmation (AKC) protocol. It is derived from the AK protocol by using the Message Authentication Code (MAC) algorithms and provides all security attributes discussed this paper. Finally, a two-pass unilateral AKC protocol is designed to provide mutual implicit key authentication and unilateral explicit key authentication, and known-key security, forward secrecy, key-compromise impersonation, and unknown key-share attributes discussed in [4, 5].

1.3 Outline

The remainder of this paper is organized as follows. Chapter 2 presents the preliminary of key establishment protocol; Section 2.1 explains general models for key establishment environment, Section 2.2 describes requirements of key establishment, and Section 2.3 introduces the ingredients of elliptic curve cryptosystem.

Chapter 3 classifies the existing key establishment protocols; Sections 3.1 and 3.2 describe the briefs of key transport protocols and key agreement protocols, respectively, and in Section 3.3, the key establishment protocols are compared according to the requirements described above.

Chapter 4 proposes new key agreement protocols; Section 4.1 presents the related AK protocols over an elliptic curve group, Section 4.2 describes new AK, AKC, and two-pass unilateral AKC protocol, and Section 4.3 compares them with the related works - the MTI/A0, Unified Model, MQV, and Protocol IIA.

Finally, Chapter 5 makes concluding remarks and further works.

II. Preliminary

2.1 Models for Key Establishment Environment

Key establishment protocols are influenced by the nature of the communication links, the trust relationships involved and the cryptographic techniques used; the entities may either communicate directly or indirectly, may belong the same or different security domains, and may or may not use the services of a trusted authority. The environments for the establishment of keys are illustrated[27].

2.1.1 Point-to-point key establishment

Point-to-point key establishment is the basic mechanism of every key establishment scheme. It is required that the entities have to share a key so that further keys may be established directly between the entities[10, 27].

- When the entities use symmetric techniques,

  the two entities involved already share a key that can be used to protect the keying material to be established.

- When the entities use asymmetric techniques,

  each of the entities has a public key with its associated private key, and the authenticated public key is known to the other entity; for data integrity or data origin authentication, the recipient requires the sender's corresponding public key certificate, and for confidentiality, the sender requires a public key certificate of the intended recipient.

2.1.2 Key establishment within one domain

Key establishment within one domain employs a single *Trusted Third Party* (TTP) for the entire system. This authority may offer key management services such as the generation, certification, distribution or translation of keying material[10, 27].

- When the entities use symmetric techniques,

  a sender and a receiver are required to share keys with the authority. The TTP then functions as one of two ways.

  - *Key Distribution Center* (KDC) : One entity asks a KDC to generate a key for subsequent distribution. The KDC then either distributes the key directly to both entities, or it sends it back to the initiator, who forwards it to the other entity.
  - *Key Translation Center* (KTC) : One entity generates a key and sends it to a KTC. The KTC may either forward directly it to the other entity, or send it back to the first entity, who forwards it to the second entity.

- When the entities use asymmetric techniques,

  each entity may need to contact its authority to get an appropriate public key certificate. Then the TTP typically is called *Certification Authority* (CA).

Key establishment for large systems usually is organized in a hierarchical way. *Security Domain* is a group of entities served by one authority[10, 27].

2.1.3 Key establishment between domains

This model involves entities belonging to different security domains which share at least one cryptographic technique. Each security domain has its own security authority[10].

- When the entities use symmetric techniques and do not have access to a common directory service that offers public key certificates,

  at least one of them has to contact its authority to receive a secret key for communication. The authorities then establish a common secret key to be used by the entities. This key may be distributed by one authority to both entities using the other authority as a KTC, or via one of the entities which is responsible for forwarding the key to the other entity.

- When the entities use asymmetric techniques,

  each entity shall contact its respective authority to get its partner's public key certificate. The authorities of A and B may exchange the public key certificates of entities A and B and forward them to A and B, respectively.

In situations where the authorities of A and B neither have a mutual trust relationship nor a direct communications path, they have to involve one of more additional TTPs until a chain of trust is established.

2.2 Requirements of Key Establishment Protocols

The goal of any authenticated key establishment protocol is to establish keying data. Ideally, the established key should have precisely the same attributes as a key established face-to-face. However, it is not an easy task to identify the precise security requirements of authenticated key establishment. Several concrete security and performance attributes have been identified as desirable[4].

2.2.1 Fundamental security attributes

The fundamental security goals of key establishment protocols are said to be implicit key authentication and explicit key authentication. Let A and B be two honest entities, *i.e.*, legitimate entities who execute the steps of a protocol correctly.

- **Implicit Key Authentication (IKA)** : Informally speaking, a key establishment protocol is said to provide implicit key authentication (of B to A) if entity A is assured that no other entity aside from a specifically identified second entity B can possibly learn the value of a particular secret key.

  A key agreement protocol which provides implicit key authentication to both participating entities is called an *authenticated key agreement (AK)* protocol.

- **Key Confirmation (KC)** : A key establishment protocol is said to provide *key confirmation* (of B to A) if entity A is assured that the second entity B actually has possession of a particular secret key.

- **Explicit Key Authentication (EKA)** : If both implicit key authentication and key confirmation (of B to A) are provided, the key establishment protocol is said to provide *explicit key authentication (EKA)* (of B to A).

  A key agreement protocol which provides explicit key authentication to both entities is called an *authenticated key agreement with key confirmation (AKC)* protocol[4, 15, 20].

- **Entity Authentication (EA)** : A key establishment protocol is said to provide *entity authentication* (of B to A) if entity A is assured of the identity of the second entity B involved in a protocol, and that B has actually participated.

2.2.2 Desirable security attributes of AK and AKC protocols

Desirable security attributes of AK and AKC protocols are as follows[4, 15, 20].

- **Known-Key Security (K-KS)** : A protocol should still achieve its goal in the face of an adversary who has learned some other session keys. The session key is a unique secret key which each run of a key agreement protocol between A and B should produce.

- **Forward Secrecy (FS)** : If static private keys of one or more entities are compromised, the secrecy of previous session keys is not affected.

- **Key-Compromise Impersonation (K-CI)** : When A's static private key is compromised, it may be desirable that this event does not enable an adversary to impersonate other entities to A.

- **Unknown Key-Share (UK-S)** : Entity B cannot be coerced into sharing a key with entity A without B's knowledge, *i.e.*, when B believes the key is shared with some entity $C \neq A$, and A correctly believes the key is shared with B.

- **Key Freshness** : The established keying material is new, as opposed to the reuse of old keying material.

- **Key Control** : It is the ability to choose the key or the parameters used in the key computation. Neither entity should be able to force the session key to a preselected value.

2.2.3 Desirable performance attributes of AK and AKC protocols

Performance attributes of AK and AKC protocols include;

- **Minimal number of passes** : the number of messages exchanged
- **Low communication overhead** : total number of bits transmitted
- **Low computation overhead** : total number of arithmetical operations required
- **Possibility of precomputations** : to minimize on-line computational overhead

2.3 Elliptic Curve Cryptosystem

Elliptic curve groups give an advantage over other groups because the former groups offer smaller key sizes and faster computation times with equivalent security as the latter. A protocol in any finite group, in which the discrete logarithm problem appears intractable, can be modified to work in the setting of the group of points on an elliptic curve defined over a finite field. This section describes the elliptic curve parameters that are common to participating entities in the protocols (*i.e.*, the *domain parameters*), and the key pairs of each entity. These are the same one used in [15].

2.3.1 Domain parameters

The domain parameters consist of a suitably chosen elliptic curve E defined over a finite field $\mathbf{F}_q$ of characteristic p, and a *base point* $P \in E(\mathbf{F}_q)$[15].

1. a field size q, where q is a prime power (in practice, either $q = p$, an old prime, or $q = 2^m$);

2. an indication FR (field representation) of the representation used for the elements of $\mathbf{F}_q$;

3. two field elements a and b in $\mathbf{F}_q$ which define the equation of the elliptic curve E over $\mathbf{F}_q$;

4. two field elements x_p and y_p in $\mathbf{F}_q$ which define a finite point $P = (x_p, y_p)$ of prime order in $E(\mathbf{F}_q)$;

5. the order n of the point P; and

6. the cofactor $c = \#E(\mathbf{F}_q)/n$.

A set of domain parameters (q, FR, a, b, P, n, c) can be verified to meet the above requirements (see [15] for more details). This process is called *domain parameter validation*.

2.3.2 Key pairs

Given a valid set of domain parameters (q, FR, a, b, P, n, c), an entity's *private key* is an integer d selected at random from the interval $[1, n-1]$, and its *public key* is the elliptic curve point $Q = dP$. The key pair is (Q, d). For the protocols described in this paper, each entity has two key pairs: a *static* or *long-term* key pair (which is bound to the entity for a certain period of time), and *ephemeral* or *short-term* key pair (which is generated for each run of the protocol)[15]. A's static key pair and ephemeral key pair are denoted (W_A, w_A) and (R_A, r_A) respectively.

For the protocols over elliptic curve groups of this paper, we will assume that static public keys are exchanged via certificates. $Cert_A$

denotes A's public-key certificate, containing a string of information that uniquely identifies A (such as A's name and address), her static public key W_A, and a certifying authority (CA)'s signature over this information. To avoid a potential unknown key-share attack, the CA should verify that A possesses the private key w_A corresponding to her static public key W_A[15].

2.3.3 Public-key validation

Before using an entity's purported public key Q, it is prudent to verify that it possesses the supposed arithmetic properties - namely that Q be a finite point in the subgroup generated by P[15]. This process is called *public-key validation*[12]. A purported public key $Q = (x_Q, y_Q)$ can be validated by verifying that:

1. Q is not equal to $\mathcal{O}$ which denotes a point at infinity;
2. x_Q and y_Q are elements in the field $\mathbf{F}_q$;
3. Q satisfies the defining equation of E; and
4. $nQ = \mathcal{O}$.

The computationally expensive operation in public-key validation is the scalar multiplication in step 4. To reduce this burden, step 4 can be omitted during key validation. It is called *embedded public-key validation*[15]. For ease of explanation, public key validation is omitted from the descriptions of protocols of this paper.

III. The Existing Key Establishment Protocols

This chapter shortly describes key establishment protocols, which are proposed so far or constructed newly in this paper. The protocols are divided widely into two category, key transport protocol and key agreement protocol, and were subdivided according to cryptographic techniques used, TTP, time-variant parameters and the number of passes.

