Two-Pass Authenticated Key Agreement Protocol with Key Confirmation

Boyeon Song Kwangjo Kim

Information and Communications University (ICU)
58-4 Hwaam-dong, Yusong-gu, Taejon, 305-732, S. Korea
{bysong, kkj}@icu.ac.kr

Abstract. This paper proposes three key agreement protocols that emphasize their security and performance. First, the two-pass authenticated key agreement (AK) protocol is presented in the asymmetric setting, which is based on Diffie-Hellman key agreement working over an elliptic curve group and provides more desirable security attributes than the MTI/A0, two-pass Unified Model and two-pass MQV protocols. Other two protocols are modifications of this protocol: the three-pass authenticated key agreement with key confirmation (AKC) protocol which uses message authentication code (MAC) algorithms for key confirmation, and the two-pass authenticated key agreement protocol with unilateral key confirmation which uses the MAC and the signature.

1 Introduction

Key establishment protocol is a process whereby a shared secret key becomes available to participating entities, for subsequent cryptographic use. It is broadly subdivided into key transport and key agreement protocol[18]. 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. 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[18]. 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[18]. In this system the two entities share only public information that has been authenticated[6]. This paper is concerned with two-entity authenticated key agreement protocol working over an elliptic curve group in the asymmetric (public-key) setting.

The goals 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].

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. Informally speaking, a key agreement protocol is said to provide *implicit key authentication (IKA)* (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. A key agreement 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. 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, 14, 18].

Desirable security attributes of AK and AKC protocols are known-key security (a protocol should still achieve its goal in the face of an adversary who has learned some other session keys - unique secret keys which each run of a key agreement protocol between A and B should produce.), forward secrecy (if static private keys of one or more entities are compromised, the secrecy of previous session keys is not affected.), key-compromise impersonation (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 (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.), *etc*[4, 14, 18]. Desirable performance attributes of AK and AKC protocols include a minimal number of passes, low communication overhead, low computation overhead and possibility of precomputations[4].

This paper firstly proposes a new two-pass AK protocol in the asymmetric setting, which is based on Diffie-Hellman key agreement working over an elliptic curve group. The protocol provides more desirable security attributes than the MTI/A0, Unified Model and MQV AK protocols. Secondly, an AKC protocol from the AK protocol is derived by using the Message Authentication Code (MAC) algorithms. Finally, a two-pass unilateral AKC protocol is designed to provide mutual IKA and unilateral EKA, and known-key security, forward secrecy, key-compromise impersonation, and unknown key-share attributes discussed in [2, 6].

The remaining of this paper is organized as follows. Section 2 reviews AK protocols like the MTI/A0, Unified Model and MQV protocols. Section 3 presents the new AK protocol and its two variants; AKC protocol and two-pass unilateral AKC protocol. Section 4 compares key agreement protocols suggested in Sections 2 and 3. Finally, Section 5 makes concluding remarks.

2 AK protocols

This section describes the MTI/A0, two-pass Unified Model, and two-pass MQV protocols. The security of AK protocols in this paper is based on the *Diffie-Hellman problem*[10] in elliptic curve group: 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)[7] and there is strong evidence that the two problems are computationally equivalent (*e.g.*, see [8] and [16]).

All protocols in this paper have been described in the setting of the group of points on an elliptic curve defined over a finite field. The following abbreviations are used for clear understanding: IKA denotes implicit key authentication, EKA explicit key authentication, K-KS known-key security, FS forward secrecy, K-CI key-compromise impersonation, and UK-S unknown key-share[4].

We first present the elliptic curve parameters that are common to both entities involved in the protocols (*i.e.*, the *domain parameters*), and the key pairs of each entity. These are the same one used in [14].

Domain Parameter

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* $P \in E(\mathbf{F}_q)$[14].

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 [14] for more details). This process is called *domain parameter validation*.

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)[14]. A's static key pair and ephemeral key pair are denoted (W_A, w_A) and (R_A, r_A) respectively.

For the remainder 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[14].

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[14]. 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*[14]. For ease of explanation, public key validation is omitted from the descriptions of protocols of this section.

2.1 MTI/A0

The MTI/A0 key agreement protocol was proposed by Matsumoto *et al.* in 1986[17]. It was designed to provide implicit key authentication. The similar protocols are KEA[19] and those proposed by Goss[11] and Yacobi[20].

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.

2.2 Two-pass Unified Model

The Unified Model proposed by Ankney *et al.*[1] is an AK protocol in the draft standards ANSI X9.42[21], ANSI X9.63[22], and IEEE P1363[23]. In the protocol, the notation $||$ denotes concatenation.

Protocol 2

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 $(R_A, Cert_A)$ to B.
2. B generates a random integer r_B, $1 \le r_B \le n-1$, 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 can impersonate any other entity B to A.

2.3 Two-pass MQV

The MQV[14] is proposed by Law *et al.*, which is in the draft standards ANSI X9.42[21], ANSI X9.63[22], and IEEE P1363[23].

