Chapter 4

Pairing-free Two-Party Authenticated Key Agreement Protocol

A two-party authenticated key agreement (2PAKA) protocol using ECC and the self-certified public key (SC-PKC) of the participants is proposed in this chapter. Although several ECC-based 2PAKA protocols using either PKI or IBC have been proposed earlier, they suffer from certain limitations. For instances, the former requires heavy computation and management of public key certificates and the latter induces key escrow problem as the private key is generated by a trusted third party, known as PKG. Also the man-in-the-middle attack (MIMA) may occur from a malicious PKG and the resilience against such an attack for a secured key agreement protocol is needed. The proposed 2PAKA protocol removes all the limitations as mentioned above. Note that the SC-PKC concept was initially proposed by Girault, where a trusted third party, called System Authority (SA) generates the public key of a user and the corresponding private key, which is known to the user only, is generated interactively by the user and SA. This protocol is computationally efficient, secured against all known attacks and may be considered as an alternative of the PKI- or IBC-based 2PAKA protocol.

4.1 Introduction

A 2PAKA protocol establishes an authenticated and contributory secret session key between two entities through an open network. The Diffie and Hellman key exchange protocol [86] is the first such an approach that uses the exchange of two messages and generates a secret session key between two entities using very simple operations. However, this technique, due to lack of authentication of the entities, is vulnerable to the MIMA. Later on, several 2PAKA protocols to avoid MIMA have been proposed in the open literature and some of their recently proposed protocols are now described. The 2PAKA protocols proposed in [185, 186, 187] used the CA-PKC/PKI to circumvent the MIMA, however, due to need of execution of the time-consuming modular exponentiation operation, they are not suitable for resource constraint environments such as in sensor networks, mobile devices, smartcards, low-end PDAs, where the computation capability and battery life-time are limited.

The ECC-based 2PAKA protocols [14, 15, 188, 189] are more efficient than PKI based protocols, because instead of modular exponentiation, the elliptic curve scalar point multiplication is used. Also the ECC-based 2PAKA protocol is more secured as its security lies on the difficulty of solving the ECDLP, which is much more difficult than the discrete logarithm problem (DLP). Since the ECC-based protocol has security, computation and communication efficiencies, however it has some disadvantages. It needs a certificate authority (CA) to validate the public key certificates and thus requires additional processing and extra storage space to maintain and store the public keys and certificates. Recently, several IBC-based 2PAKA protocols using ECC have been proposed in [45, 54, 55, 56, 68, 70, 71, 72, 74, 75, 76, 77, 78, 81, 82, 190, 191]. However, the IBC-based 2PAKA protocols suffer from the private key escrow problem as the private key is known to PKG and the MIMA may occur from a malicious PKG. In 1991, the SC-PKC proposed by Girault [37] appears to be better than the PKI or IBC, where a trusted third party SA generates user's public key by signing the identity

of the user using own secret key and user's private key is computed by the corresponding user. The advantage of the SC-PKC is that the authenticity of an user's public key can be verified publicly without using any separate certificate issued by SA and the private key known to the user only, so the private key escrow problem is resolved. As compared with conventional public key systems (e.g., CA-PKC or IBC), SC-PKC requires lower communication overheads, storage space, and computation efforts since no certificate is required. In this chapter, we proposed a 2PAKA protocol using SC-PKC for efficient and secure peer-to-peer communication over insecure and hostile networks.

The rest of the chapter is organized as follows. The Section 4.2 describes the details of the proposed 2PAKA protocol. In Section 4.3, the security analysis and the comparison of our proposed protocol with other comparable protocols are addressed. The estimation and the computation efficiency of the proposed protocol over relevant protocols are given in Section 4.4 and finally, the Section 4.5 concludes the chapter.

4.2 Proposed 2PAKA Protocol

The proposed 2PAKA protocol consists of three phases - Setup phase, User registration phase and Key agreement phase, the details of them are described below.

4.2.1 Setup Phase

In this phase, SA generates the system's parameter Ω. For this, it selects a security parameter $k \in Z^+$ and an elliptic curve group G_q (Eq. 3.1), defined over the finite field F_q of prime order q, where P is a base point, is a large prime number. The SA selects an integer $s \in_R Z_q^*$ as his private key and computes the corresponding public key as $P_S = sP$. It chooses a secure one-way hash function $H(\cdot)$ (e.g., SHA-1), a key derivation function $kdf : \{0,1\}^* \rightarrow \{0,1\}^k$ and finally, SA publishes the system's parameter $\Omega = \{F_q, E/F_q, G_q, P, P_S, H, kdf\}$.