3.1 Assumption

For the implementation of the protocols specified in this paper[28, 29], it is assumed that

- A is a initiator and B is a responder.
- the entities are aware of each other's claimed identities, which may be achieved by the inclusion of identifiers information exchanged between the two entities, or be apparent from the context of the use of the mechanism.
- when the entities use symmetric techniques,
  - Point-to-Point
    a key K_{AB} is shared by A and B.
  - TTP (KDC, KTC)
    A and B respectively share secret keys K_{AT} and K_{BT} with a T, which shall be able to generate or otherwise acquire a key K.

- when the entities use asymmetric techniques,
  - the entity who registers a public key surely is possession of the corresponding private key.
  - the authenticated public keys of entities A and B, z_A and z_B, are known between A and B.

In practical implementations of key establishment protocols the key data may be needed to process further prior to being used for cryptographic communication. The protocols may contain optional data, depending upon the specific application, like message authentication.

3.2 Key Transport Protocol

Key Transport is the process that one entity transfers securely a chosen secret key to another entity.

3.2.1 Key transport protocol on symmetric techniques

This section presents the briefs of key transport protocols based on symmetric techniques. They are subdivided into protocols using three environments models: Point to Point, KDC and KTC. A point-to-point environment exists when two entities already share a key that can establish further keys. If two entities wish to communicate with each other using only symmetric techniques but do not currently share such a key, they shall make use of a KDC or KTC. The inclusion of a MAC over K in messages exchanged between entities may ensure key confirmation of the other entities.

Point-to-Point

1. Timestamp-based protocol

   - **KTS 1: Mechanism 1 of [28]**

     $A \longrightarrow B \ \ : \ TVP$

     $K = f(K_{AB}, TVP)$

   - **KTS 2: Mechanism 2 of [28]**

     $A \longrightarrow B \ \ : \ E_{K_{AB}}(K)$

   - **KTS 3: Mechanism 3 of [28]**

     $A \longrightarrow B \ \ : \ E_{K_{AB}}(t/n, I_B, K)$

2. Random number-based protocol

   - **KTS 4: Mechanism 4 of [28]**

     $A \longrightarrow B \ \ : \ r_A$

     $A \longleftarrow B \ \ : \ E_{K_{AB}}(r_A, I_A, K)$

   - **KTS 5-1: Authenticated Key Exchange Protocol 1 [20]**

     $A \longrightarrow B \ \ : \ r_A$

     $A \longleftarrow B \ \ : \ I_B, I_A, r_B, r_A, r'_B, K \oplus f(r'_B, K'_{AB}),$
     $\qquad\qquad\quad MAC_{K_{AB}}(I_B, I_A, r_B, r_A, r'_B, K \oplus f(r'_B, K'_{AB}))$

     $A \longrightarrow B \ \ : \ I_A, r_B, MAC_{K_{AB}}(I_A, r_B)$

     $K = f(r_B, K'_{AB})$

   - **KTS 5-2: Authenticated Key Exchange Protocol 2 [20]**

     $A \longrightarrow B \ \ : \ r_A$

     $A \longleftarrow B \ \ : \ I_B, I_A, r_B, r_A, MAC_{K_{AB}}(I_B, I_A, r_B, r_A)$

     $A \longrightarrow B \ \ : \ I_A, r_B, MAC_{K_{AB}}(I_A, r_B)$

     $K = f(r_B, K'_{AB})$

3. Protocol without any shared key between entities

   - **KTS 6: Shamir's no-key protocol**
     $A \longrightarrow B \;\;:\; K^{r_A}\ mod\ p$
     $A \longleftarrow B \;\;:\; (K^{r_A})^{r_B}\ mod\ p$
     $A \longrightarrow B \;\;:\; (K^{r_A r_B})^{{r_A}^{-1}}\ mod\ p$
     $B:\; K = (K^{r_B})^{{r_B}^{-1}}\ mod\ p$

TTP : KDC

One of the two entities requests a key K to a KDC for later distribution to the other entity. Then the KDC generates the key K and either sends a message containing K to only the requesting entity, or sends messages directly to both entities, respectively. In the former, the message is protected by a key shared with the first entity and contains a second message protected by a key shared between the KDC and the second entity, while in the latter the messages are protected using the shared keys between the KDC and each communicating entity.

1. **Indirect transmission**
   One entity asks a KDC to generate a key. The KDC then sends the key back to the entity, who forwards it to other entity.

   (a) *A's key asking*
   A asks the KDC to generate a key.

   - **KTS 7: Mechanism 7 of [28]**
     $A \longrightarrow T \;\;:\; I_B$
     $A \longleftarrow T \;\;:\; E_{K_{AT}}(K, I_B), E_{K_{BT}}(K, I_A)$
     $A \longrightarrow B \;\;:\; E_{K_{BT}}(K, I_A)$

- **KTS 8-1: Mechanism 8 of [28]**

$A \longrightarrow T \;:\; TVP_A, I_B$

$A \longleftarrow T \;:\; E_{K_{AT}}(TVP_A, K, I_B), E_{K_{BT}}(t_T/n_T, K, I_A)$

$A \longrightarrow B \;:\; E_{K_{BT}}(t_T/n_T, K, I_A), E_K(t_A/n_A, I_B)$

$A \longleftarrow B \;:\; E_K(t_B/n_B, I_A)$

- **KTS 8-2: Basic Kerberos protocol [20]**

$A \longrightarrow T \;:\; r_A, I_A, I_B$

$A \longleftarrow T \;:\; E_{K_{AT}}(r_A, K, I_B, L), E_{K_{BT}}(K, I_A, L)$

$A \longrightarrow B \;:\; E_{K_{BT}}(K, I_A, L), E_K(t_A, I_A)$

$A \longleftarrow B \;:\; E_K(t_A, I_B)$

- **KTS 9: Needham-Schroeder shared-key protocol [20]**

$A \longrightarrow T \;:\; I_A, I_B, r_A$

$A \longleftarrow T \;:\; E_{K_{AT}}(r_A, I_B, K, E_{K_{BT}}(K, I_A))$

$A \longrightarrow B \;:\; E_{K_{BT}}(K, I_A)$

$A \longleftarrow B \;:\; E_K(r_B)$

$A \longrightarrow B \;:\; E_K(r_B - 1)$

(b) *B's key asking*

A requests B to communicate each other. Then B asks a KDC to generate a key.

- **KTS 10: Otway-Rees protocol [20]**

$A \longrightarrow B \;:\; m, I_A, I_B, E_{K_{AT}}(r_A, m, I_A, I_B)$

$B \longrightarrow T \;:\; m, I_A, I_B, E_{K_{AT}}(r_A, m, I_A, I_B), E_{K_{BT}}(r_B, m, I_A, I_B)$

$B \longleftarrow T \;:\; E_{K_{AT}}(r_A, K), E_{K_{BT}}(r_B, K)$

$A \longleftarrow B \;:\; E_{K_{AT}}(r_A, K)$

- **KTS 11: Mechanism 9 of [28]**

$A \longrightarrow B \;:\; r_A$

$B \longrightarrow T \;:\; r_A, r_B, I_A$

$B \longleftarrow T \;:\; E_{K_{AT}}(r_A, K, I_B), E_{K_{BT}}(r_B, K, I_A)$

$A \longleftarrow B \;:\; E_{K_{AT}}(r_A, K, I_B), E_K(r'_B, r_A)$

$A \longrightarrow B \;:\; E_K(r_A, r'_B)$

2. **Direct transmission**

   When one entity asks a KDC to generate a key, the KDC distributes the key to both entities.

   (a) *A's key asking*

   A asks a KDC to generate a key.

   - **KTS 12: Mechanism 10 of [28]**

     $A \longrightarrow T \;:\; E_{K_{AT}}(t_A/n_A, I_B)$

     $A \longleftarrow T \;:\; E_{K_{AT}}(t_T/n_T, K, I_B)$

     $T \longrightarrow B \;:\; E_{K_{BT}}(t'_T/n'_T, K, I_A)$

   (b) *B's key asking*

   A requests B to communicate each other. Then B asks a KDC to generate a key.

   - **KTS 13: Capable Protocol 1**

     $A \longrightarrow B \;:\; r_A$

     $B \longrightarrow T \;:\; r_A, r_B, I_A$

     $B \longleftarrow T \;:\; E_{K_{BT}}(r_B, K, I_A)$

     $A \longleftarrow T \;:\; E_{K_{AT}}(r_A, K, I_B)$

TTP : KTC

One of the entities sends a key K to a KTC. K is enciphered with a key shared between the KTC and the entity. The KTC deciphers the key K and enciphers it again with a key shared between the KTC and the other entity. And then the KTC either sends it back to the first entity so to forward it to the second entity, or forwards directly the key to the second entity.

1. **Indirect transmission**

   One entity generates a key and sends it to a KTC. The KTC then sends it back to the entity, who forwards it to the other entity.

(a) *A's key generating*

A generates a key and sends it to a KTC.

- **KTS 14: Mechanism 11 of [28]**

$$\begin{aligned} A \longrightarrow T &: E_{K_{AT}}(I_B, K) \\ A \longleftarrow T &: E_{K_{BT}}(K, I_A) \\ A \longrightarrow B &: E_{K_{BT}}(K, I_A) \end{aligned}$$

- **KTS 15: Mechanism 12 of [28]**

$$\begin{aligned} A \longrightarrow T &: E_{K_{AT}}(TVP_A, I_B, K) \\ A \longleftarrow T &: E_{K_{AT}}(TVP_A, I_B), E_{K_{BT}}(t_T/n_T, K, I_A) \\ A \longrightarrow B &: E_{K_{BT}}(t_T/n_T, K, I_A), E_K(t_A/n_A, I_B) \\ A \longleftarrow B &: E_K(t_B/n_B, I_A) \end{aligned}$$

(b) *B's key generating*

A asks B to communicate each other. Then B generates a key and sends it to a KTC.

