Certificateless Cross-Domain Group Authentication Key Agreement Scheme Based on ECC

Liling Cao, Mei Liang, Zheng Zhang, and Shouqi Cao

Department of Engineering Science and Technology, Shanghai Ocean University, Shanghai 201306, China

Correspondence should be addressed to Shouqi Cao; 2212157493@qq.com

Received 10 February 2022; Revised 14 October 2022; Accepted 15 November 2022; Published 5 December 2022

Copyright © 2022 Liling Cao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Focusing on the problem that existing traditional cross-domain group authentication schemes have a high complexity, a certificateless cross-domain group authentication key agreement scheme based on ECC is proposed. The protocol provides scalability and can meet the requirements of cross-domain key negotiation by multiple participants in different domains. Security analysis shows that the proposed scheme is secure in the random oracle security model, it can resist some attacks under the extended Canetti-Krawczyk (eCK) security model. Performance analysis shows that the proposed scheme is of strong practical application value with high efficiency; it costs relatively low amount of calculation and communication.

1. Introduction

The development of technologies such as wireless networks, multicast, and distributed computing has brought new demands for group-oriented network applications, such as multiparty video conferencing, remote video teaching, and online games. As a typical application scenario in the above applications, cross-domain group communication can realize information exchange and transmission between remote cross-domain group members and will provide users with richer services and maximize the use of resources. However, the increase in the size of group members and the heterogeneous network access caused by cross-domains have also brought new security challenges to the design of user identity authentication systems. A secure cross-domain group authentication key agreement protocol will establish a shared key for remote cross-domain group members, establish a secure cross-domain communication channel, ensure the confidentiality and integrity of cross-domain group communication data, and effectively prevents attackers from stealing, tampering, and forging communication data [1–4].

The design of the existing cross-domain group authentication key agreement protocol mainly relies on three types of cryptosystems: public key infrastructure, identity-based cryptosystems, and certificateless public key cryptosystems. Public key infrastructure (PKI) is an important practical cryptographic technology to ensure network security. In PKI-based identity authentication, the certificate service system binds the digital certificate to the public key of the communicating entity, and the communicating entities verify the authenticity of the certificate to perform identity authentication. WF et al. [5] adopted elliptic curve cryptosystem based on threshold scheme to construct enterprise cross-domain authentication system with the help of virtual bridge CA model. However, due to the high interaction cost caused by the threshold scheme by splitting key factors, the expandability of joining and withdrawing members is not strong. Bin [6] improved the existing certificate revocation mechanism, but the certificate verification needs to detect from the book to be verified to the root certificate, so that the verification path is too long and the path verification efficiency is low, which greatly affects the application scope of the cross-domain identity authentication technology. Basin et al. proposed a new PKI architecture, ARPKI [7], which was designed by using formal models to ensure the transparency and reliability of certification-related operations (such as certificate issuance, update, revocation, and verification) and effectively deal with security events such as key loss or disclosure. Zhicheng et al. improved the traditional PKI-based cross-domain authentication by using the alliance
chain technology [8] and used the alliance chain to manage
the license domain, which simplified the authentication sig-
nature steps between entities and had high scalability. Dong
et al. proposed a cross-domain authentication scheme that
can enhance trust between hospitals [9], which solves the
problem of fragmentation and isolation caused by tradi-
tional hospitals maintaining their own PKI-based informa-
tion systems and provides better privacy protection
services while realizing the sharing of medical data. Chen
et al. used blockchain to replace part of CA functions [10];
the scheme has sufficient scalability and can meet the trust
transmission requirements of multiple PKI systems. As the
number of legitimate users increases, the calculation and
communication overhead of the certificate maintenance pro-
cess increases. Cross-domain identity authentication has
problems such as long trust paths, low certificate verifica-
tion efficiency, and complex interdomain trust path construc-
tion. It is not suitable for cross-domain identity authentication
of large-scale group members.

In identity-based cryptosystems, there is no need for PKI
to verify users’ public keys and identities. Private key gener-
ator (PKG) is trusted to generate private keys for users,
which can effectively solve the problems such as the over-
head of public key certificate management [11]. Changhai-
yuan et al. [12] proposed an identity-based signature algorithm
to achieve cross-domain authentication by using elliptic curve
additive group, avoiding complex bilinear pairings. How-
ever, the scheme only analyzes the certification process
between the entity and the certification authority and does
not consider the extra cost of the certification authority
and local resources to verify each other’s legitimacy. The
identity-based authenticated key agreement protocol with-
out bilinear pairs proposed by Farash and Attari [13] sets
multiple independent PKG, and each PKG sets the private
key for the user under its jurisdiction. However, this proto-
col cannot resist temporary key leakage attack and im-
personation attack, and it cannot realize implicit key
authentication and key confirmation. Cao et al. [14] pro-
posed a key agreement protocol for identity-based authenti-
cation with hierarchical PKG, which uses bilinear pair
operation in its design, but the operation is not efficient
and it is difficult to resist basic impersonation attacks. Kefei
et al. [15] proposed an improved scheme on the basis of the
literature [14], but this scheme is difficult to resist the tem-
porary key disclosure attacks, and the operation efficiency
was low due to the use of bilinear pair operation in the
scheme. Since the user’s private key is completely deter-
mined by the key generation center, PKG can decrypt any
user’s information and forge any user’s signature. There
are key escrowing problems, and user identity multiplicity
occurs when cross-domain, making cross-domain identity
authentication extremely complicated.