4.2.2 User Registration Phase

When a user i with identity ID_i is going to join the system, he selects a number $x_i \in_R Z_q^*$ and then computes

$$X_i = H(ID_i \parallel x_i)P \tag{4.1}$$

where "$\parallel$" represents the concatenation operation. Now the user ID_i sends (ID_i, X_i) to SA through a secure channel. The SA chooses $t_i \in_R Z_q^*$ for ID_i and then computes

$$P_i = H(ID_i \parallel t_i)P_S + X_i \tag{4.2}$$

$$r_i = [H(ID_i \parallel t_i) + H(ID_i \parallel P_i)]s \bmod q \tag{4.3}$$

Now the SA sends (ID_i, P_i, r_i) to the user ID_i through a secure channel. Upon receiving (ID_i, P_i, r_i) from SA, user ID_i computes his private key

$$d_i = [r_i + H(ID_i \parallel x_i)] \bmod q \tag{4.4}$$

and verifies the validity of (ID_i, P_i, r_i) by checking that

$$d_i P = P_i + H(ID_i \parallel P_i)P_S \tag{4.5}$$

If the equation (4.5) holds, then the user ID_i accepts d_i as his private key and computes $Q_i = P_i + H(ID_i \| P_i)P_S$ as his public key. After registration, SA publishes the public key Q_i of the user ID_i. It is worth to note that, SA needs not issue any extra certificate with respect to Q_i. The verification of the equation (4.5) is as follows

$$
\begin{aligned}
Q_i &= d_i P \\
&= (r_i + H(ID_i \parallel x_i))P \quad [Eq.\ (4.4)] \\
&= [H(ID_i \parallel t_i) + H(ID_i \parallel P_i)]sP + H(ID_i \parallel x_i)P \quad [Eq.\ (4.3)] \\
&= H(ID_i \parallel t_i)sP + H(ID_i \parallel P_i)sP + H(ID_i \parallel x_i)P \\
&= H(ID_i \parallel t_i)P_S + H(ID_i \parallel P_i)P_S + X_i \quad [Eq.\ (4.1)] \\
&= H(ID_i \parallel t_i)P_S + X_i + H(ID_i \parallel P_i)P_S \\
&= P_i + H(ID_i \parallel P_i)P_S \quad [Eq.\ (4.2)]
\end{aligned}
$$

The details of the registration phase is given in Fig. 4.1.