- **KTS 16: Capable Protocol 2**

$$\begin{aligned} A \longrightarrow B &: r_A \\ B \longrightarrow T &: E_{K_{BT}}(r_A, r_B, I_A, K) \\ B \longleftarrow T &: E_{K_{AT}}(r_A, I_B), E_{K_{BT}}(r_B, K, I_A) \\ A \longleftarrow B &: E_{K_{AT}}(r_A, K, I_B) \end{aligned}$$

- **KTS 17: Mechanism 13 of [28]**

$$\begin{aligned} A \longrightarrow B &: r_A \\ B \longrightarrow T &: E_{K_{BT}}(r_A, r_B, I_A, K) \\ B \longleftarrow T &: E_{K_{AT}}(r_A, K, I_B), E_{K_{BT}}(r_B, I_A) \\ A \longleftarrow B &: E_{K_{AT}}(r_A, K, I_B), E_K(r'_B, r_A) \\ A \longrightarrow B &: E_K(r_A, r'_B) \end{aligned}$$

2. **Direct transmission**

   One entity generates a key and sends it to a KTC. And then the KTC forwards directly it to the other entity.

   (a) *A's key generating*

   A generates a key and sends it to a KTC.

   - **KTS 18: Capable Protocol 3**

     $A \longrightarrow T \quad : \; E_{K_{AT}}(t_A/n_A, I_B, K)$

     $B \longleftarrow T \quad : \; E_{K_{BT}}(t_T/n_T, I_A, K)$

   - **KTS 19: Capable Protocol 4**

     $A \longrightarrow T \quad : \; E_{K_{AT}}(t_A/n_A, I_B, K)$

     $B \longleftarrow T \quad : \; E_{K_{BT}}(t_T/n_T, I_A, K)$

     $A \longleftarrow B \quad : \; E_K(t_B/n_B, I_A)$

     $A \longrightarrow B \quad : \; E_K(t'_A/n'_A, I_B)$

   (b) *B's key generating*

   A asks B to communicate each other. Then B generates a key and sends it to a KTC.

   - **KTS 20: Capable Protocol 5**

     $A \longrightarrow B \quad : \; r_A$

     $B \longrightarrow T \quad : \; E_{K_{BT}}(r_A, r_B, I_A, K)$

     $A \longleftarrow T \quad : \; E_{K_{AT}}(r_A, r_B, I_B, K)$

   - **KTS 21: Capable Protocol 6**

     $A \longrightarrow B \quad : \; r_A$

     $B \longrightarrow T \quad : \; E_{K_{BT}}(r_A, r_B, I_A, K)$

     $A \longleftarrow T \quad : \; E_{K_{AT}}(r_A, r_B, I_B, K)$

     $A \longrightarrow B \quad : \; E_K(r_B, I_B)$

     $A \longleftarrow B \quad : \; E_K(r_A, I_A)$

3.2.2 Key transport protocol on asymmetric techniques

This section discusses key transport protocols based on asymmetric techniques. They are subdivided into protocols with and without signature.

Protocol without signature

1. Timestamp-based protocol

   - **KTA 1: Key transport mechanism 1 of [29]**
     $A \longrightarrow B \;\; : \; P_B(I_A, K, TVP)$

2. Random number-based protocol

   - **KTA 2: Capable Protocol 7**
     $A \longrightarrow B \;\; : \; r_A$
     $A \longleftarrow B \;\; : \; P_A(I_B, r_A, r_B, K)$

Protocol with signature

1. Timestamp-based protocol

   - **KTA 3: Encrypting signed keys [20]**
     $A \longrightarrow B \;\; : \; P_B(K, t_A, S_A(I_B, K, t_A))$
   - **KTA 4: Signing encrypted keys [20]**
     $A \longrightarrow B \;\; : \; t_A, P_B(I_A, K), S_A(I_B, t_A, P_B(I_A, K))$

2. Random number-based protocol

   - **KTA 5: Key transport mechanism 4 of [29]**
     $A \longrightarrow B \;\; : \; r_A$
     $A \longleftarrow B \;\; : \; I_A, r_A, r_B, P_A(I_B, K), S_B(I_A, r_A, r_B, P_A(I_B, K))$

Hybrid Key Transport Protocol

Hybrid key transport protocol uses symmetric encryption in addition to both asymmetric encryption and signature.

- **KTA 6: 2-pass Beller-Yacobi key transport [20]**
  $A \longrightarrow B$: $r_A, Cert_A$
  $A \longleftarrow B$: $P_A(v), E_v(Cert_B, w)$ (where $(v, w) = S_B(r_A, I_A)$)
  $Cert_A$ is of RSA's encryption public key.
  $Cert_B$ is of ElGaml's signature public key.

- **KTA 7: 4-pass Beller-Yacobi key transport [20]**
  $A \longrightarrow B$: $Cert_A$
  $A \longleftarrow B$: $P_A(K)$
  $A \longrightarrow B$: $E_K(m)$
  $A \longleftarrow B$: $E_K(S_B(m), Cert_B)$

3.3 Key Agreement protocol

Key agreement is the process that a shared secret key is established between both entities in such a way that neither of them can predetermine the value of the key.

3.3.1 Key agreement protocol on symmetric techniques

This section concisely presents key agreement protocols on symmetric techniques. The protocols are grouped in accordance with the use of a TTP.

Point-to-Point

1. Timestamp-based protocol

   - **KAS 1: Mechanism 5 of [28]**

     $A \longrightarrow B \quad : \; E_{K_{AB}}(t_A/n_A, I_B, K_A)$

     $A \longleftarrow B \quad : \; E_{K_{AB}}(t_B/n_B, I_A, K_B)$

     $K = f(K_A, K_B)$

2. Random number-based protocol

   - **KAS 2: Mechanism 6 of [28]**

     $A \longrightarrow B \quad : \; r_A$

     $A \longleftarrow B \quad : \; E_{K_{AB}}(r_B, r_A, I_A, K_B)$

     $A \longrightarrow B \quad : \; E_{K_{AB}}(r_A, r_B, K_A)$

     $K = f(K_A, K_B)$

TTP (CA)

- **KAS 3: Blom's scheme (the largest size coalition : $k = 1$) [20]**

  (public information) $p_A, p_B \in_R Z_p$

  $T: \; a, b, c \in_R Z_p$

  $\quad f(x,y) = a + b(x+y) + cxy \; mod \; p$

  $\quad g_A(x) = f(x, p_A)$

  $\quad g_B(x) = f(x, p_B)$

  $T \Longrightarrow A \quad : \; g_A(x)$

  $T \Longrightarrow B \quad : \; g_B(x)$

  $A: \; K = g_A(p_B), \quad B: \; K = g_B(p_A)$

  $K = f(p_A, p_B)$

3.3.2 Key agreement protocol on asymmetric techniques

This section describes the briefs of key agreement protocols using asymmetric techniques. Some are based on Diffie-Hellman key agreement protocol[9].

Protocol without signature

1. Timestamp-based protocol

   - **KAA 1: Capable Protocol 8**

     $A \longrightarrow B \ : \ P_B(t_A/n_A, K_A, I_A)$

     $A \longleftarrow B \ : \ P_A(t_B/n_B, K_B, I_B)$

     $K = f(K_A, K_B)$

2. Random number-based protocol

   - **KAA 2: Key transport mechanism 6 of [29]**

     $A \longrightarrow B \ : \ P_B(I_A, K_A, r_A)$

     $A \longleftarrow B \ : \ P_A(I_B, K_B, r_A, r_B)$

     $A \longrightarrow B \ : \ r_B$

     $K = f(K_A, K_B)$

   - **KAA 3: Needham-Schroeder public-key protocol [20]**

     $A \longrightarrow B \ : \ P_B(K_A, I_A)$

     $A \longleftarrow B \ : \ P_A(K_A, K_B)$

     $A \longrightarrow B \ : \ P_B(K_B)$

     $K = f(K_A, K_B)$

Protocol with signature

1. Timestamp-based protocol

   - **KAA 4: X.509 strong two-way authentication [20]**

     $A \longrightarrow B \;\;:\; t_A, r_A, I_B, P_B(K_A), S_A(t_A, r_A, I_B, P_B(K_A))$

     $A \longleftarrow B \;\;:\; t_B, r_B, I_A, r_A, P_A(K_B), S_B(t_B, r_B, I_A, r_A, P_A(K_B))$

     $K = f(K_A, K_B)$

2. Random number-based protocol

   - **KAA 5: Key transport mechanism 5 of [29]**

     $A \longrightarrow B \;\;:\; r_A$

     $A \longleftarrow B \;\;:\; S_B(r_A, r_B, I_A, P_A(I_B, K_B))$

     $A \longrightarrow B \;\;:\; S_A(r_A, r_B, I_B, P_B(I_A, K_A))$

     $K = f(K_A, K_B)$

   - **KAA 6: X.509 strong three-way authentication [20]**

     $A \longrightarrow B \;\;:\; r_A, I_B, P_B(K_A), S_A(r_A, I_B, P_B(K_A))$

     $A \longleftarrow B \;\;:\; r_B, I_A, r_A, P_A(K_B), S_B(r_B, I_A, r_A, P_A(K_B))$

     $A \longrightarrow B \;\;:\; r_B, I_B, S_A(r_B, I_B)$

     $K = f(K_A, K_B)$

Diffie-Hellman and related key agreement protocols

The security of the following protocols rests on the intractability of the Diffie-Hellman problem.