Based on the certificateless public key cryptosystem, the
user’s private key is composed of two parts, that is, the par-
tial private key provided by KGC and the secret value
selected by the user. Neither KGC nor the user can generate
a complete private key independently, which solves the key
escrow problem in the identity-based cryptosystem. Litera-
ture [16, 17] proposed a cloud user identity authentication
scheme based on certificateless password system, but there
was no relevant research on cross-domain authentication.
Li et al. [18] proposed a cross-domain authentication key
exchange protocol in a wireless grid environment, but the
use of too much symmetric encryption causes a large
amount of computational overhead. In 2015, Sun et al.
[19] proposed a certificateless scheme, but it involves the
calculation of linear pairs, so the amount of calculation is huge.
In 2016, Cheng et al. [20] proposed a one-round certificate-
less authenticated group key agreement protocol for mobile
ad hoc networks, but Luo et al. [21] indicated that their pro-
tocol could not achieve user anonymity that was an impor-
tant aspect of user privacy protection and could not resist
known temporary key attack. In 2018, Yang et al. [22] pro-
posed a cross-domain certificateless key agreement protocol
for electronic health systems. Although it achieved the secu-
rity guarantees of dynamic user management, authentication,
and session keys, it did not achieve the real cross-
domain and could not resist known temporary key attack
indicated by Luo et al. [21]. In 2018, Semal et al. [23] pro-
posed a certificateless group authenticated key agreement
protocol for secure communication in untrusted UAV net-
works, and in 2020, Luo et al. [21] proposed a cross-
domain certificateless authenticated group key agreement
protocol for 5G network slicings. However, in 2022, Ren
et al. [24] indicated that the protocol proposed by Semal
et al. could not resist public key replacement attack and pro-
tocol proposed by Luo only had a secondary security level;
the malicious KGC could collude with some malicious users
to attack the protocol.

To sum up, the current cross-domain group communica-
tion solutions are mainly focused on the security issues
of users’ cross-domain communication between two
domains. Such solutions not only cannot meet the security
requirements of users’ cross-domain group communication,
but also the communication process is too cumbersome and
requires relatively high communication capabilities and
computing capabilities, so it cannot provide good guarantees
for the security of cross-domain group communication.
Therefore, an efficient and scalable key agreement scheme
for cross-domain group authentication is urgently needed
to solve and realize the communication security problem
of efficient cross-domain group authentication. Focusing
on the characteristics and requirements of the existing
cross-domain group communication, this paper proposes a
certificateless cross-domain group key agreement scheme
based on ECC.

2. Preliminaries

2.1. Notation. The symbols and their meanings involved in
the cross-domain group key negotiation scheme are shown in Table 1.

2.2. Security Model. In our proposed protocol, a novel eCK
security model presented by Lippold et al. [25] is adopted.
In the subsections, detailed descriptions about the security
model including the adversary model, attack game, and
security definition are explained.
Table 1: Notation used in this paper.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>A collection of all cross-domain participants</td>
</tr>
<tr>
<td>Di</td>
<td>The i-th domain</td>
</tr>
<tr>
<td>RA</td>
<td>The registry</td>
</tr>
<tr>
<td>Ppub</td>
<td>Public key of the system</td>
</tr>
<tr>
<td>KGCi</td>
<td>Key generation center for domain Di</td>
</tr>
<tr>
<td>SKi = (ki, μi)</td>
<td>The private key of KGCi</td>
</tr>
<tr>
<td>PKi = Ppubi</td>
<td>The public key of KGCi</td>
</tr>
<tr>
<td>uj</td>
<td>The j-th participant of the i-th domain Di</td>
</tr>
<tr>
<td>IDj</td>
<td>The identity of uj</td>
</tr>
<tr>
<td>skj = (sjx, xj)</td>
<td>The private key of the user uj</td>
</tr>
<tr>
<td>pkj = Pj</td>
<td>The public key of the user uj</td>
</tr>
<tr>
<td>ski</td>
<td>Intragroup key negotiated by the user of the i-th domain Di</td>
</tr>
<tr>
<td>SK</td>
<td>Cross-domain group key</td>
</tr>
<tr>
<td>⊥</td>
<td>Represents no message or an unknown value</td>
</tr>
</tbody>
</table>

2.2.1. Adversary Model. There are two types of adversaries. As a dishonest participant, adversary $\mathcal{A}_1$ has no idea of the master key of KGC but has the capability to replace the public key of any participant, which means $\mathcal{A}_1$ can replace the secret value $x_i$ of the participant with a value of his choice. As a malicious KGC, adversary $\mathcal{A}_2$ cannot replace the public key of any participant but can obtain the master key of KGC, which means $\mathcal{S}$ is easily accessible for $\mathcal{A}_2$.

The security model is defined by an attack game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A} \in \{\mathcal{A}_1, \mathcal{A}_2\}$. The adversary asks the challenger for a polynomial number of queries, while the challenger issues the replies using simulators and random oracles owned by him. Let $\Pi^m_{ij}$ represent a simulator of the challenger, which simulates the behavior of participant $i$ in the $m$-th session with intended participant $j$. Then, simulated as participant $i$, the simulator $\Pi^m_{ij}$ executes all the steps that participant $i$ should do in the protocol session. That is, the simulator $\Pi^m_{ij}$ takes private key of participant $i$ and messages transmitted from participant $j$ (or pseudo-$j$ personated by the adversary) as inputs and sends the corresponding outputs to the adversary.

**Definition 1** (accepted session). The session $\Pi^m_{ij}$ is accepted when it can generate the session key $skij^m$.

**Definition 2** (session identity). The session identity (sID) is denoted as the concatenation of the participants’ identities and messages in the session. For instance, sID$\Pi^m_{ij} = \{ID_i, ID_j, M1, M2\}$, where $M1$ is the message transmitted from $\Pi^m_{ij}$ and $M2$ is the message received by $\Pi^m_{ij}$.

**Definition 3** (matched session). Honest participants $i$ and $j$ are involved in the protocol session. The session $\Pi^m_{ij}$ matches $\Pi^m_{ij}$ when they have the same session identity.

2.2.2. Attack Game. Steps of the attack game are described as follows:

1. The challenger $\mathcal{C}$ executes the SETUP algorithm:

   \[
   \{s, \text{ system params}\} \xrightarrow{\text{challenger}} \text{SETUP}(k) \quad (1)
   \]

   For adversary $\mathcal{A}_1$, the challenger $\mathcal{C}$ sends system params to $\mathcal{A}_1$ but keeps $s$ in secret. For adversary $\mathcal{A}_2$, the challenger $\mathcal{C}$ sends system params and $s$ to $\mathcal{A}_2$.