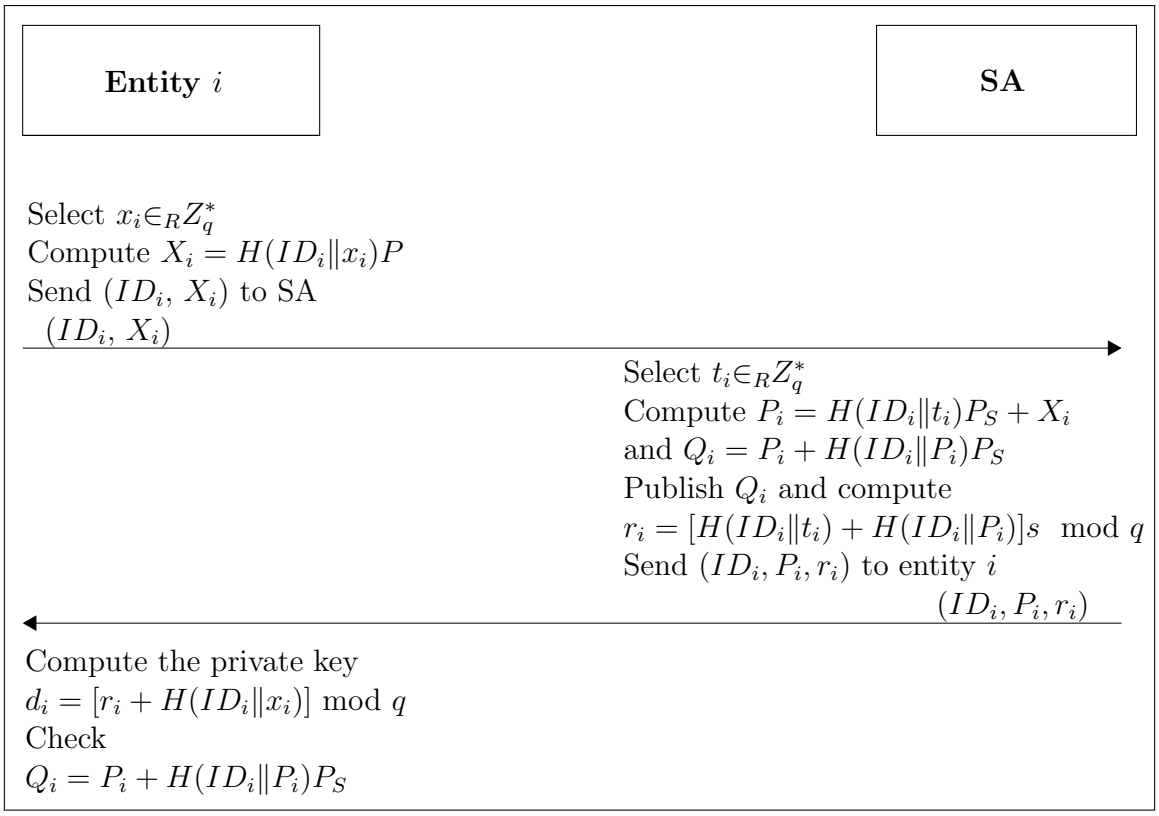

Figure 4.1: Entity registration phase of the proposed protocol.

4.2.3 Key Agreement Phase

Assume that two entities A and B want to establish a secret session key between them. We assume that the entity A acts as an initiator and the entity B is a responder. Now the following two rounds are executed to establish a secret session key between them.

Step 1. Entity A performs the followings:

(a) Select a number $a \in_R Z_q^*$, compute $T_A = aQ_A$ and $R_A = H(T_A \| d_A Q_B)$.

(b) Send (ID_A, T_A, R_A) to B.

Step 2. On receiving (ID_A, T_A, R_A) from A, B will:

(a) Choose a number $b \in_R Z_q^*$, compute $T_B = bQ_B$ and $R_B = H(T_B \| d_B Q_A)$.

(b) Send (ID_B, T_B, R_B) to A.

Now entity A computes $R_B^* = H(T_B \| d_A Q_B)$ and compares with received R_B, and if it is hold, i.e., if $R_B^* = R_B$, then A computes the partial session key as

$$\begin{aligned} K_A &= ad_A T_B \\ &= ad_A bd_B P \\ &= abd_A d_B P \end{aligned}$$

Similarly, entity B computes $R_A^* = H(T_A \| d_B Q_A)$ and compares with received R_A. If $R_A^* = R_A$, then B computes the partial session key as

$$\begin{aligned} K_B &= bd_B T_A \\ &= bd_B ad_A P \\ &= abd_A d_B P \end{aligned}$$

After successful completion of the above processes, entities A and B generate a common session key $SK = kdf(ID_A \| ID_B \| Trans \| K)$, where $K = K_A = K_B$ and $Trans = (T_A \| T_B \| R_A \| R_B)$. Detail of the proposed protocol is illustrated in Fig. 4.2.