1. Non-interactive key agreement protocol

   - **KAA 7: Static Diffie-Hellman key agreement [20]**
     $A \cdots B$
     $A:\ K = z_B^a \ mod\ p, \quad B:\ K = z_A^b \ mod\ p$
     $K = g^{ab}\ mod\ p$

2. One-Pass

   - **KAA 8: ElGamal key agreement [20]**
     $A \longrightarrow B \quad : \ g^x \ mod\ p$
     $A:\ K = z_B^x \ mod\ p, \quad B:\ K = (g^x)^b \ mod\ p$
     $K = g^{xb}\ mod\ p$

   - **KAA 9: Nyberg-Rueppel key agreement [28]**
     $A:\ e = g^r \ mod\ p$
     $\quad K = z_B^r \ mod\ p$
     $\quad e' = e \cdot H(K, I_A, TVP)\ mod\ p$
     $\quad y = r - ae'$
     $A \longrightarrow B \quad : \ I_A, TVP,\ e,\ y$
     $B:\ K = e^b \ mod\ p$
     $\quad e' = e \cdot H(K, I_A, TVP)\ mod\ p$
     $\quad e = g^y \cdot z_A^{e'} \ mod\ p$

3. Two-Pass

   - **KAA 10: Ephemeral Diffie-Hellman key agreement [9]**
     $A \longrightarrow B \quad : \ g^x \ mod\ p$
     $A \longleftarrow B \quad : \ g^y \ mod\ p$
     $A:\ K = (g^y)^x \ mod\ p, \quad B:\ K = (g^x)^y \ mod\ p$
     $K = g^{xy}\ mod\ p$

- **Matsumoto-Takashima-Imai (MTI) key agreement [19]**
  - **KAA 11-1: MTI A(0) key agreement**

    $A \longrightarrow B \quad : \; g^x \; mod \; p$

    $A \longleftarrow B \quad : \; g^y \; mod \; p$

    $A: \; K = (g^y)^a \cdot z_B^{\,x} \; mod \; p, \quad B: \; K = (g^x)^b \cdot z_A^{\,y} \; mod \; p$

    $K = g^{ay+bx} \; mod \; p$
  - **KAA 11-2: MTI B(0) key agreement**

    $A \longrightarrow B \quad : \; z_B^{\,x} \; mod \; p$

    $A \longleftarrow B \quad : \; z_A^{\,y} \; mod \; p$

    $A: \; K = (z_A^{\,y})^{a^{-1}} g^{\;x} \; mod \; p, \quad B: \; K = (z_A^{\,y})^{b^{-1}} g^{\;y} \; mod \; p$

    $K = g^{x+y} \; mod \; p$
  - **KAA 11-3: MTI C(0) key agreement**

    $A \longrightarrow B \quad : \; z_B^{\,x} \; mod \; p$

    $A \longleftarrow B \quad : \; z_A^{\,y} \; mod \; p$

    $A: \; K = (z_A^{\,y})^{a^{-1}x} \; mod \; p, \quad B: \; K = (z_A^{\,y})^{b^{-1}y} \; mod \; p$

    $K = g^{xy} \; mod \; p$
  - **KAA 11-4: MTI C(1) key agreement**

    $A \longrightarrow B \quad : \; z_B^{\,ax} \; mod \; p$

    $A \longleftarrow B \quad : \; z_A^{\,by} \; mod \; p$

    $A: \; K = (z_A^{\,by})^x \; mod \; p, \quad B: \; K = (z_B^{\,ax})^y \; mod \; p$

    $K = g^{abxy} \; mod \; p$
- **KAA 12: LLK key agreement [16]**

  $A \longrightarrow B \quad : \; z_B^{\,x} \; mod \; p$

  $A \longleftarrow B \quad : \; z_A^{\,y} \cdot (z_B^{\,x})^{y/b} \; mod \; p$

  $A: \; K = z_B^{\,x} \cdot (z_A^{\,y} \cdot (z_B^{\,x})^{y/b})^{a/(a+x)} \; mod \, p, \;\; B: \; K = z_B^{\,x} \cdot z_A^{\,y} \; mod \, p$

  $K = g^{ay+bx} \; mod \; p$
- **KAA 13: KEA key agreement [21]**

  $A \longrightarrow B \quad : \; g^x \; mod \; p$

  $A \longleftarrow B \quad : \; g^y \; mod \; p$

  $A: \; K = (g^y)^a + z_B^{\,x} \; mod \; p, \quad B: \; K = (g^x)^b + z_A^{\,y} \; mod \; p$

  $K = g^{ay} + g^{bx} \; mod \; p$

- **KAA 14 : Unified Model key agreement [1]**

  $A \longrightarrow B \quad : \ g^x \ mod \ p$

  $A \longleftarrow B \quad : \ g^y \ mod \ p$

  $A: \ K = H(z_B^{\,a}, (g^y)^x), \ \ B: \ K = H(z_A^{\,b}, (g^x)^y)$

  $K = K = H(g^{ab}, g^{xy})$

- **KAA 15 : MQV key agreement [15]**

  $A \longrightarrow B \quad : \ g^x \ mod \ p$

  $A \longleftarrow B \quad : \ g^y \ mod \ p$

  $A: \ s_A = (x + a\overline{g^x}) \ mod \ q, K = (g^y (z_B)^{\overline{g^y}})^{s_A} \ mod \ p$

  $B: \ s_B = (y + b\overline{g^y}) \ mod \ q, K = (g^x (z_A)^{\overline{g^x}})^{s_B} \ mod \ p$

  $K = g^{s_A s_B} \ mod \ p$

- **KAA 16: Song-Kim key agreement [22]**

  $A \longrightarrow B \quad : \ g^x \ mod \ p$

  $A \longleftarrow B \quad : \ g^y \ mod \ p$

  $A: \ K = (g^y)^{(a+x)} \cdot z_B^{\,x} \ mod \ p, \quad B: \ K = (g^x)^{(b+y)} \cdot z_A^{\,y} \ mod \ p$

  $K = g^{xy+ay+bx} \ mod \ p$

4. Three-Pass

- **KAA 17: Protocol using MACs [1]**

  $A \longrightarrow B \quad : \ g^x \ mod \ p$

  $A \longleftarrow B \quad : \ g^y \ mod \ p, MAC_K(2, g^y, g^x, I_B, I_A)$

  $A \longrightarrow B \quad : \ MAC_K(3, g^x, g^y, I_A, I_B)$

- **KAA 18: Station-to-Station protocol (STS) [5]**

  $A \longrightarrow B \quad : \ g^x \ mod \ p$

  $A \longleftarrow B \quad : \ g^y \ mod \ p, E_K(S_B(g^y, g^x, I_A))$

  $A \longrightarrow B \quad : \ E_K(S_A(g^x, g^y, I_B))$

  $A: \ K = (g^y)^x \ mod \ p, \quad B: \ K = (g^x)^y \ mod \ p$

  $K = g^{xy} \ mod \ p$

3.4 Comparison

This section compares key establishment protocols classified above. Table 3.1 and Table 3.2 summarize the properties of the key transport protocols on symmetric techniques and asymmetric techniques, respectively. In Table 3.3 and Table 3.4, the major properties of the key agreement protocols on symmetric techniques and asymmetric techniques respectively are presented.

All the key establishment protocols using symmetric techniques offer at least IKA, because only entities knowing a specific secret key can recover the correct key. KC and EA can be achieved for every protocol by using the some techniques[28]. Distinguishing identifiers included in messages are used to protect against certain type of substitution attacks.

A	The property is provided to A in the protocol.
A, B	The property is provided to both entities, A and B in the protocol.
$A \rightarrow B$	The property is provided from A to B in the protocol.
$A \leftrightarrow B$	The property is provided from A to B and from B to A.
$\cdot$	The property is not provided in the protocol.
$\circ$	The property is provided in the protocol.
t, n, r	The property is provided by using t, n, r.
i, j	The number of public key operations of each A and B is i and j, respectively.

Table 3.1: Key transport protocols on symmetric techniques

KTS	TTP	NP	KA	KC	EA	Cl	Fr
1	$\cdot$	1	$\cdot$	$\cdot$	$\cdot$	A	TVP
2	$\cdot$	1	$\cdot$	$\cdot$	$\cdot$	A	$\cdot$
3	$\cdot$	1	$\circ$	$\cdot$	A	A	t/n
4	$\cdot$	2	$\circ$	$\cdot$	B	B	r
5-1	$\cdot$	3	$\circ$	$\cdot$	A, B	B	r
5-2	$\cdot$	3	$\circ$	$\circ$	A, B	B	r
6	$\cdot$	3	$\cdot$	$\cdot$	$\cdot$	A	r
7	KDC	3	$\circ$	$\cdot$	$\cdot$	KDC	$\cdot$
8-1	KDC	4	$\circ$	$\circ$	A, B	KDC	t/n
8-2	KDC	4	$\circ$	$\circ$	A, B	KDC	t/n
9	KDC	5	$\circ$	$\circ$	A	KDC	r
10	KDC	4	$\circ$	$\cdot$	$\cdot$	KDC	r, m
11	KDC	5	$\circ$	$\circ$	A, B	KDC	r
12	KDC	3	$\circ$	$\cdot$	$\cdot$	KDC	t/n
13	KDC	4	$\circ$	$\cdot$	A, B	KDC	r
14	KTC	3	$\cdot$	$\cdot$	$\cdot$	A	$\cdot$
15	KTC	4	$\circ$	$\circ$	A, B	A	t/n
16	KTC	4	$\circ$	$\cdot$	$\cdot$	B	r
17	KTC	5	$\circ$	$\circ$	A, B	B	r
18	KTC	2	$\circ$	$\cdot$	$\cdot$	A	t/n
19	KTC	4	$\circ$	$\circ$	$\circ$	A	t/n
20	KTC	3	$\circ$	$\cdot$	$\cdot$	B	r
21	KTC	5	$\circ$	$\circ$	$\circ$	B	r

Table 3.2: Key transport protocols on asymmetric techniques

KTA	Sig	NP	IKA	KC	EA	Cl	Fr	PKO
1	·	1	$A \leftarrow B$	·	A	A	TVP	1,1
2	·	2	$A \rightarrow B$	·	B	B	r	1,1
3	∘	1	$A \leftarrow B$	$A \rightarrow B$	A	A	t	2,2
4	∘	1	$A \leftarrow B$	$A \rightarrow B$	A	A	t	2,2
5	∘	2	$A \rightarrow B$	$A \leftarrow B$	B	B	r	2,2
6	∘	2	$A \leftrightarrow B$	$A \leftrightarrow B$	A, B	A, B	r	2,2
7	∘	4	$A \leftrightarrow B$	$A \leftrightarrow B$	A, B	B	m	2,2