2. The adversary asks the challenger $\mathcal{C}$ for a polynomial number of the following queries:

   (i) Create (ID$_i$); $\mathcal{C}$ generates private key/public key pair ($sk_i, pk_i$) of participant $i$

   (ii) Rs$_i$: $\mathcal{C}$ reveals to $\mathcal{A}$ the partial private key $s_i$ of participant $i$

   (iii) Rx$_i$: $\mathcal{C}$ reveals to $\mathcal{A}$ the secret value $x_i$ of participant $i$

   (iv) Rt$_i(\Pi^m_{ij})$: $\mathcal{C}$ reveals to $\mathcal{A}$ the ephemeral private key $t_i$ of participant $i$ in the session $\Pi^m_{ij}$

   (v) Rs: $\mathcal{C}$ reveals to $\mathcal{A}$ the master key of KGC. Then, $\mathcal{A}$ can obtain the partial private keys of all participants

   (vi) Rp$_k$: $\mathcal{C}$ replaces the public key of participant $i$ with the value chosen by $\mathcal{A}$, which means that the secret values of all participants can be set by $\mathcal{A}$

   (vii) Rsk$_ij(\Pi^m_{ij})$: $\mathcal{C}$ reveals to $\mathcal{A}$ the accepted session key $sk_{ij}$ if $\Pi^m_{ij}$ is accepted. Otherwise, $\mathcal{C}$ returns ⊥ to $\mathcal{A}$

   (viii) Send ($\Pi^m_{ij}, M)$: $\mathcal{A}$ (pseudoparticipant $j$) sends the message $M$ to the session $\Pi^m_{ij}$ (simulated participant $i$) and gets a reply according to the protocol. If $M = ⊥$, simulated participant $i$ of the session $\Pi^m_{ij}$ is an initiator. Otherwise, it is a responder

3. When deciding to end aforementioned queries, $\mathcal{A}$ chooses a fresh session (defined later) $\Pi^m_{ij}$ and asks a test ($\Pi^m_{ij}$) query. By tossing a fair coin with $b \in \{0, 1\}$, $\mathcal{C}$ replies the session key held by $\Pi^m_{ij}$ if $b = 1$, or a random string if $b = 0$

4. The adversary asks the challenger $\mathcal{C}$ for a polynomial number of the above queries about fresh session $\Pi^m_{ij}$

5. When terminating the game, $\mathcal{A}$ makes a guess bit $b'$. If $b' = b$, $\mathcal{A}$ wins the game. The advantage of $\mathcal{A}$ for winning the game is defined as $\text{Adv}_{\mathcal{A}}(k) = | \text{pr}[b = b'] - (1/2) |$
Definition 4 (fresh session against $\mathcal{A}_1$). The accepted session $\Pi_{ij}^m$ is fresh if none of the following condition holds:

(i) $\mathcal{A}_1$ raises the query $Rsk_{ij}(\Pi_{ij}^m)$ or $Rsk_{ij}(\Pi_{ij}^m)$ (if the matched session $\Pi_{ij}^m$ of $\Pi_{ij}^m$ exists)

(ii) Matching: if honest participant $j$ is engaged in $\Pi_{ij}^m$ that matches $\Pi_{ij}^m$, $\mathcal{A}_1$ either inquires both $R_s$ (or $R_S$) and $R_t(\Pi_{ij}^m)$ or both $R_s$ (or $R_S$) and $R_t(\Pi_{ij}^m)$

(iii) Not matching: if there is no session matched to $\Pi_{ij}^m$, $\mathcal{A}_1$ either inquires both $R_s$ and $R_t(\Pi_{ij}^m)$ or $R_s$ (or $R_S$)

Definition 5 (fresh session against $\mathcal{A}_2$).

(i) $\mathcal{A}_2$ raises the query $Rsk_{ij}(\Pi_{ij}^m)$ or $Rsk_{ij}(\Pi_{ij}^m)$ (if the matched session $\Pi_{ij}^m$ of $\Pi_{ij}^m$ exists)

(ii) Matching: if honest participant $j$ is engaged in $\Pi_{ij}^m$ that matches $\Pi_{ij}^m$, $\mathcal{A}_2$ either inquires both $R_x$ (or $R_p$) and $R_t(\Pi_{ij}^m)$ or both $R_x$ (or $R_p$) and $R_t(\Pi_{ij}^m)$

(iii) Not matching: if there is no session matched to $\Pi_{ij}^m$, $\mathcal{A}_2$ either inquires both $R_x$ and $R_t(\Pi_{ij}^m)$ or $R_x$ (or $R_p$)

According to the adversary model stated in Section 2.2.1, $x_i$ and $s_j$ are deemed to be knowable for $\mathcal{A}_1$ and $\mathcal{A}_2$ by $R_p$ query and $R_s$ query, respectively. Supposing that participant $A$ and $B$ want to establish a session key, the session is not fresh in seven cases for $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively, which is shown in Table 2. Then, the fresh session can appear in seven cases for $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively, as shown in Table 3. As a case of fresh sessions, type F11 1 can be regarded as F11 and type F11 1 1 can be regarded as F11.

2.2.3. Security Definition. A certificateless cross-domain group key agreement protocol is deemed authenticated key agreement (AKA) secure if no adversary except the protocol participants can get the session key. Detailed and accurate definition is as follows.

Definition 6. A protocol is secure when the following set of conditions is assumed:

1. In the presence of an adversary $\mathcal{A} \in \{\mathcal{A}_1, \mathcal{A}_2\}$, sessions $\Pi_{ij}^m$ and $\Pi_{ij}^m$ always agree on the same session key that distributed uniformly at random

2. For any adversary $\mathcal{A} \in \{\mathcal{A}_1, \mathcal{A}_2\}$, $\text{Adv}_{\mathcal{A}}(k)$ is negligible

3. Key Agreement Scheme for Cross-Domain Group Authentication

Figure 1 shows the member structure of cross-domain group authentication in the scheme. Suppose there are $n$ domains, each with $m$ members participating in cross-domain authentication, where $u^j(1 \leq i \leq n, 1 \leq j \leq m)$ represents any participant and let $U = \{u^j\}$ represent the collection of remote participants scattered across domains $D_i(1 \leq i \leq n)$ that intend to negotiate a shared session key. Let the symbol $U^j = \{u^1, u^2, \ldots, u^m\}$ represent the set of all participants in the same service domain $D_i$. The protocol consists of two phases: intradomain key negotiation and interdomain key negotiation.