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, $1 \le r_A \le n-1$, computes the point $R_A = r_A P$, and sends $(R_A, Cert_A)$ to B.
2. B generates a random integer r_B, $1 \le r_B \le n-1$, 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[14].
- It does not provide UK-S attribute, which has recently been observed by Kaliski[13].

3 New Key Agreement Protocols

This section proposes the new two-pass AK, AKC and two-pass unilateral AKC protocols. Domain parameters and static keys are set up and validated as described in Section 2.

3.1 AK protocol

Protocol 4 provides more desirable security attributes than AK protocols described in Section 2.

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[14].

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 [15]. The check $K = \mathcal{O}$ ensures that K is a finite point[14].

A and B computes 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[5], heuristic arguments suggest that Protocol 4 provides mutual IKA[14]; 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 Section 1 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 elliptic curve Diffie-Hellman problem 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 other entity B to A, since the adversary still faces the elliptic curve Diffie-Hellman 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.

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 Y_A, an adversary E can't register A's public key Y_A as its own and subsequently deceive B into believing that A's messages are originated from E. In addition, it does not have the *duplicate-signature key selection (DSKS)* property[6]. 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 $r_A P$, $r_A W_B$ and $(w_A + r_A)R_B$. Refer to an efficient scalar multiplication method using Frobenius expansions suggested by Cheon *et al*[9]. Hence the work required by each entity is 3 (full) scalar multiplications. Since $r_A P$ and $r_A W_B$ can be computed off-line by A, the on-line work required by each entity is only 1 scalar multiplications. (if A and B do previously possess authentic copies of each other's static public key)

Instead of $k = kdf(c(r_A w_B + r_B w_A + r_A r_B)P)$, the following variations can be used as the session key.

(i) $k = kdf(c(r_A w_B + r_B w_A)P || c r_A r_B P)$.
A computes $k = kdf(c r_A W_B + c w_A R_B || c r_A R_B)$, and B does $k = kdf(c r_B W_A + c w_B R_A || c r_B R_A)$.

(ii) $k = kdf(c r_A w_B P || c r_B w_A P || c r_A r_B P)$.
A computes $k = kdf(c r_A W_B || c w_A R_B || c r_A R_B)$, and B does $k = kdf(c w_B R_A || c r_B W_A || c r_B R_A)$.

The protocol using these session keys requires 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 protocol.

3.2 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. Practical instantiations of $\mathcal{H}_1$ and $\mathcal{H}_2$ include $\mathcal{H}_1$=SHA-1$(01, z)$ and $\mathcal{H}_2$=SHA-1$(10, z)$[14].

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. 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) 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 [5] was derived from AK Protocol 3 of [5]. Protocol 2 of [5] was formally proven to be a secure AKC protocol. Heuristic arguments suggest that Protocol 5 has all the desirable security attributes listed in Section 1[14].

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].

3.3 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 from the responder, who has computed the secret key with information of the initiator and his own to use in message encryption. 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 is possible by only exchanging two flows.

Protocol 6 is a two-pass key agreement protocol which provides mutual IKA and unilateral EKA. It also provides many desirable security attributes. In 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) 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) 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) 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 entity authentication (the process whereby one entity is assured of the identity of a second entity involved in a protocol, and that the second has actually participated) of A to B. The MAC sent from B to A provides entity authentication of B to A and key confirmation 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.*, key confirmation 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 reasons as 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, because a flow is the dominant burden in performance.

4 Comparison

This section compares the security attributes and performance of all the protocols discussed so far. Table 1 presents the shared secret of Protocol 1 to Protocol 6, 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 (MTI/A0)	$(r_A w_B + r_B w_A)P$	$K = r_A W_B + w_A R_B$
Protocol 2 (Unified Model)	$(w_A w_B)P \| \| (r_A r_B)P$	$K = w_A W_B \| \| r_A R_B$
Protocol 3 (MQV)	$(r_A + \overline{R}_A w_A)(r_B + \overline{R}_B w_B)P$	$\overline{R}_A = R_A \bmod 2^{\lceil f/2 \rceil} + 2^{\lceil f/2 \rceil}$ $s_A = r_A + \overline{R}_A w_A \bmod n$ $K = s_A(R_B + \overline{R}_B W_B)$
Protocols 4,5,6 (proposed)	$(r_A r_B + r_A w_B + r_B w_A)P$	$K = r_A W_B + (w_A + r_A)R_B$

Table 1. The Shared Secret of AK protocols and A's required computations

Table 2 contains a summary of the services that are believed to be provided by the AK and AKC protocols discussed in Sections 2 and 3. 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 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 (quantities involving the entity's static and ephemeral keys and the other entity's static keys) and require 1 scalar multiplications with precomputations. Protocol 3 requires 2.5 and 1.5 scalar multiplications, respectively. As noted in Section 3, 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 two flows