Table 3.3: Key agreement protocols on symmetric techniques

KAS	TTP	NP	KA	KC	EA	Fr
1	·	2	∘	·	A, B	t/n
2	·	3	∘	·	A, B	r
3	∘	1	∘	·	·	·

Table 3.4: Key agreement protocols on asymmetric techniques

KAA	Sig	NP	IKA	KC	EA	Fr	PKO
1	$\cdot$	2	$A \leftrightarrow B$	$\cdot$	$\cdot$	t/n	2,2
2	$\cdot$	3	$A \leftrightarrow B$	$\cdot$	$\cdot$	r	2,2
3	$\cdot$	3	$A \leftrightarrow B$	$\cdot$	$\cdot$	$\cdot$	3,3
4	$\circ$	2	$A \leftrightarrow B$	$\cdot$	A, B	t, r	4,4
5	$\circ$	3	$A \leftrightarrow B$	$\cdot$	A, B	r	4,4
6	$\circ$	3	$A \leftrightarrow B$	$\cdot$	A, B	r	5,5
7	$\cdot$	0	$A \leftrightarrow B$	$\cdot$	$\cdot$	$\cdot$	1,1
8	$\cdot$	1	$A \leftarrow B$	$\cdot$	$\cdot$	$\circ$	2,1
9	$\circ$	1	$A \leftrightarrow B$	$A \leftarrow B$	A	$\circ$	2,3
10	$\cdot$	2	$\cdot$	$\cdot$	$\cdot$	$\circ$	2,2
11-1	$\cdot$	2	$A \leftrightarrow B$	$\cdot$	$\cdot$	$\circ$	3,3
11-2	$\cdot$	2	$A \leftrightarrow B$	$\cdot$	$\cdot$	$\circ$	3,3
11-3	$\cdot$	2	$A \leftrightarrow B$	$\cdot$	$\cdot$	$\circ$	2,2
11-4	$\cdot$	2	$A \leftrightarrow B$	$\cdot$	$\cdot$	$\circ$	2,2
12	$\cdot$	2	$A \leftrightarrow B$	$\cdot$	$\cdot$	$\circ$	2,2
13	$\cdot$	2	$A \leftrightarrow B$	$\cdot$	$\cdot$	$\circ$	3,3
14	$\cdot$	2	$A \leftrightarrow B$	$\cdot$	$\cdot$	$\circ$	3,3
15	$\cdot$	2	$A \leftrightarrow B$	$\cdot$	$\cdot$	$\circ$	2.5,2.5
16	$\cdot$	2	$A \leftrightarrow B$	$\cdot$	$\cdot$	$\circ$	3,3
17	$\cdot$	3	$A \leftrightarrow B$	$A \leftrightarrow B$	A, B	$\circ$	3,3
18	$\circ$	3	$A \leftrightarrow B$	$A \leftrightarrow B$	A, B	$\circ$	3,3

IV. New Key Agreement Protocols based on Diffie-Hellman Problem

This chapter proposes the new AK, AKC and two-pass unilateral AKC protocols. Domain parameters and static keys are set up as described in previous chapter.

4.1 The related AK protocols

This section describes AK protocols like the MTI/A0, two-pass Unified Model, and two-pass MQV protocols. All protocols in this section have been described in the setting of the group of points on an elliptic curve defined over a finite field. The security of AK protocols in this paper is based on the *Diffie-Hellman problem*[9] in elliptic curve group (ECDHP): given an elliptic curve E defined over a finite field $\mathbf{F}_q$, a base point $P \in E(\mathbf{F}_q)$ of order n and two points generated by P, xP and yP (where x and y are integer), find xyP. This problem is closely related to the well-known *elliptic curve discrete logarithm problem* (ECDLP) (given $E(\mathbf{F}_q), P, n$ and xP, find x)[6] and there is strong evidence that the two problems are computationally equivalent. (*e.g.*, see [7] and [18])

4.1.1 MTI/A0

The MTI/A0 key agreement protocol was proposed by Matsumoto *et al.* in 1986[19]. It was designed to provide implicit key authentication. The similar protocols are KEA[21] and those proposed by Goss[11] and Yacobi[23].

Protocol 1

1. A generates a random integer r_A, $1 \leq r_A \leq n-1$, computes the point $R_A = r_A P$, and sends $(R_A, Cert_A)$ to B.

2. B generates a random integer r_B, $1 \leq r_B \leq n-1$, computes the point $R_B = r_B P$, and sends $(R_B, Cert_B)$ to A.

3. A computes $K = r_A W_B + w_A R_B$.

4. B computes $K = r_B W_A + w_B R_A$.

5. The session key is $k = kdf(K)$ (where kdf is a key derivation function).

Remark

- A and B commonly compute $K = (r_A w_B + r_B w_A)P$.

- It does not provide FS since an adversary who learns w_A and w_B can compute all session keys established by A and B.

4.1.2 Two-pass Unified Model

The Unified Model proposed by Ankney *et al.*[1] is an AK protocol in the draft standards ANSI X9.42[24], ANSI X9.63[25], and IEEE P1363[26]. In the protocol, the notation $||$ denotes concatenation.

Protocol 2

1. A generates a random integer r_A, $1 \leq r_A \leq n-1$, computes the point $R_A = r_A P$, and sends $(R_A, Cert_A)$ to B.

2. B generates a random integer r_B, $1 \leq r_B \leq n-1$, computes the point $R_B = r_B P$, and sends $(R_B, Cert_B)$ to A.

3. A computes $Z_s = w_A W_B$ and $Z_e = r_A R_B$

4. B computes $Z_s = w_B W_A$ and $Z_e = r_B R_A$

5. The session key is $k = kdf(Z_s || Z_e)$.

Remark

- A and B commonly compute $K = (w_A w_B P)||(r_A r_B P)$.

- It does not possess the K-CI attribute, since an adversary who learns w_A(resp. w_B) can impersonate the other entity B(resp. A) to A(resp. B).

4.1.3 Two-pass MQV

The MQV[15] is proposed by Law *et al.*, which is in the draft standards ANSI X9.42[24], ANSI X9.63[25], and IEEE P1363[26].

The following notations are used. f denotes the bitlength of n, the prime order of the base point P; *i.e.*, $f = \lfloor log_2 n \rfloor + 1$. If Q is a finite elliptic curve point, then $\overline{Q}$ is defined as follows. Let x be the x-coordinate of Q, and let $\overline{x}$ be the integer obtained from the binary representation of x. (The value of $\overline{x}$ will depend on the representation chosen for the elements of the field $\mathbf{F}_q$). Then $\overline{Q}$ is defined to be the integer $(\overline{x} \bmod 2^{\lceil f/2 \rceil}) + 2^{\lceil f/2 \rceil}$. Observe that $(\overline{Q} \bmod n) \neq 0$[4].

Protocol 3

1. A generates a random integer r_A, $1 \leq r_A \leq n-1$, computes the point $R_A = r_A P$, and sends $(R_A, Cert_A)$ to B.

2. B generates a random integer r_B, $1 \leq r_B \leq n-1$, computes the point $R_B = r_B P$, and sends $(R_B, Cert_B)$ to A.

3. A computes $s_A = r_A + \overline{R}_A w_A \bmod n$, $K = s_A(R_B + \overline{R}_B W_B)$

4. B computes $s_B = r_B + \overline{R}_B w_B \bmod n$, $K = s_B(R_A + \overline{R}_A W_A)$

5. The session key is $k = kdf(K)$.

Remark

- A and B commonly compute $K = (s_A s_B)P = (r_A r_B + r_A w_B \overline{R}_B + r_B w_A \overline{R}_A + w_A w_B \overline{R}_A \overline{R}_B)P$. The expression for $\overline{R}_A$ uses only half the bits of the x-coordinate of R_A. This was done in order to increase the efficiency of computing K because the scalar multiplication $\overline{R}_A W_A$ can be done in half the time of a full scalar multiplication[15].

- It does not provide UK-S attribute, which has recently been observed by Kaliski[14].

4.2 New Key Agreement Protocols

4.2.1 AK protocol

Protocol 4 provides more desirable security attributes than AK protocols described in previous section.

Protocol 4

If A and B do not previously possess authentic copies of each other's static public keys, the certificates should be included in the flows[15].

1. A generates a random integer r_A, $1 \leq r_A \leq n-1$, computes the point $R_A = r_A P$, and sends this to B.

2. B generates a random integer r_B, $1 \leq r_B \leq n-1$, computes the point $R_B = r_B P$, and sends this to A.

3. A does an embedded key validation of R_B. If the validation fails, then A terminates the protocol run with failure. Otherwise, A computes $K = cr_A W_B + c(w_A + r_A)R_B$. If $K = \mathcal{O}$, then A terminates the protocol run with failure.

4. B does an embedded key validation of R_A. If the validation fails, then B terminates the protocol run with failure. Otherwise, B computes $K = cr_B W_A + c(w_B + r_B)R_A$. If $K = \mathcal{O}$, then B terminates the protocol run with failure.

5. The session key is $k = kdf(K)$.

Multiplication by c ensures that the shared secret K is a point in the subgroup of order n in $E(\mathbf{F}_q)$ to protect against small subgroup attack as suggested in [17]. The check $K = \mathcal{O}$ ensures that K is a finite point[15].

A and B compute the shared secret $K = c(r_A w_B + r_B w_A + r_A r_B)P$.

Security

Although the security of Protocol 4 has not been formally proven in a model of distributed computing[3], heuristic arguments suggest that Protocol 4 provides mutual IKA[15]; entity A is assured that no other entity aside from B can possibly learn the value of a particular secret key, since the secret key can be computed only by entity who knows B's private keys, r_B and w_B, vice versa. In addition, Protocol 4 also appears to have many desirable security attributes listed in previous chapter as follows.

The protocol provides *known-key security*. Each run of the protocol between two entities A and B should produce a unique session key which depends on r_A and r_B. Although an adversary has learned some other session keys, he can't compute $r_A w_B P$, $r_A w_A P$ and $r_A r_B P$ from them, because he doesn't know ephemeral private keys r_A and r_B. Therefore the protocol still achieve its goal in the face of the adversary.

It also possesses *forward secrecy* provided that EKA of all session keys is supplied. Suppose that static private keys w_A and w_B of two entities are compromised. However, the secrecy of previous session keys established by honest entities is not affected, because it is difficult by the ECDLP that an adversary gains $r_A r_B P$ of the shared secret from $r_A P$ and $r_B P$ which he can know.

It resists *key-compromise impersonation*. Though A's static private key w_A is disclosed, this loss does not enable an adversary to impersonate the other entity B to A by the fact that the adversary still faces the ECDLP style problem of working out $r_A w_B P$ from w_A, r_B, $r_A P$, $r_B P$, $w_A P$ and $w_B P$ to learn the shared secret. Similarly, though B's static private key w_B is disclosed, it is difficult that an adversary impersonates A to B.