3.1. System Initialization. This algorithm takes a security parameter $k \in Z_n^*$ as input, and the RA performs the following steps to generate a master secret key $s$ and a set of public parameter param.

(i) RA generates a group with prime order $p$ and determines a point $P$ of order $p$ as a generator of $G$

(ii) RA chooses a master secret key $s \in Z_n^*$ and computes the corresponding public key $P_{pub} = sp$

(iii) RA chooses seven cryptographic secure hash functions: $H1(\ast), H3(\ast), H4(\ast), H5(\ast), H6(\ast) \in Z_n^*$, $H2(\ast) \in \{0, 1\}^k, H7 : \{0, 1\}^* \times \{0, 1\}^* \times \cdots \times \{0, 1\}^k \times Z_n \times Z_n \rightarrow Z_n^*$

(iv) RA publishes the system generated parameters $\{F, p, E(F_p), G, P, P_{pub}, n, H1, H2, H3, H4, H5, H6, H7\}$ while keeping the master key $s$ secret

3.2. Registration Phase. The registration phase consists of KGC i registration and user registration.

3.2.1. KGC i registration. When KGC i applies to RA as the key generation center of domain $D_i$, it needs to register with RA. The algorithm is as follows:

1. Set secret value: KGC i randomly chooses $\mu_i \in Z_q^*$, calculates $P_{pubi} = \mu_i \cdot P$, and sets $\mu_i$ as a secret value

2. Extract partial private key: key generation center KGC i sends identity information ID i to RA, and RA picks a random number $\lambda i \in Z_n^*$ for KGC i; calculates $W_i = \lambda i \cdot P, l_i = H3(\text{ID} i, W, P_{pub}), \text{and } k_i = \lambda i + l_i \cdot s \mod n$; and issues $\{k_i, W_i\}$ to KGC i by secure channel

3. Set private key: KGC i sets $SK_i = (k_i, \mu_i)$ as his private key

4. Set public key: KGC i sets $PK_i = P_{pubi} = \mu_i \cdot P$ as his public key

3.2.2. User Registration. When user $u^j$ applies to KGC i for joining the domain as the $j$th member, the registration needs to be completed at KGC i. The algorithm is as follows:
(1) Set secret value: $u_j^x$ randomly chooses $xj \in \mathbb{Z}_n^*$, calculates $P_{ji} = xji \cdot P$, and sets $xj$ as a secret value.

(2) Extract partial private key: user $u_j^x$ sends identity information $ID_{ji}$ to $KGC_i$, and $KGC_i$ picks a random number $r_j \in \mathbb{Z}_n^*$ for $u_j^x$; calculates $R_j = r_j \cdot P$, $h_j = H3(ID_j^x, R_j, P_{ji})$, and $s_j = r_j + h_j \cdot \mu_i \bmod n$; and issues $\{s_j, R_j\}$ to $u_j^x$ by secure channel.

(3) Set private key: $u_j^x$ sets $sk_j = (s_j, x_j)$ as his private key.

(4) Set public key: $u_j^x$ sets $pk_j = P_j = x_j \cdot P$ as his public key.

3.3. Intradomain Key Negotiation. As shown in Figure 2, in the key negotiation phases of intradomain, all participants in set $U$ negotiate the session key of the intradomain group (no negotiation is required if there is only one user in the domain). Pick a participant randomly from all participants in domain $Di$ as the domain head $Hi$. If user $umi$ is selected as domain header $Hi$, users of domain $Di$ are classified into common user $u_j^x (1 \leq j \leq m-1)$ and domain header $umi$.

Step 1: $u_j^x$ randomly chooses $t_j, a_j \in \mathbb{Z}_n^*$ and calculates $T_j = t_jP$ and then sends message $M1 = \{ID_j^x, T_j, R_j\}$ to $umi$.

Step 2: $umi$ randomly chooses $tm, am \in \mathbb{Z}_n^*$ and calculates $Tm = tmP$ and then sends message $M2 = \{IDm_i, Tm, Rm\}$ to $u_j^x$. 

Table 2: Cases of not fresh session.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Type</th>
<th>Queries Known</th>
<th>Known</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matching</td>
<td>A1</td>
<td>RsA (or Rs)</td>
<td>xA</td>
</tr>
<tr>
<td></td>
<td>A2</td>
<td>RsB (or Rs)</td>
<td>xA</td>
</tr>
<tr>
<td></td>
<td>A3</td>
<td>RskA, B(Π_{A,B}^{m})</td>
<td></td>
</tr>
<tr>
<td>Type I adversary $A_1$</td>
<td>A4</td>
<td>RskB, A(Π_{B,A}^{m})</td>
<td></td>
</tr>
<tr>
<td>Not matching</td>
<td>A5</td>
<td>RsA</td>
<td>xA</td>
</tr>
<tr>
<td></td>
<td>A6</td>
<td>RsB (or Rs)</td>
<td>xA</td>
</tr>
<tr>
<td></td>
<td>A7</td>
<td>RskA, B(Π_{A,B}^{m})</td>
<td></td>
</tr>
<tr>
<td>Matching</td>
<td>A8</td>
<td>RxA (or Rpki)</td>
<td>sA</td>
</tr>
<tr>
<td></td>
<td>A9</td>
<td>RxB (or Rpki)</td>
<td>sA</td>
</tr>
<tr>
<td>Type II adversary $A_2$</td>
<td>A10</td>
<td>RskA, B(Π_{A,B}^{m})</td>
<td></td>
</tr>
<tr>
<td>Not matching</td>
<td>A11</td>
<td>RskB, A(Π_{B,A}^{m})</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A12</td>
<td>RxA</td>
<td>sA</td>
</tr>
<tr>
<td></td>
<td>A13</td>
<td>RxB</td>
<td>sA</td>
</tr>
<tr>
<td></td>
<td>A14</td>
<td>RskA, B(Π_{A,B}^{m})</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Cases of fresh session.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Type</th>
<th>Queries Known</th>
<th>Known</th>
<th>Queries Known</th>
<th>Queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matching</td>
<td>FI1</td>
<td>RsA</td>
<td>xA</td>
<td>xB</td>
<td>FII1</td>
</tr>
<tr>
<td></td>
<td>FI2</td>
<td>RtA</td>
<td>xA</td>
<td>xB</td>
<td>FII2</td>
</tr>
<tr>
<td></td>
<td>FI3</td>
<td>RsA</td>
<td>xA</td>
<td>xB</td>
<td>FII3</td>
</tr>
<tr>
<td></td>
<td>FI4</td>
<td>RtA</td>
<td>xA</td>
<td>xB</td>
<td>FII4</td>
</tr>
<tr>
<td></td>
<td>FI'</td>
<td>Rs</td>
<td>xA</td>
<td>xB</td>
<td>FII'</td>
</tr>
<tr>
<td>Not matching</td>
<td>FI5</td>
<td>RsA</td>
<td>xA</td>
<td>xB</td>
<td>FII5</td>
</tr>
<tr>
<td></td>
<td>FI6</td>
<td>RtA</td>
<td>xA</td>
<td>xB</td>
<td>FII6</td>
</tr>
</tbody>
</table>