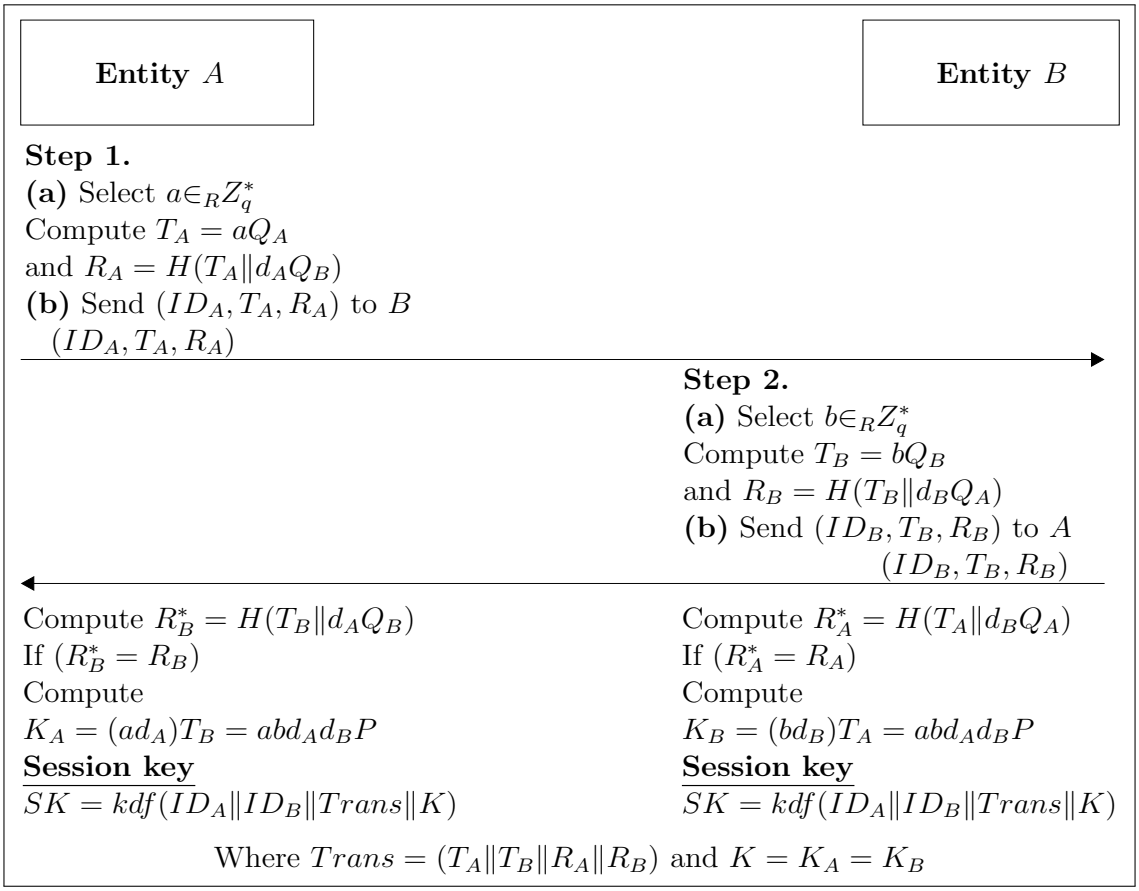

Figure 4.2: Key agreement phase of the proposed protocol.

4.3 Security Analysis of the Proposed Protocol

The security analysis of the proposed protocol is given in this section. First of all, it supports both the implicit and explicit key authentication properties. Because the proposed protocol instead of two short-term secrets a and b, exchanges $T_A = aQ_A$ and $T_B = aQ_B$ with their corresponding signatures R_A and R_B of the entities A and B through a public channel. Since the adversary $\mathscr{A}$ cannot forge the signatures R_A and R_B without the knowledge of static private keys d_A or d_B, so $\mathscr{A}$ cannot impersonate to any entity and would not be able to compute the final session key, and thus confirms the correct session key of the participants. So the implicit key authentication property is provided by the proposed protocol. Also it supports explicit key

authentication property as according to Blake-Wilson et al. [95] that a key agreement protocol supports explicit key authentication if it has both session key confirmation and implicit key agreement properties. The proposed protocol also satisfies other security attributes, for instances, *man-in-the-middle attack*, *known-key secrecy*, *key-compromise impersonation resilience*, *unknown key-share resilience*, *perfect forward secrecy*, *known session-specific temporary information attack*, *no key control*, etc., the details of which can be found in [95]. The detailed analysis of the proposed protocol to support the above security attributes are given below.

4.3.1 Man-in-the-middle Attack

As stated, the proposed protocol exchanges $T_A = aQ_A$ and $T_B = bQ_B$ along with the signatures R_A and R_B, and generates the session key $SK = kdf(ID_A \| ID_B \| Trans \| K)$ using two static private keys d_A and d_B, and two short-term keys a and b of the participants. Since the users can authenticate T_A and T_B using R_A and R_B very easily, a valid session key SK is generated. Note that the man-in-the-middle attack is only possible in the proposed technique if an adversary can forge R_A and R_B and/or compute $d_A d_B P$ from the pair $(Q_A, Q_B) = (d_A P, d_B P)$ which is not possible as it is a hard computational Diffie-Hellman problem (CDHP). Thus, the proposed protocol protects the man-in-the-middle attack.