It also prevents *unknown key-share*. According to the assumption of this protocol that CA has verified that A possesses the private key w_A

corresponding to her static public key W_A, an adversary can't register A's public key W_A as its own and subsequently deceive B into believing that A's messages are originated from the adversary. In addition, it does not have the *duplicate-signature key selection (DSKS)* property[5]. Therefore B cannot be coerced into sharing a key with entity A without B's knowledge.

Performance

From A's point of view, the dominant computational steps in a run of Protocol 4 are the scalar multiplications r_AP, r_AW_B and $(w_A+r_A)R_B$. Refer to an efficient scalar multiplication method using Frobenius expansion suggested by Cheon *et al*[8]. Hence (full) scalar multiplications required by each entity are 3. Since r_AP and r_AW_B can be computed off-line by A provided that A and B do previously possess authentic copies of each other's static public key, the on-line work required by each entity is only 1 scalar multiplication.

Instead of $k = kdf(c(r_Aw_B + r_Bw_A + r_Ar_B)P)$, the following variations can be used as a session key.

(i) $k = kdf(c(r_Aw_B + r_Bw_A)P||cr_Ar_BP)$.
A computes $k = kdf(cr_AW_B + cw_AR_B||cr_AR_B)$.
B computes $k = kdf(cr_BW_A + cw_BR_A||cr_BR_A)$.

(ii) $k = kdf(cr_Aw_BP||cr_Bw_AP||cr_Ar_BP)$.
A computes $k = kdf(cr_AW_B||cw_AR_B||cr_AR_B)$.
B computes $k = kdf(cw_BR_A||cr_BW_A||cr_BR_A)$.

The protocols using these session keys require 4 scalar multiplications and if precomputations are discounted, the total number of on-line scalar multiplications per entity is 2. Therefore the using of these will cause some degradation in performance of the protocols.

4.2.2 AKC protocol

This section describes the AKC variant of Protocol 4. Protocol 5 (AKC protocol) is derived from Protocol 4 (AK protocol) by adding the MACs of the flow number, identities, and the ephemeral public keys. Here, MACs are used to provide key confirmation, and $\mathcal{H}_1$ and $\mathcal{H}_2$ are (independent) key derivation functions[15].

Protocol 5

1. A generates a random integer r_A, $1 \le r_A \le n-1$, computes the point $R_A = r_A P$, and sends this to B.

2. (a) B does an embedded key validation of R_A. If the validation fails, then B terminates the protocol run with failure.

   (b) Otherwise, B generates a random integer r_B, $1 \le r_B \le n-1$ and computes the point $R_B = r_B P$.

   (c) B computes $K = cr_B W_A + c(w_B + r_B)R_A$. If $K = \mathcal{O}$, then B terminates the protocol run with failure. Otherwise, the shared secret is the point K.

   (d) B uses the x-coordinate z of the point K to compute two shared keys $k = \mathcal{H}_1(z)$ and $k' = \mathcal{H}_2(z)$.

   (e) B computes $\mathrm{MAC}_{k'}(2, ID_B, ID_A, R_B, R_A)$ and sends this together with R_B to A.

3. (a) A does an embedded key validation of R_B. If the validation fails, then A terminates the protocol run with failure.

   (b) Otherwise, A computes $K = cr_A W_B + c(w_A + r_A)R_B$. If $K = \mathcal{O}$, then A terminates the protocol run with failure.

   (c) Otherwise, A uses the x-coordinate z of the point K to compute two shared keys $k = \mathcal{H}_1(z)$ and $k' = \mathcal{H}_2(z)$.

(d) A computes $\mathrm{MAC}_{k'}(2, ID_B, ID_A, R_B, R_A)$ and verifies that this equals what was sent by B.

(e) A computes $\mathrm{MAC}_{k'}(3, ID_A, ID_B, R_A, R_B)$ and sends to B.

4. B computes $\mathrm{MAC}_{k'}(3, ID_A, ID_B, R_A, R_B)$ and verifies that this equals what was sent by A.

5. The session key is k.

Security

AKC Protocol 5 is derived from AK Protocol 4 by adding key confirmation to the latter using the fact that key confirmation of k' implies that of k. This is done in exactly the same way AKC Protocol 2 of [3] was derived from AK Protocol 3 of [3]. Protocol 2 of [3] was formally proven to be a secure AKC protocol. Heuristic arguments suggest that Protocol 5 has all the desirable security attributes listed in previous chapter.

Performance

Since MACs can be computed efficiently, this method of adding key confirmation to an AK protocol does not place a significant computational burden on the key agreement mechanism. However, the number of messages exchanged is increased one more[4].

4.2.3 Two-pass unilateral AKC protocol

This section describes a new two-pass authenticated key agreement protocol providing unilateral key confirmation. In practice, key agreement protocol is that two entities agree a secret key being used for subsequently cryptographic communication. An entity, initiator, who wants to communicate with a second entity, responder, first sends his information to the responder. And then he receives any information and an encrypted message over a key computed by the responder, who has computed the secret key with information of the initiator and his own. He also computes the secret key, and assures the responder possesses the particular secret key through verifying the encrypted message from the responder. Such process can be possible by only exchanging two flows.

Protocol 6 is the two-pass key agreement protocol which provides mutual IKA and unilateral EKA. It also provides many desirable security attributes. In this protocol, $S_A(M)$ denotes A's signature over M and ID_A denotes A's identity information.

Protocol 6

1. (a) A generates a random integer r_A, $1 \leq r_A \leq n-1$, and computes the point $R_A = r_A P$
   (b) A computes $S_A(R_A, ID_A)$ and sends this with R_A to B.
2. (a) B does an embedded key validation of R_A. If the validation fails, then B terminates the protocol run with failure.
   (b) Otherwise, B verifies the signature $S_A(R_A, ID_A)$ with A's public key. If the verification fails, then B terminates the protocol run with failure.
   (c) Otherwise, B generates a random integer r_B, $1 \leq r_B \leq n-1$, and computes the point $R_B = r_B P$.
   (d) B computes $K = cr_B W_A + c(w_B + r_B)R_A$. If $K = \mathcal{O}$, then B terminates the protocol run with failure.
   (e) Otherwise, B uses the x-coordinate z of the point K to compute two shared keys $k = \mathcal{H}_1(z)$ and $k' = \mathcal{H}_2(z)$.
   (f) B computes $\mathrm{MAC}_{k'}(ID_B, ID_A, R_B, R_A)$ and sends this with R_B to A.

3. (a) A does an embedded key validation of R_B. If the validation fails, then A terminates the protocol run with failure.

   (b) Otherwise, A computes $K = cr_A W_B + c(w_A + r_A)R_B$. If $K = \mathcal{O}$, then A terminates the protocol run with failure.

   (c) Otherwise, A uses the x-coordinate z of the point K to compute two shared keys $k = \mathcal{H}_1(z)$ and $k' = \mathcal{H}_2(z)$.

   (d) A computes $\mathrm{MAC}_{k'}(ID_B, ID_A, R_B, R_A)$ and verifies that this equals what was sent by B.

4. The session key is k.

Security

Entity A sends R_A and $S_A(R_A, ID_A)$ to B, then the signature of A provides EA of A to B. The MAC sent from B to A provides EA of B to A and KC of B to A. Subsequently A and B are on any communication using the shared key. Then an encrypted message sent from A gives an assurance to B that A actually possesses the key (*i.e.*, KC of A to B) during a 'real-time' communication. Protocol 6 also provides K-KS, FS, K-CI and UK-S attributes by the same reason as one of Protocol 5.

Performance

This protocol requires additional computations of the signature. However, since $S_A(R_A, ID_A)$ can be precomputed by A, the computations of entity B for verifying the signature only are increased in the on-line work. Most of all, Protocol 6 has the advantage in the number of flow than Protocol 5, as the number of flow is the dominant burden in performance.

4.3 Comparison

This section compares the security attributes and performance of Protocol 1 to Protocol 6. Table 4.1 presents the shared secret of the protocols, and A's computations for generating the shared secret. The shared secret of Protocol 1 is composed of $r_A w_B P$ and $r_B w_A P$, Protocol 2 is $r_A r_B P$ and $w_A w_B P$, Protocol 3 is $r_A r_B P$, $r_A w_B \overline{R}_B P$, $r_B w_A \overline{R}_A P$ and $w_A w_B \overline{R}_A \overline{R}_B P$, and Protocols 4, 5 and 6 are $r_A r_B P$, $r_A w_B P$ and $r_B w_A P$, respectively.

Protocol	Shared secret (K)	A's computations
Protocol 1	$(r_A w_B + r_B w_A)P$	$K = r_A W_B + w_A R_B$
Protocol 2	$(w_A w_B)P \vert\vert (r_A r_B)P$	$K = w_A W_B \vert\vert r_A R_B$
Protocol 3	$(r_A r_B +$ $r_A w_B \overline{R}_B + r_B w_A \overline{R}_A +$ $w_A w_B \overline{R}_A \overline{R}_B)P$	$\overline{R}_A = R_A \bmod 2^{\lceil f/2 \rceil} + 2^{\lceil f/2 \rceil}$ $s_A = r_A + \overline{R}_A w_A \bmod n$ $K = s_A(R_B + \overline{R}_B W_B)$
Protocol 4,5,6	$(r_A r_B + r_A w_B + r_B w_A)P$	$K = r_A W_B + (w_A + r_A) R_B$

Table 4.1: The shared secret of AK protocols and A's computations

Table 4.2 contains a summary of the services that are believed to be provided by the AK and AKC protocols discussed in previous chapter. The services are discussed in the context of an entity A who has successfully executed the key agreement protocol over an open network wishing to establish keying data with entity B. The provision of these assurances is considered that both A and B are honest and have always executed the protocol correctly[4].