(1) By exchanging messages with domain header $umi$, each common user $u_j^x (1 \leq j \leq m-1)$ proves that they are legitimate members of the same domain $Di$.
Each user contributes in group key generation.

Each head contributes in group key generation.

Step 3: $u_j$ calculates $k_{jm} = (H(3(ID_j, T_j, P_j) + sj + xj) \cdot (H(ID_j, T_m, P_m)T_m + (R_m + H(ID_j, T_m, P_m))P_{pub} i + P_m) + kbj = aj - P$ and then sends the signature message $\sigma_j = (kbj, k_{jm})$ to $um_i$.

Step 4: $um_i$ calculates $k_{mn} = (H3(ID_{mn}, T_m, P_m)tm + sm + xmn) \cdot (H5(ID_{mn}, T_j, P_j)T_j + (R_j + H3(ID_j, T_j, P_j))P_{pubj} + P_j)$. If $k_{mn} = k_{jm}$, $u_j$ is a legitimate member of domain $Di$.

(2) Domain header $um_i$ calculates the session key $sk_i$ of domain $Di$ based on the information received from the negotiation members and sends it to the negotiation members in encrypted form. The legitimate members who in the domain can calculate the session key.

Step 1: At first, $um_i$ calculates $k_{bm} = am \cdot P$, $sk_{jm} = H2(ID_j||ID_{mn}||T_j||T_m||k_{jm})$, $qm = H4(xm||sm)$, $mbj = H1(ID_j||ID_{mn}||am \cdot kbj \oplus qm)$, $dm = xm + sm \cdot H1(ID_j||ID_{mn}||qm \cdot P)$, $zam = am + dm \cdot bmj$, $hn = H5(k_{bm}, qm)$, $f_m = H6(sk_1m + sk_2m + \ldots + sk(m - 1) + amxmP)$, and $key_j = H5(am \cdot kbj, qm) \oplus f_m$, then calculates the session key $sk_j = H7(ID_j||ID_{mn}||\ldots||ID_{mn}||f_m||hm)$ of domain $Di$, and finally sends $\{ID_j, ID_{mn}, \ldots, ID_{mn}, key_j, hm, s_{mn}\}$ to $u_j$.

Step 2: $u_j$ receives the message $\{ID_j, ID_{mn}, \ldots, ID_{mn}, key_j, hm, s_{mn}\}$ sent by $um_i$ and then calculates $q = bmj \oplus H1(ID_j||ID_{mn}||aj \cdot k_{bm})$ and $k_j = zam \cdot P - bmj||P_{mn} + (R_n + H3(ID_{mn}, R_m, P_{mn})P_{pubj} \cdot H1(ID_{mn}||q \cdot P)$. If $H5(k_j, q) = hm$, $u_j$ calculates $f_j = key_j \oplus H5(k_j \cdot aj, q)$. Finally, the intradomain session key $sk_j = H7(ID_j||ID_{mn}||\ldots||ID_{mn}||f_j||hm) = H7(ID_j||ID_{mn}||\ldots||ID_{mn}||f_j||hm)$ of the domain $Di$ is calculated.

3.4. Interdomain Key Negotiation. As shown in Figure 3, in the key negotiation phase of interdomain, remote participants scattered in each domain $Di(1 \leq i \leq n)$ who intend to negotiate the shared session key randomly choose user $um_i$ ($1 \leq i \leq n$) as domain head $Hi$ in their respective domains, and the key negotiation between domain heads is carried out. Assume that the domain header is divided into $Hi(um_i, 1 \leq i \leq n - 1)$ and $Hn(um_i)$, and the process of interdomain key negotiation is as follows:

(1) To prove that domain heads $Hi(1 \leq i \leq n - 1)$ and domain head $Hn$ are legitimate members of the same system, they should send messages to each other

Step 1: $Hi$ randomly chooses $ti, bi \in Z^+$ and calculates $T_j = tiP$ and then sends message $M_1 = \{ID_j, T_j, R_j\} \rightarrow um_i$.

Step 2: $um_i$ randomly chooses $tm, am \in Z^+$ and calculates $T_m = tmP$ and then sends message $M_2 = \{ID_{mn}, T_m, R_m\} \rightarrow um_i$.

Step 3: $Hi$ calculates $K_i = (H3(ID_{mn}, T_i, P_{mn})ti + sm + xmn) \cdot (H3(ID_{mn}, T_j, P_{mn})T_j + (R_j + H3(ID_{mn}, T_j, P_{mn})P_{pubj} + P_{mn})$ and $K_{bi} = bi \cdot P$ and then sends the signature message $\sigma_i = (K_{bi}, K_i)$ to $Hn$.

Step 4: $Hn$ calculates $K_{ni} = (H3(ID_{mn}, T_i, P_{mn})ti + sm + xmn) \cdot (H3(ID_{mn}, T_m, P_{mn})T_m + (R_m + H3(ID_{mn}, T_m, P_{mn})P_{pubj} + P_{mn})$. If $K_{ni} = K_i, Hi(um_i, 1 \leq i \leq n - 1)$ and $Hn(um_i)$ are legitimate members of the same system.