4.3.2 Known-key Attack

The 2PAKA protocol satisfies the known-key security if the knowledge of previously computed session key does not allow an adversary to compromise the past or future session keys. Assume that a previous session key generated by the proposed protocol is disclosed to an adversary $\mathscr{A}$. However, $\mathscr{A}$ is unable to derive all past and future session keys from the knowledge of the disclosed session key. To derive a past session key $SK = kdf(ID_A \| ID_B \| Trans \| K)$, $\mathscr{A}$ has to compute the partial session key $K =$

$K_A = K_B = abd_Ad_BP$ of that session, which depends on two ephemeral secrets a and b, and the private keys d_A and d_B of the participating entities. Since T_A and T_B of each session are known to $\mathscr{A}$, and until (d_A, d_B) and (a, b) are computed from (Q_A and Q_B) and (T_A and T_B), respectively, no past or future session keys are compromised. Because these computations involve solving ECDLP in polynomial time, which is not possible as no such algorithm exists in reality. Therefore, the known-key security (K-KS) property is preserved in the proposed protocol.

4.3.3 Key-compromise Impersonation Attack

In key-compromise impersonation (K-CI) attack, an adversary $\mathscr{A}$ having the knowledge of long-term private key of a participant, say d_A of the participant A, may impersonate other participant B to A and try to obtain the correct session key established between the participants. A well-sound authenticated key agreement protocol should resist this attack. Suppose that $\mathscr{A}$ who knows d_A can compute d_Ad_BP using B's public key $Q_B = d_BP$ and sends (ID_B, $T_B = bQ_B$, $R_B = H(T_B \| d_AQ_B)$) to A, where $b \in_R Z_q^*$ is selected by $\mathscr{A}$. Similarly the participant A computes (ID_A, T_A, R_A) and sends them to $\mathscr{A}$. However, in order to derive the session key $SK = kdf(ID_A \| ID_B \| Trans \| K)$, $\mathscr{A}$ must obtain $K = abd_Ad_BP$, which requires the knowledge of the private key d_B of B. Since due to difficult of the ECDLP, it is not possible to derive d_B from B's public key $Q_B = d_BP$, the correct session key SK cannot be computed by $\mathscr{A}$. Thus, the proposed protocol resists the key-compromise impersonation attack.

4.3.4 Unknown Key-share Attack

After successful completion of a key agreement protocol, an entity, say A out of two entities A and B, believes that a correct session key with entity B has been established but, the same may not be true to the entity B and he mistakenly believes that the session key instead of A has been established with an adversary $\mathscr{A}$. Note that this

cannot exist in the proposed protocol, because both the entities compute the common session key from the authentication tokens T_A and T_B validated by their signatures R_A and R_B, and due to ECDLP, the long-term private keys never be obtained from the public keys of the entities. Thus, the proposed protocol is immune from unknown key-share (U-KS) attack.

4.3.5 Perfect Forward Secrecy

A key agreement protocol satisfies forward secrecy (FS) if the secrecy of a previously generated session key is not compromised even if the private key of one or more entities but, not all are known to an adversary. And a protocol has perfect forward secrecy (PFS) property if an adversary having the knowledge of the private keys of all entities is unable to acquire any previously generated session key. Now if the long-term private keys d_A and d_B of A and B in the proposed protocol are disclosed to an adversary $\mathscr{A}$, the session key cannot be computed because $\mathscr{A}$ needs to derive the session ephemeral secrets a and b from T_A and T_B by solving ECDLP. Since it is not solvable with a polynomial-time bounded algorithm, therefore, our protocol supports PFS attribute.