In Table 4.3, the number of scalar multiplications required in each protocol is compared. Protocols 1, 2, 4 and 5 commonly require 3 scalar multiplications without precomputations (of quantities involving the entity's static and ephemeral keys and the other entity's static keys) and require 1 scalar multiplication with precomputations. Protocol 3 requires 2.5 and 1.5 scalar multiplications, respectively. As noted in previous section, MACs can be computed efficiently and hence the AKC variants have essentially the same computational overhead as their AK counterparts[4]. However, they require an extra flow which is dominated in performance. In this sense, Protocol 6 is considerable because it requires only two flows like AK Protocol 4. It satisfies more security attributes than Protocol 4 and the same ones as Protocol 5 except merely providing EKA of B to A. However, since EKA of A to B can be provided in subsequently cryptographic communication, it doesn't matter in security to omit the service in key agreement protocol. On the other hand, the computation overhead of the signature is required. As the signature can be precomputed, the computation for verifying the signature only is added to the on-line work. Consequently, Protocol 6 is said to be more efficient than Protocol 5 and more secure than Protocol 4.

As shown in Table 4.2, Protocol 4 provides more desirable security attributes than other AK protocols discussed above. If EKA is additionally provided in AK protocols (*i.e.*, if AK protocols extend to AKC protocols), they possess many desirable security attributes. By the way, the AKC MTI/A0 still does not provide FS and the AKC Unified Model does not K-CI, while the AKC MQV provides UK-S which the AK MQV doesn't exhibit.

AKC Protocol 5 provides all security attributes in Table 4.2 as well as the AKC MQV. The AKC MQV requires 2.5 scalar multiplications per entity without precomputations and 1.5 scalar multiplications with precomputations like Protocol 3, while Protocol 5 requires 3 and 1 scalar multiplications, respectively. So, in the case that precomputations are discounted, Protocol 5 is more efficient than the MQV protocol in on-line processing.

It has been founded later that Protocol IIA, which was proposed by Just and Vaudenay at Asiacrypt'96[13], are similar to Protocol 4, 5 and 6 in session key. However, the works in this paper have been done independently with different approaches. In [13], Just and Vaudenay described the possible attacks of Protocol A0 (MTI/A0), *i.e.*, two minor (repairable) attacks, two more serious attacks and deficiency of FS. So, they proposed a Diffie-Hellman based 3-pass protocol (Protocol IIA) which provides for (i) KA, (ii) KC and (iii) FS. They examined the security of their protocol against some passive and active attacks as Theorem 1 and 2 in [13].

On the other hand, Protocol 4, 5 and 6 are designed to solve the weakness of KEA or MTI/A0, Unified Model and MQV referring to Table 1 in [4]. First, Protocol 4 are proposed to provide against the faults of previous AK protocols; absence of FS in KEA and MTI/A0, K-CI attack in Unified Model and UK-S attack in MQV. It could be extended to Protocol 5 easily by a general framework using MACs to provide mutual EKA. Protocol 6 is for trade-off between the performance of Protocol 4 and the security of Protocol 5. The proposed protocols in this paper are considered over an elliptic curve group for efficiency.

Protocol IIA and Protocol 5 have the similar framework. Table 4.4 shows the comparison of these protocols. First, Protocol IIA is took place in a finite field $\mathbf{F}_q$ and Protocol 5 is in setting of the group of points on an elliptic curve E defined over a finite field $\mathbf{F}_q$. For comparison, Protocol 5 can be thought in $\mathbf{F}_q$. For the generation of session key, in the case of Protocol IIA, A computes $K = (g^y \cdot g^b)^x \cdot (g^y)^a \; mod \; p$ and B computes $K = (g^x \cdot g^a)^y \cdot (g^x)^b \; mod \; p$, whereas in Protocol 5, A and B compute $K = (g^y)^{(a+x)} \cdot (g^b)^x \; mod \; p$ and $K = (g^x)^{(b+y)} \cdot (g^a)^y \; mod \; p$, respectively. So, Protocol IIA requires 2 modular multiplications and 2 modular exponentiations per entity for the session key, while Protocol 5 needs 1 modular addition, 1 modular multiplication and 2 modular exponentiations per entity. In the viewpoint of the whole protocol, the total number of modular exponentiations per entity for Protocol IIA and Protocol 5 is 5 and 4, respectively. If quantities of the entity's static and ephemeral keys and the other entity's static key are precomputed, the total number of on-line modular exponentiations per entity reduces to 3 and 2. Hence, Protocol IIA requires one more modular exponentiation than Protocol 5 due to computation of pseudo-corroboration[13]. This paper also proposes two other session keys.

Protocol	IKA	EKA	K-KS	FS	K-CI	UK-S
Protocol 1	$A \leftrightarrow B$	$\cdot$	$\circ$	$\cdot$	$\circ$	$\circ$
AKC MTI/A0	$A \leftrightarrow B$	$A \leftrightarrow B$	$\circ$	$\cdot$	$\circ$	$\circ$
Protocol 2	$A \leftrightarrow B$	$\cdot$	$*^a$	$*^b$	$\cdot$	$\circ$
AKC Unified Model	$A \leftrightarrow B$	$A \leftrightarrow B$	$\circ$	$\circ$	$\cdot$	$\circ$
Protocol 3	$A \leftrightarrow B$	$\cdot$	$\circ$	$*^b$	$\circ$	$\cdot$
AKC MQV	$A \leftrightarrow B$	$A \leftrightarrow B$	$\circ$	$\circ$	$\circ$	$\circ$
Protocol 4	$A \leftrightarrow B$	$\cdot$	$\circ$	$*^b$	$\circ$	$\circ$
Protocol 5	$A \leftrightarrow B$	$A \leftrightarrow B$	$\circ$	$\circ$	$\circ$	$\circ$
Protocol 6	$A \leftrightarrow B$	$A \leftarrow B$	$\circ$	$\circ$	$\circ$	$\circ$

$A \leftrightarrow B$: the assurance is provided from A to B and from B to A.

$A \rightarrow B$: the assurance is provided to from A to B.

$\circ$: the assurance is provided to A no matter whether A initiated the protocol or not.

$\cdot$: the assurance is not provided to A by the protocol.

$*^a$: Here the technicality hinges on the definition of what contributes another session key. The service of known-key security is certainly provided if the protocol is extended so that explicit authentication of all session keys is supplied[4].

$*^b$: Again the technicality concerns key confirmation. Both protocols provide forward secrecy if explicit authentication is supplied for all session keys. If not supplied, then the service of forward secrecy cannot be guaranteed[4].

Table 4.2: Security services offered by AK and AKC protocols

Total Number	Without precomputations	With precomputations
Protocol 1	3	1
Protocol 2	3	1
Protocol 3	2.5	1.5
Protocols 4,5	3	1

Table 4.3: The number of scalar multiplications required per entity

	Protocol IIA	Protocol 5
Group	Finite field, $\mathbf{F}_q$	Elliptic curve, $E(\mathbf{F}_q)$
Session key - modular computations	$A : (g^y \cdot g^b)^x \cdot (g^y)^a$ $B : (g^x \cdot g^a)^y \cdot (g^x)^b$ 0 addition 2 multiplications 2 exponentiations	$A : (g^y)^{(a+x)} \cdot (g^b)^x$ $B : (g^x)^{(b+y)} \cdot (g^a)^y$ 1 addition 1 multiplication 2 exponentiations
Total exponentiations (in precomputations)	5 (3)	4 (2)

Table 4.4: Comparison of Protocol IIA and Protocol 5

V. Concluding Remarks and Further Works

5.1 Concluding Remarks

We have surveyed and classified many key establishment protocols proposed so far in accordance with key control, cryptographic techniques used, TTP, time-variant parameters and the number of passes. The major properties of the protocols have been compared into four groups. Most important properties of key establishment protocols depend on the structure of the messages exchanged, not on the underlying cryptographic algorithms. Therefore, for reducing errors in key establishment protocol, the prudent design of protocol messages is rather of consequence.

New key agreement protocols also are proposed. First, Protocol 4 (AK protocol) has been proposed to provide the desirable security attributes which are not provided by the MTI/A0, two-pass Unified Model and two-pass MQV protocols.

Second, Protocol 5 (AKC protocol) derived from Protocol 4 provides all the security requirement discussed in [5]; implicit key authentication, explicit key authentication, known-key security, forward secrecy, key-compromise impersonation and unknown key-share attributes. The three-pass MQV protocol (AKC protocol) also possesses the same security attributes as Protocol 5. However, if precomputations (of quantities of the entity's static and ephemeral keys and the other entity's static keys) are discounted, Protocol 5 requires less scalar multiplications than the MQV protocol.

Last, Protocol 6, the two-pass unilateral AKC protocol, has been designed as the other variant of Protocol 4. It provides not only mutual implicit key authentication but also unilateral explicit key authentication. Instead, it requires the computation overhead related to the signature and the MAC. However, since the signature can be precomputed and the MAC can be efficiently computed, in practice, the computations increased in on-line work are only quantities for verifying the signature. So, Protocol 6 has the advantage of security than Protocol 4 and of performance than Protocol 5.

The proposed protocols have been compared with the MTI/A0, Unified Model and MQV in security and performance. Protocol 5 also has been compared with the similar protocol, Protocol IIA of [13].

5.2 Further Works

The proposed protocols have been shown informally by heuristic arguments to provide the claimed attributes of security. One of the important further works is to prove formal security of the protocols[2, 3].

The following three conjectures are required to show that the security of the proposed protocols can be rigorously proved in Bellare-Rogaway model[2] of distributed computing as done in [3].

(i) Protocol 4 is a secure AK protocol provided that Elliptic Curve Diffie-Hellman Scheme (ECDHS) is secure.

(ii) Protocol 5 is a secure AKC protocol provided that ECDHS and MAC are secure, and $\mathcal{H}_1$ and $\mathcal{H}_2$ are independent random oracles.

(iii) Protocol 6 is a secure unilateral AKC protocol provided that ECDHS, MAC and digital signature scheme are secure, and $\mathcal{H}_1$ and $\mathcal{H}_2$ are independent random oracles.

디피 헬만 문제에 기반한 키 합의 프로토콜에 관한 연구

송보연

불안전한 통신로를 통해 정보 전송을 안전하게 하기 위해서는 정보를 암호화해서 주고 받는 암호 통신이 필수 불가결하다. 이러한 암호 통신을 위해 프로토콜에 참여하는 실체들은 암·복호화에 이용하게 될 키를 공유하는 과정을 선행해야 한다.