(2) Domain head $Hn$ calculates the interdomain session key $SK$ for interdomain negotiation based on the information received from domain head member $H_i(um_i, 1 \leq i \leq n - 1)$ that participates in the negotiation in each domain and sends the key $SK$ to other members in the negotiation in encrypted form. Legitimate members who in the system can calculate the interzone session key

Step 1: At first, $Hn$ calculates $K_{bn} = bn \cdot P$, $SK_{bn} = H2(ID_{mn}||ID_{mn}||T_i||T_m||K_{bi}||Q_n) \oplus Q_n = H4(xm||sm) + B_{ni} = H1(ID_{mn}||ID_{mn}||bn \cdot K_{bi} \oplus Q_n), Dn = xm + sm \cdot H1(ID_{mn}||ID_{mn}||k_{bn}||Q_n \cdot P), 
\hat{Z}_{an} = bn + Dn \cdot Bni, Hn = H5(K_{bn}, Q_n), Fn = H6(SK_{bn} + SK_{2n} + \ldots + SK(n - 1)n + bmn^P)$, and $KEY_i = H5(bn \cdot K_{bi}, Q_n) \oplus Fn$, then calculates the interdomain session key $SK = H7(ID_{mn}||ID_{mn}||\ldots||ID_{mn}||Fn||hn)$, and finally sends $\{ID_{mn}, ID_{mn}^2, \ldots, ID_{mn}^{n-1}, KEY_i, Hn, s_{an}\}$ to $Hi$.

Step 2: $Hi$ receives the message $\{ID_{mn}, ID_{mn}^2, \ldots, ID_{mn}^{n-1}, KEY_i, Hn, s_{an}\}$ sent by $Hn$ and then calculates $Q_n = B_{ni} \oplus H1(ID_{mn}||ID_{mn}||bi \oplus K_{bn})$ and $K_n = zn \cdot P - B_n \cdot P_{mn} + (R_n + H3(ID_{mn}, R_m, P_{mn})P_{pubj} \cdot H1(ID_{mn}||Q_n \cdot P)$. If $H5(k_n, Q_n) = Hn, Hi$ calculates $F_n = KEY_i \oplus H5(k_n \cdot bn, Q_n)$. Finally, the interdomain session key $SK = H7(ID_{mn}||ID_{mn}^2||\ldots||ID_{mn}^{n-1}||F_n||hn) = H7(ID_{mn}||ID_{mn}^2||\ldots||ID_{mn}^{n-1}||F_n||hn)$ is calculated.
3.5. Join Phase. Assume that user $um + 1$ wants to join the protocol, obtains the system parameters of KGC, generates its own public key $pk_{ji} = P_{ji} = x_{ji} \cdot P$ and private key $sk_{ji} = (s_{ji}, x_{ji})$, and performs the key negotiation again. The procedure is the same as steps (1) to (2) in intradomain key negotiation.

**Figure 2:** Key negotiation phases of intradomain.
3.6. Removal Phase

(1) If the domain head leaves, the remaining members will reselect a computationally powerful member as the domain head, and intradomain key negotiation and interdomain key negotiation are restarted. Perform steps (1) to (2) in intradomain key negotiation, and perform steps (1) to (2) in interdomain key negotiation.

(2) If it is a common intradomain member that leaves, the intradomain member initiates key negotiation.
again. The procedure is the same as steps (1) to (2) in intradomain key negotiation

4. Security Analysis

Take intradomain key negotiation as an example. The intradomain session key is \(sk_i = H^7(ID_1||ID_2||\cdots||ID_m||f(m))\), where \(f(m) = H_6(sk_1m + sk_2m + \cdots sk_m(m - 1)m + amxmP)\). Hence, we can prove the security of the protocol by proving the security of \(sk_jm(1 \leq j \leq m - 1)\).

For ease of understanding, the negotiation process of \(sk_jm\) in the protocol is assumed to be the negotiation process of \(A\) and \(B\) in domain \(Di\) to generate \(sk_i\).

Suppose \(H_1(\ast), H_2(\ast)\), and \(H_3(\ast)\) are treated as random oracles owned by \(\xi\). DDRH \((aP, bP, cP)\) is a DDRH oracle, which outputs 1 if \(abc = cP\), otherwise 0. Assume that \(\xi\) makes at most \(q_i\) queries to \(H_i(2 \leq i \leq 3)\), \(q_s\) queries to Create \((ID_i)\), \(q_j\) queries to Rt \((\Pi_{ij}^1)\), \(q_{sk}\) queries to Rsk \((\Pi_{ij}^2)\), \(q_{id}\) queries to Send \((\Pi_{ij}^3, M)\), \(q_{b}\) queries to Rs, \(q_{ski}\) queries to Rsk \((\chi^{\ast}_i)\), and \(q_{iddh}\) queries to RDDH \((aP, bP, cP)\). Assume also that bounded running time of query \(H_i(0 \leq i \leq 3)\) is \(t_i\). Create \((ID_i)\) is \(t_i\), \(Rt_i(\Pi_{ij}^1)\) is \(t_{ij}\), Rsk \((\Pi_{ij}^2)\) is \(t_{skij}\), Send \((\Pi_{ij}^3, M)\) is \(t_{ad}\), Rs is \(t_i\), Rsk \((\chi^{\ast}_i)\) is \(t_{ski}\), and RDDH is \(t_{iddh}\).

The challenger \(\xi\) maintains the query lists as follows:

- \(L_{RHD-1}:\) a tuple of \((ID_i, R_i, P_i, h_i)\)
- \(L_{RHD-2}:\) a tuple of \((ID_i, ID_j, \Pi_{ij}^m, T_j, \Pi_{ij}^m, K_{ij}, h_{2j})\)
- \(L_{RHD-3}:\) a tuple of \((ID_i, T_j, P_i, h_{TP})\)
- \(L_{C}:\) a tuple of \((ID_i, x_i, x_j, P_i, R_i)\)
- \(L_{C}^{(1)}:\) a tuple of \((\Pi_{ij}^1, t_i, T_i)\)

Given an instance of the GDH problem, for unknown \(A, B \in Z_n^\ast\), by giving \(P, AP, BP, P \in E/Fq\) and an oracle DDRH, compute \(CP = APBP\).