4.3.6 Known Session-specific Temporary Information Attack

In 2001, Canetti and Krawczyk [94] initially investigated the known session-specific temporary information attack (KSSTIA) and later on, it is further studied by Cheng et al. [97], where it is pointed out that the security of the generated session keys should not be compromised even if the session ephemeral secrets a and b are leaked to an adversary $\mathscr{A}$. In the proposed protocol, both the entities A and B compute their session key using $SK = kdf(ID_A \| ID_B \| Trans \| K)$. Note that $\mathscr{A}$ can derive the session key SK if he knows the partial session key $K = abd_Ad_BP$. But $\mathscr{A}$ cannot derive K even if the session short-term secrets a and b are disclosed, because d_A and d_B are not known to $\mathscr{A}$. Also $\mathscr{A}$ cannot derive directly from the pair $(T_A, T_B) = (ad_AP, bd_BP)$ because

it is required to solve the CDHP problem and it is not breakable by a polynomial-time bounded algorithm. Furthermore, $\mathscr{A}$ cannot use the pair (Q_A, Q_B) to derive the session key SK, because it needs to compute $(d_A d_B P)$ from $(Q_A, Q_B) = (d_A P, d_B P)$, which is difficult as to solve the CDHP. Thus, we may conclude that the proposed protocol sustains against known session-specific temporary information attack.

4.3.7 Key Off-set Attack/Key Replicating Attack

The key off-set/key replicating attack (KOA/KRA) is a variation of the man-in-the-middle attack, where an active adversary intercepts and modifies the messages exchanged between two entities in a session, and enforces the entities to agree upon a wrong session key, although this attack does not allow the adversary to gain any knowledge of the agreed session key. It is also a violation of the key integrity property, which indicates that any accepted session key should depend on the inputs, exchanged using the protocol. Blake-Wilson et al. [95] pointed out that a two-flow authenticated key agreement protocol without key conformation is vulnerable to KOA/KRA. The entities A and B in the proposed protocol exchange (ID_A, T_A, R_A) and (ID_B, T_B, R_B) between each other and an active adversary $\mathscr{A}$ can easily offset some of these values, say T_A and T_B by an unknown exponent ϵ and produces ϵT_A and ϵT_B. Nevertheless $\mathscr{A}$ cannot compute $R_A^* = H(\epsilon T_A \| d_A Q_B)$ and $R_B^* = H(\epsilon T_B \| d_B Q_A)$, because the derivation of $d_A Q_B$ or $d_B Q_A$ requires the knowledge of the long-term static private keys d_A or d_B of the entities. Therefore, the entities A and B easily detect this attack using R_A and R_B, and hence, the KOA/KRA attack is infeasible in the proposed protocol.

4.3.8 No Key Control

As stated earlier, both the entities in the proposed protocol generate a common secret session key $SK = kdf(ID_A \| ID_B \| Trans \| K)$. Since $K = abd_A d_B P$ and the values of a, b in K are chosen by the entities A and B randomly, so neither an entity nor an

adversary can enforce the session key to be a predetermined value and/or lie within a set having small number of elements. Hence, we claim that the proposed protocol provides no key control (NKC) attribute.

4.3.9 Reflection Attack

The proposed protocol is free from reflection attack (RA) and unknown key-share (U-KS) attack, because according to Boyed and Choo [70], the identities of the participating entities are included in the key derivation function (kdf). Also the proposed protocol provides freshness and data origin authentication as the transcript $Trans$ is included in the kdf function.

The proposed protocol in terms of security attacks as mentioned above has been compared with a number of protocols proposed recently and the outputs are given in Table 4.1. As shown, none of protocols except our proposed one can protect all the attacks. However, the protocols [54, 57, 192] are comparatively efficient as they cannot defend a single attack. Note that the KOA/KRA are only protected by the two protocols namely our proposed technique and Choie et al. [57]. The reason of the failure of the remaining protocols is that they exchange messages in the form aP or aQ_A without their signatures as according to Blake-Wilson et al. [95], and McCullagh-Barrato [56], until the two-flow authenticated key agreement protocols support key conformation property, the KOA/KRA is not protected.

4.4 Performance Evaluation of the Proposed Protocol

The performance of any key agreement protocol mainly depends on the execution of the following two decisive factors, and a brief introduction of them is given below:

Table 4.1: Security comparisons.

Protocols	PKG-FS	K-CI	PFS	MIMA	K-KS	RA	KSSTIA	KOA	NKC	IA	U-KS
Chen-Kudla [54]	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Choie-I [57]	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Choie-II [57]	Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes
Choie-III [57]	Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes
Hölbol-Welzer [77]	Yes	No	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Yes
Hsieh et al. [74]	Yes	No	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
McCullagh-Barreto [56]	No	No	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Kudla-Paterson [192]	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Ryu et al. [69]	Yes	No	Yes	Yes	Yes	No	Yes	No	Yes	Yes	Yes
Shim [68]	Yes	Yes	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes
Smart [55]	No	No	No	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Xie [71]	Yes	No	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes
Proposed	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

PKG-FS: PKG Forward Secrecy; **IA:** Impersonation Attack; **Yes:** Protect the attack; **No:** Do not protect the attack.