본 논문는 이와 같이 실체들이 암호 통신에 이용할 공유키를 확립하는 과정인 키 확립 프로토콜에 관한 연구이다. 먼저, 기존의 키 확립 프로토콜들을 키를 생성하는 방법, 이용하는 암호 기술, 신뢰 기관이 참여하는 여부, 시간 변수 사용 여부, 주고받는 메시지 수 등에 따라 분류해서 기술한다. 그리고 크게 네 그룹 즉, 대칭키 기술을 사용하는 키 전송 프로토콜, 공개키를 사용하는 키 전송 프로토콜, 대칭키 기술을 사용하는 키 합의 프로토콜, 공개키를 사용하는 키 합의 프로토콜로 나누어 각 프로토콜의 특성을 비교한다.

다음으로 키 확립 프로토콜 중 디피 헬만 문제에 기반한 키 합의 프로토콜을 집중해서 연구한다. 기존의 제시된 인증된 키 합의 프로토콜인 MTI/A0, Unified Model, MQV를 살펴보고 이들의 약점을 해결하는 새로운 AK 프로토콜, AKC 프로토콜, 2개의 메시지로 구성된 일방향 AKC 프로토콜을 타원곡선 상에서 제안한다. 그리고 나서 기존의 프로토콜들과 새롭게 제시한 프로토콜들을 안전성과 성능면에서 비교한다. 또 제안한 프로토콜들을 그와 유사한 프로토콜 IIA와 비교한다.

References

[1] R. Ankney, D. Hohnson and M. Matyas, "The Unified Model", contribution to X9F1, October 1995.

[2] M. Bellare and P. Rogaway, "Entity Authentication and Key Distributions", *Advances in Cryptology - Crypto '93*, LNCS 773, Springer-Verlag, pp.232-249, 1994.

[3] S. Blake-Wilson, C. Johnson and A. Menezes, "Key Agreement Protocols and their Security Analysis", *Proceedings of the sixth IMA International Conference on Cryptography and Coding*, LNCS 1355, Springer-Verlag, pp.30-45, 1997.

[4] S. Blake-Wilson and A. Menezes, "Authenticated Diffie-Hellman Key Agreement Protocols", *Proceedings of the 5th Annual Workshop on Selected Areas in Cryptography (SAC '98)*, LNCS 1556, Springer-Verlag, pp.339-361, 1999.

[5] S. Blake-Wilson and A. Menezes, "Unknown Key-Share Attacks on the Station-To-Station (STS) Protocol", Technical report CORR 98-42, University of Waterloo, 1998.

[6] I. F. Blake and G. Seroussi, *Elliptic Curves in Cryptography*, London Mathematical Society Lecture Note Series 265, Cambridge University Press, 1999.

[7] D. Boneh and R. Lipton, "Algorithms for Black-Box Fields and their Application to Cryptography", *Advances in Cryptology – Crypto '96*, LNCS 1109, Springer-Verlag, pp.283-297, 1996.

[8] J. Cheon, S. Park, C. Park and S. Hahn, "Scalar Multiplication on Elliptic Curves by Frobenius Expansions", *ETRI J.*, Vol.21, No.1, pp.27-38, March 1999.

[9] W. Diffie and M. E. Hellman, "New Directions in Cryptography", *IEEE Transaction on Information Theory*, 22, pp.644-654, 1976.

[10] W. Fumy, "Key Management Techniques", *COSIC'97 Course*, LNCS 1528, Springer-Verlag, pp.142-162, 1998.

[11] K. C. Goss, "Cryptographic Method and Apparatus for Public Key Exchange with Authentication", U.S. patent 4,956,865, September 1990.

[12] D. Johnson, Contribution to ANSI X9F1 working groups, June 1997.

[13] M. Just and S. Vaudenay, "Authenticated Multi-Party Key Agreement", *Advances in Cryptology – Asiacrypt '96*, LNCS 1163, Springer-Verlag, pp.36-49, 1996.

[14] B. Kaliski, Contribution to ANSI X9F1 and IEEE P1363 working groups, June 1998.

[15] L. Law, A. Menezes, M. Qu, J. Solinas and S. Vanstone, "An Efficient Protocol for Authenticated Key Agreement Protocol", Technical report CORR 98-5, University of Waterloo, Canada, March 1998.

[16] C. Lee, J. Lim, and J. Kim, "An Efficient and Secure Key Agreement", IEEE p1363a draft, 1998.

[17] C. Lim and P. Lee, "A Key Recovery Attack on Discrete Log-based Schemes using a Prime Order Subgroup", *Advances in Cryptology – Crypto '97*, LNCS 1294, Springer-Verlag, pp.249-263, 1997.

[18] U. Maurer and S. Wolf, "Diffie-Hellman Oracles", *Advances in Cryptology – Crypto '96*, LNCS 1109, Springer-Verlag, pp.283-297, 1996.

[19] T. Matsumoto, Y. Takashima and H. Imai, "On Seeking Smart Public-Key Distribution Systems", *Transaction of the IECE of Japan*, E69, pp.99-106, 1986.

[20] A. Menezes, P. van Oorschot and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

[21] National Security Agency, "SKIPJACK and KEA Algorithm Specification", Version 2.0, May 29, 1998.

[22] B. Song and K. Kim, "Two-Pass Authenticated Key Agreement Protocol with Key Confirmation", *Progress in Cryptology – Indocrypt 2000*, LNCS 1977, Springer-Verlag, pp.237-249, December 2000.

[23] Y. Yacobi, "A Key Distribution Paradox", *Advances in Cryptology – Crypto '90*, LNCS 537, Springer-Verlag, pp.268-273, 1991.

[24] ANSI X9.42, *Agreement of Symmetric Algorithm Keys using Diffie-Hellman*, working draft, May 1998.

[25] ANSI X9.63, *Elliptic Curve Key Agreement and Key Transport Protocols*, working draft, July 1998.

[26] IEEE P1363, *Standard Specifications for Public-Key Cryptography*, working draft, July 1998.

[27] ISO/IEC 11770-1, *Information Technology - Security Techniques - Key Management - Part 1: Key Management Framework*, draft, 1996.

[28] ISO/IEC 11770-2, *Information Technology - Security Techniques - Key Management - Part 2: Mechanisms Using Symmetric Techniques*, draft, 1996.

[29] ISO/IEC 11770-3, *Information Technology - Security Techniques - Key Management - Part 3: Mechanisms Using Asymmetric Techniques*, draft, 1996.

Acknowledgement

I am deeply grateful to my advisor, Prof. Kwnagjo Kim, for his assistance, encouragement and advice throughout the master's course. He has got me trained to develop my abilities for various aspects of research and project. I also express my appreciation to Prof. Chulsoo Lee and Prof. Junghee Cheon, committee members of my thesis, for approving this work and counseling of my study, which will give good help in my further works. I would like to extend my thanks to Prof. Yongjin Kwon, a teacher of my university, who instructed and guided me to have an interest in crytography. I owe Simon Blake-Wilson much gratitude for his kind answers when I had questions about his papers. His papers are references of my thesis.

I can't help forgetting my thanks to the associates of C&IS Lab.; W. Huh, M. Seo and J. Baek at SECUi.COM, K. Hwang at Ahnchulsu Lab., A. Jo at SDS, H. Kim, B. Lee, K. An, J. Go, H. Park, M. Lee, J. Kim, J. Park, and M. Kim. They influenced my study large or small.

Special thanks goes to Pastor Jinhan Jeon, a preacher of ICF service, and Pastor Seungryong Jung of Everlove Baptist Church, for constant praying, attending and offering words of cheer. There are a lot of people for me to appreciate, even though I don't mention their names one by one. They are relatives and members of Everlove Baptist Church, CCC, and ICU who have prayed for me.

Most of all, I thank from the bottom of my heart to my parents for their trust, prayer, support, and love, and to my lovely young brother.

Everything is due to God's grace. Thank you, my Lord! May I praise You and delight to do Your will all my life. Please, bless these people!.

"*Do your best to present yourself to God as one approved, a workman who does not need to be ashamed and who correctly handles the word of truth*". (2TIMOTHY 2:15)

Curriculum Vitae

Name : Boyeon Song

Date of Birth : Sep. 7. 1976

Sex : Female

Nationality : Korean

Education

1994.3–1998.2	Communications and Information Engineering Hankuk Aviation University (B.S.)
1998.3–2001.2	Information Security Group, School of Engineering Information and Communications University (M.S.)

Career

1999.3–1999.8	Apprentice Researcher at Korea Telecom
1999.9–1999.12	Apprentice Researcher of ICU Joint Research Center
1999.3–2001.2	Graduate Research Assistant
2000.3–2000.6	Graduate Teaching Assistant ICE514 Discrete Mathematics

Publications

Domestic Paper

(1) 2000.9 송보연, 김광조, “키 공유 확인이 가능한 새로운 키 합의 프로토콜 (2-pass Unilateral Authenticated Key Agreement Protocol with Key Confirmation)”, *WISC2000*, ’00.9.5∼9.7. 천안 상록 호텔

(2) 2000.10 송보연, 김광조, “타원곡선 상에서 Diffie-Hellman문제에 기반한 키 합의 프로토콜 (Key Agreement Protocol based on Diffie-Hellman Problem over Elliptic Curve)”, 한국정보처리학회(KIPS2000) 추계 학술발표 논문집, Vol.7, No.2, pp.785-788.

(3) 2000.11 송보연, 김광조, “Comparison of Existing Key Establishment Protocols”, KIISC 종합학술발표회(CISC2000), 종합 학술발표 논문집, Vol.10, No.1, pp.677-690.

International Paper

(4) 2000.12 Boyeon Song, Kwangjo Kim, “Two-Pass Authenticated Key Agreement Protocol with Key Confirmation”, *Progress in Cryptology – Indocrypt 2000*, LNCS 1977, Springer-Verlag, pp.237-249.

Domestic Patent

(5) 2000.11 송보연, 김광조, “안전한 디피 헬만 형 키 합의 프로토콜 구성 방법 및 장치 (Method and Apparatus to construct Secure Diffie-Hellman type Authenticated Key Agreement Protocol)”, 출원번호 2000-66177

(6) 2000.11 송보연, 김광조, "디피 헬만 형 키 공유 확인이 가능한 인증된 키 합의 프로토콜의 구성 방법 및 장치 (Method and Apparatus to construct Secure Diffie-Hellman type Authenticated Key Agreement Protocol with Key Confirmation)", 출원번호 2000-66178