**Lemma 7.** Suppose \(\mathcal{A}\) wins the game in case FHl with advantage \(\varepsilon\) and running time \(t\), with the help of \(\mathcal{A}\); an algorithm \(\Gamma\) can be constructed to solve the above instance of the GDH problem with advantage \(\varepsilon^\prime\) and running time \(t^\prime\) by interacting with \(\mathcal{A}\).

\[
\varepsilon^\prime = \frac{4}{q_{id}q_{sk}(q_{id} - 1)(q_{sk} - 1)} \cdot \varepsilon,
\]

\[
t^\prime \leq 3 \sum_{i=2} q_i t_i + q_{net} t_s + q_{s} t_t + q_{sk} t_{sk} + q_{id} t_{id} + q_{id} t_{sk} + q_{sk} t_{sk} + q_{iddh} t_{iddh} + t + t_{CP},
\]

**Proof.** To interact with \(\mathcal{A}\), a GDH slover \(\Gamma\) simulates as \(\xi\) and runs the following steps to solve the above instance of the GDH problem with the help of \(\mathcal{A}\):

(C1) \(\Gamma\) executes the SETUP algorithm and sends system params to \(\mathcal{A}\).

(C2) Suppose that \(\mathcal{A}\) will choose \(\Pi_{AB}^m\) for challenge in the next step. \(\mathcal{A}\) asks the \(\Gamma\) for a polynomial number of the queries.

Create \((ID_1)\): on receiving \((ID_1)\), \(\Gamma\) performs as follows:

1. If \(L_C\) contains a tuple of \((ID_1, s_j, x_i, P_i, R_i)\), \(\Gamma\) returns all the elements of the tuple to \(\mathcal{A}\).

2. Otherwise, \(\Gamma\) randomly chooses \(x_i, s_j, h_i\) and then computes \(P_i = x_iP\) and \(R_i = s_jP - h_iP_{pub}\); \(\Gamma\) inserts \(\{ID_1, s_j, x_i, P_i, R_i\}\) to \(L_C\) and \((ID_1, R_i, P_i, h_i)\) to \(L_{RHD-1}\) and returns \((ID_1, s_j, x_i, P_i, R_i)\) to \(\mathcal{A}\).

All the following queries should be asked after Create \((ID_1)\).

- \(H_{i+1}\) query: on receiving \((ID_1, R_i, P_i)\), after query Create \((ID_1)\), there must be a tuple of \((ID_1, R_i, P_i, h_i)\) in \(L_{RHD-1}\). \(\Gamma\) returns \(h_i\) to \(\mathcal{A}\).

- \(H_{-1}\) query: on receiving \((ID_1, T_j, P_i)\), if \(L_{RHD-2}\) contains a tuple of \((ID_1, T_j, P_i, h_{TP})\), \(\Gamma\) returns \(h_{TP}\) to \(\mathcal{A}\). Otherwise, \(\Gamma\) randomly chooses \(h_{TP}\) that has not been chosen by \(\Gamma\) and inserts \((ID_1, T_j, P_i, h_{TP})\) to \(L_{RHD-2}\) and returns \(h_{TP}\) to \(\mathcal{A}\).

- \(R_s(\Pi_{ij}^1)\): on receiving \((ID_1, R_i, P_i)\), \(\Gamma\) returns \(s_j\) to \(\mathcal{A}\) from \(L_C\). \(R_s(\Pi_{ij}^1)\): on receiving \((ID_1, R_i, P_i)\), \(\Gamma\) returns \(s_j\) to \(\mathcal{A}\) from \(L_C\). Rx \((\Pi_{ij}^1)\): on receiving \((ID_1, R_i, P_i)\), \(\Gamma\) returns \(x_i\) to \(\mathcal{A}\) from \(L_C\). Rx \((\Pi_{ij}^1)\): on receiving \((ID_1, R_i, P_i)\), \(\Gamma\) returns \(x_i\) to \(\mathcal{A}\) from \(L_C\). Rx \((\Pi_{ij}^1)\): on receiving \((ID_1, R_i, P_i)\), \(\Gamma\) returns \(x_i\) to \(\mathcal{A}\) from \(L_C\). Rx \((\Pi_{ij}^1)\): on receiving \((ID_1, R_i, P_i)\), \(\Gamma\) returns \(x_i\) to \(\mathcal{A}\) from \(L_C\). Rx \((\Pi_{ij}^1)\): on receiving \((ID_1, R_i, P_i)\), \(\Gamma\) returns \(x_i\) to \(\mathcal{A}\) from \(L_C\). Rx \((\Pi_{ij}^1)\): on receiving \((ID_1, R_i, P_i)\), \(\Gamma\) returns \(x_i\) to \(\mathcal{A}\) from \(L_C\).

\[
\text{Rt}_i(\Pi_{ij}^1):
\]

1. If \(i = A, j = B, f = m\) or \(i = B, j = A, f = n\)

(a) If \(i = A, j = B, f = m\), \(\Gamma\) sets \(T_A = AP\), inserts \(\Pi_{AB}^m\) to \(L_i(\Pi_{ij}^1)\), and returns \(T_A = AP\) to \(\mathcal{A}\).

(b) If \(i = B, j = A, f = n\), \(\Gamma\) sets \(T_B = BP\), inserts \(\Pi_{BA}^m\) to \(L_i(\Pi_{ij}^1)\), and returns \(T_B = BP\) to \(\mathcal{A}\).

(2) Otherwise,

(a) If \(L_i(\Pi_{ij}^1)\) contains a tuple of \((\Pi_{ij}^1, t_i, T_i)\), \(\Gamma\) returns \(t_i, T_i\) to \(\mathcal{A}\).

(b) \(\Gamma\) randomly chooses \(t_i\), \(T_i = t_iP_i\) inserts \(\Pi_{ij}^1, t_i, T_i\) to \(L_i(\Pi_{ij}^1)\), and returns \(t_i, T_i\) to \(\mathcal{A}\).