- **Computation Cost:** It is the execution cost required to perform different operations and their repetition involved in generating a common session key between the entities. Since different key agreement techniques follow different operations and accordingly the operations like elliptic curve point addition, simple hash operation, etc., require comparatively lesser computation cost than the operations such as bilinear pairing, elliptic curve scalar point multiplication, modular exponentiation, etc. The computation cost of 2PAKA protocols can be reduced if a fewer numbers of expensive operations are used in their implementation. Such protocols have an advantage that they are the most suitable for resource-limited (e.g., power, storage, bandwidth, etc.) environments such as in smartcards, mobile networks, etc.

- **Communication Cost:** The communication cost is another important factor for measuring the performance of a 2PAKA protocol. It includes the number of rounds, the number of steps per round and message-length used by the entities for establishing the authenticated session key between them. The higher is amount of communication costs required means to spend more time by participating entities to establish the session key. The increase in communication cost leads to more communication latency and thus involves more delay in transmit-response phase of the users. For this reasons a 2PAKA protocol with high communication cost is unsuitable for telecommunication system such as online pay-TV, online money transaction, online e-voting, etc., that needs a quick response for any required service.

In order to calculate the computation cost, the type and number of operations involved in some relevant protocols are given below.

Hölbl and Welzer protocol-I [77] requires each entity to perform 4 modular exponentiations, 2 modular multiplications and for Hölbl and Welzer protocol-II [77] requires 3 modular exponentiations, 2 modular multiplications. Chen and Kudla's protocol [54] executes 2 elliptic curve scalar point multiplications, 1 elliptic curve point addition and 2 pairing operation per entity. Smart's 2PAKA protocol [55] established the session key

with executing 2 scalar point multiplications, 2 pairing operations (one of which can be partially pre-computed, thus consider its cost equivalent to 1 scalar point multiplication), 1 pairing operation and 1 pairing-based exponentiation operation per entity. McCullagh and Barreto's protocol [56] requires 1 pairing operation, 2 elliptic curve scalar point multiplications and 1 pairing-based exponentiation operation per entity. Wang et al.'s protocol (improved Ryu's protocol) [81] employs 2 elliptic curve scalar point multiplications and 1 pairing operation. Choie et al. protocol-I [57] executes 2 elliptic curve scalar point multiplications and 3 elliptic curve point additions per entity and Choie et al.'s protocol-II (modification of Smart's protocol) [57] requires 2 elliptic curve scalar point multiplications and 2 elliptic curve point additions per entity. Cao et al.'s protocol [15] requires 5 elliptic curve scalar point multiplications and 2 elliptic curve point additions per entity. Wang et al.'s protocol [193] is implemented using modified Weil pairing or Tate pairing and executes 1 elliptic curve scalar point multiplication, 1 pairing operation, and 1 pairing-based exponentiation operation per entity.

It may be mentioned that in a pairing based 2PAKA, either Tate pairing or Weil pairing is used to evaluate bilinear pairing operation $\hat{e}$:$G_1 \times G_1 \rightarrow G_2$, where G_1 is an additive group on elliptic curve E/F_p defined over F_p and its order is q, a 160-bit prime number and G_2, in order to provide an equivalent level of security to that of 1024-bit RSA with a 512-bit prime p [191], is a q-order (160 bit) subgroup of the multiplicative group of the finite field $F^*_{p^2}$. As an estimation of pairing cost, one bilinear pairing, according to [36, 194], requires two to three times more multiplications than a elliptic curve scalar point multiplication, and also it has been seen from Cao et al.'s experimental result [15] that the execution of one pairing-based exponentiation is approximately equal to one-half of the time needed to execute one pairing operation.

In order to estimate the communication cost, we assume that the length of the identity is 16 bits and the output of the hash function (e.g., SHA-1) is 160 bits, and according to Cao et al.'s protocol [15], the longest message, which contains two points

in elliptic curve group and one identity, requires $(2 \times 160 + 16)/8 = 42$ bytes bandwidth for communication. We define different time complexity notations and their conversions in terms of T_{ML} as shown in Table 4.2. The time complexity, number of rounds and bandwidth requirements of different relevant protocols have been calculated and shown in Table 4.3.