Protocol	IKA	EKA	K-KS	FS	K-CI	UK-S
Protocol 1 (AK)	○	×	○	×	○	○
Three-pass MTI/A0 (AKC)	○	○	○	×	○	○
Protocol 2 (AK)	○	×	$\dagger^a$	$\dagger^b$	×	○
Three-pass Unified Model (AKC)	○	○	○	○	×	○
Protocol 3 (AK)	○	×	○	$\dagger^b$	○	×
Three-pass MQV (AKC)	○	○	○	○	○	○
Protocol 4 (proposed AK)	○	×	○	$\dagger^b$	○	○
Protocol 5 (proposed AKC)	○	○	○	○	○	○
Protocol 6 (proposed unilateral AKC)	○	▷	○	○	○	○

○ : the assurance is provided to A no matter whether A initiated the protocol or not.

× : the assurance is not provided to A by the protocol.

▷ : the assurance is provided to A only if A is the protocol's initiator.

$\dagger^a$: 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].

$\dagger^b$: 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 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 3. The number of scalar multiplications per entity of key agreement protocols

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 3, Protocol 4 provides more desirable security attributes than other AK protocols suggested so far. If EKA is additionally provided in AK 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 2 as well as the AKC MQV. The AKC MQV requires 2.5 scalar multiplication per entity without precomputations and 1.5 scalar multiplication 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.

5 Concluding Remarks

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; known-key security, forward secrecy, key-compromise impersonation and unknown key-share attributes.

Protocol 5 (AKC protocol) derived from Protocol 4 provides all the security requirement discussed in [6]; 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 (quantities of the entity's static and ephemeral keys and the other entity's static keys) is discounted, Protocol 5 requires less scalar multiplication than the MQV protocol.

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 has been shown informally by heuristic arguments to provide the claimed attributes of security. The following three conjectures are required to show that the security of the proposed protocols can be rigorously proved in Bellare-Rogaway model[3] of distributed computing as done in [5].

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

References

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

2. J. Baek and K. Kim, "Remarks on the Unknown Key Share Attacks", To appear Trans. of IEICE.
3. M. Bellare and P. Rogaway, "Entity Authentication and Key Distributions - the Three Party Case", *Advances in Cryptology – Crypto '93*, LNCS 773, Springer-Verlag, pp232-249, 1994.
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, pp339-361, 1999.
5. 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, pp30-45, 1997.
6. S. Blake-Wilson and A. Menezes, "Unknown Key-Share Attacks on the Station-To-Station (STS) Protocol", Technical report CORR 98-42, Univ. of Waterloo, 1998.
7. I. F. Blake and G. Seroussi, *Elliptic Curves in Cryptography*, London Mathematical Society Lecture Note Series 265, Cambridge University Press, 1999.
8. D. Boneh and R. Lipton, "Algorithms for Black-Box Fields and their Application to Cryptography", *Advances in Cryptology – Crypto '96*, LNCS 1109, Springer-Verlag, pp283-297, 1996.
9. J. Cheon, S. Park, C. Park and S. Hahn, "Scalar Multiplication on Elliptic Curves by Frobenius Expansions", ETRI J., Vol.21, No.1, pp27-38, March 1999.
10. W. Diffie and M. E. Hellman, "New Directions in Cryptography", IEEE Trans. on Information Theory, 22, pp644-654, 1976.
11. K. C. Goss, "Cryptographic Method and Apparatus for Public Key Exchange with Authentication", U.S. patent 4,956,865, September 11, 1990.
12. D. Johnson, Contribution to ANSI X9F1 working groups, June 1997.
13. B. Kaliski, Contribution to ANSI X9F1 and IEEE P1363 working groups, June 1998.
14. L. Law, A. Menezes, M. Qu, J. Solinas and S. Vanstone, "An Efficient Protocol for Authenticated Key Agreement Protocol ", Technical report CORR 98-5, Univ. of Waterloo, Canada, March 1998.
15. 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, pp249-263, 1997.
16. U. Maurer and S. Wolf, "Diffie-Hellman Oracles", *Advances in Cryptology – Crypto '96*, LNCS 1109, Springer-Verlag, pp283-297, 1996.
17. T. Matsumoto, Y. Takashima and H. Imai, "On Seeking Smart Public-Key Distribution Systems", Trans. of IEICE, Vol.E69, pp99-106, 1986.
18. A. Menezes, P. van Oorschot and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997
19. National Security Agency, "SKIPJACK and KEA Algorithm Specification", Version 2.0, May 29, 1998.
20. Y. Yacobi, "A Key Distribution Paradox", *Advances in Cryptology – Crypto '90*, LNCS 537, Springer-Verlag, pp268-273, 1991.
21. ANSI X9.42, *Agreement of Symmetric Algorithm Keys using Diffie-Hellman*, working draft, May 1998.
22. ANSI X9.63, *Elliptic Curve Key Agreement and Key Transport Protocols*, working draft, July 1998.
23. IEEE P1363, *Standard Specifications for Public-Key Cryptography*, working draft, July 1998.