Table 4.2: Definition and conversion of various operation units

Notations	Definition and conversion
T_{ML}	Time complexity for executing the modular multiplication
T_{EX}	Time complexity for executing the modular exponentiation, $T_{EX} \approx 240T_{ML}$ [35, 194]
T_{EM}	Time complexity for executing the elliptic curve scalar point multiplication, $T_{EM} \approx 29T_{ML}$ [35, 194]
T_{BP}	Time complexity for executing the bilinear pairing operation, $T_{BP} \approx 3T_{EM} \approx 87T_{ML}$ [36, 194]
T_{PX}	Time complexity for executing pairing-based exponentiation operation, $T_{PX} \approx 43.5T_{ML}$ [14, 15]
T_{EA}	Time complexity for executing the addition of two elliptic curve points, $T_{EA} \approx 0.12T_{ML}$ [35, 194]
T_{MTP}	Time complexity for executing the map-to-point function, $T_{MTP} \approx T_{EM} \approx 29T_{ML}$ [96]
T_{IN}	Time complexity for executing the modular inversion operation, $T_{IN} \approx 11.6T_{ML}$ [195]
T_H	Time complexity for executing the simple hash function, which is negligible [194]
T_X	Time complexity for executing a XOR operation, which is negligible

Table 4.3: Performance comparisons.

Protocol	No. of Rounds	Bandwidth (bytes)	Computation Cost
Cao et al. [15]	2	$(16 + 2 \times 160)/8 = 42$	$10T_{EM} + 2T_{EM} \approx 290T_{ML}$
Chen-Kudla [54]	2	$(16 + 512)/8 = 66$	$4T_{BP} + 4T_{EM} + 2T_{EA} \approx 464T_{ML}$
Choie et al.-I [57]	2	$(16 + 512)/8 = 66$	$4T_{BP} + 6T_{EM} \approx 522T_{ML}$
Choie et al.-II [57]	2	$(16 + 512)/8 = 66$	$4T_{BP} + 8T_{EM} \approx 580T_{ML}$
Hölbl-Walzer-I [77]	2	$(16 + 2 \times 1024)/8 = 258$	$8T_{PX} \approx 368T_{ML}$
Hölbl-Walzer-II [77]	2	$(16 + 2 \times 1024)/8 = 258$	$6T_{PX} \approx 261T_{ML}$
Kudla-Paterson [192]	2	$(16 + 4 \times 1024)/8 = 514$	$6T_{PX} \approx 261T_{ML}$
McCullagh-Barreto [56]	2	$(16 + 512)/8 = 66$	$2T_{BP} + 4T_{EM} + 2T_{PX} \approx 379T_{ML}$
Smart [55]	2	$(16 + 2 \times 512)/8 = 132$	$2T_{BP} + 2T_{EM} + 2T_{PX} \approx 321T_{ML}$
Wang et al. [81]	2	$(16 + 512)/8 = 66$	$2T_{BP} + 4T_{EM} \approx 292T_{ML}$
Wang et al. [193]	2	$(16 + 2 \times 512)/8 = 132$	$2T_{BP} + 2T_{EM} + 2T_{EX} \approx 714T_{ML}$
Zhu et al. [16]	3	$(16 + 4 \times 160)/8 = 82$	$12T_{EM} \approx 348T_{ML}$
Proposed	2	$(16 + 2 \times 160)/8 = 42$	$6T_{EM} + 4T_{EA} \approx 176T_{ML}$

4.5 Chapter Summary

A two-party authenticated key agreement protocol based on ECC and the self-certified public keys of the participants, where the security of the session key lies on the difficulties of solving the CDHP problem is proposed. The detail security analysis of the proposed 2PAKA protocol against several known cryptographic attacks is made and it has been found that all attacks are protected. The proposed protocol is also efficient in terms of storage-space, bandwidth requirements and the computation costs, which thus efficiently usable as an alternative protocol to the PKI- of IBC-based 2PAKA protocol and suitable for peer-to-peer communication in resource constrained environments.

In the next chapter i.e., Chapter 5, we devised an ECC- and IBC-based authenticated group key agreement protocol usable for group communication in imbalanced mobile networks.