Send \((\Pi_{ij}^1, M)\): If the matched session \(\Pi_{ij}^1\) of \(\Pi_{ij}^m\) exists, this query should be asked after Create \((ID_1)\) and \(\text{Rt}_i(\Pi_{ij}^1)\) when \(\mathcal{A}\) gets \(T_i\) and \(R_i\).

1. \(\mathcal{A}\) gets \(\{ID_1, T_i, R_i\}\) from \(L_C\) and \(L_i(\Pi_{ij}^1)\) and sets \(M = \{ID_1, T_i, R_i\}\) and \(\Gamma\) returns \(\{ID_1, T_i, R_i\}\) from \(L_C\) and \(L_i(\Pi_{ij}^1)\). That is, \(\mathcal{A}\) initiates the session
\( \Pi_{ij}^m \) with message \{ID\(_i\), T\(_i\), R\(_i\)\} and gets respond with \{ID\(_j\), T\(_j\), R\(_j\)\} of the matched session \( \Pi_{ji}^n \).

(2) If \( M = \bot, \Gamma \) gets \( T\(_i\), T\(_j\), R\(_i\), \) and \( R\(_j\) \) from \( L\(_C\) \) and \( L\(_T\) \) of \( \Pi_{ij}^m \) and initiates the session \( \Pi_{ji}^n \) with \{ID\(_j\), T\(_j\), R\(_j\)\}.

Then, \( \alpha_1 \) gets \{ID\(_j\), T\(_j\), R\(_j\)\} from \( \Gamma \) as the message of the matched session \( \Pi_{ji}^n \).

**H2 query:** on receiving \( \{ID\(_j\), ID\(_j\), (T\(_j\))\Pi_{ij}^m \}, \Gamma \) returns \( h_{2ij} \) to \( \alpha_1 \).

(1) If \( L_{RH2} \) contains a tuple of \( \{ID\(_j\), ID\(_j\), (T\(_j\))\Pi_{ij}^m \}, \Pi_{ji}^n \), \( K_{ij}, h_{2ij} \), \( \Gamma \) returns \( h_{2ij} \) to \( \alpha_1 \).

(2) Otherwise, \( \Gamma \) randomly chooses \( h_{2ij} \) that has not been chosen by \( \Gamma \) and inserts \( \{ID\(_j\), ID\(_j\), (T\(_j\))\Pi_{ij}^m \}, (T\(_j\))\Pi_{ji}^n \), \( K_{ij}, h_{2ij} \) to \( L_{RH2} \) and returns \( h_{2ij} \) to \( \alpha_1 \).

**Rsk\(_j\)(\( \Pi_{ij}^m \)) / Rsk\(_j\)(\( \Pi_{ji}^n \))**:

(1) If \( i \neq A \) and \( j \neq B \), \( \Gamma \) gets \( K_{ij} \) from \( L_{RH2} \); gets \( s_1, s_2, x_1, x_2, t_1, t_2, h_{TBA}, \) and \( h_{TPA} \) from \( L\(_C\), L\(_T\) \) of \( \Pi_{ij}^m \), and \( L_{RH2} \); computes \( K^*_i = h_{TBA}h_{TPA}t_1P + h_{TPA}t_2P + s_1s_2P + h_{TBA}x_1P + h_{TPA}x_2P \).

(2) \( \Gamma \) computes \( K^*_\(AB\) \) with the candidate solution of \( ABP \) and gets \( h^*_\(AB\) \) from \( L_{RH2} \) with \( \{ID\(_A\), ID\(_B\), (T\(_A\), T\(_B\))\Pi_{AB}^\#, \Pi_{BA}^\# \), \( K^*_\(AB\), \Pi_{ij}^m \). \)

(3) \( \Gamma \) picks randomly \( b \in \{0, 1\} \). If \( b = 1 \), \( \Gamma \) replies \( h^\#_{AB} = h^*_\(AB\) \) to \( \alpha_1 \); otherwise, \( \Gamma \) replies \( h^\#_{AB} \) as a random string to \( \alpha_1 \).

(4) \( \alpha_1 \) asks \( \Gamma \) for a polynomial number of the queries about fresh session \( \Pi_{ij}^m \).

(5) \( \alpha_1 \) makes a guess bit \( b' \).

\( \alpha_1 \) wins the game in case F11 by guessing \( b' = b \). After the test \( (\Pi_{ij}^m, b') \) query, \( \alpha_1 \) asks \( H_2 \) query with \( \{ID\(_A\), ID\(_B\), (T\(_A\), T\(_B\))\Pi_{AB}^\#, \Pi_{BA}^\#, h^\#_{AB} \}, \Gamma \) gets \( K^\#_{AB} \) in \( L_{RH2} \) and asks RDDH oracle with RDDH(\( H_3\) \{ID\(_A\), T\(_A\), P\(_A\)\}, \( T\(_A\) \) + \( R\(_A\) + \( H\(_A\) \{ID\(_B\), R\(_B\), P\(_B\)\}, \( T\(_B\) = \) \( T\(_B\) \) + \( R\(_B\) + \( H\(_B\) \{ID\(_D\), R\(_D\), P\(_D\)\}, \( P\(_D\) \) \}). If \( b = b' = 1 \) with the probability of 1/2, the above RDDH oracle will return 1; then, \( \Gamma \) can get \( K^\#_{AB} = K^*_{AB} \) and compute \( C\bullet P \) with:

\[
C\bullet P = t_1A\bullet b \cdot (h_{TBA}h_{TPA})^{-1}(K^*_{AB} - x_Bh_{TBA}T_A - x_Bh_{TPA}T_A - s_1h_{TBA}T_B = - x_Bh_{TPA}T_B - x_Ah_{TBA}P - s_Ah_{TPA}P - s_Ah_{TBA}P - x_Bh_{TPA}T_B - x_Ah_{TBA}P).
\]

(3)

Let \( T_{mul} \) be the time for one scalar multiplication operation and \( T_{add} \) be the point addition operation over elliptic curve. The time to compute \( CP \) is \( T_{CP} = 8T_{mul} + 8T_{add} \).

Then, if \( \alpha_1 \) wins the game in case F11 with advantage \( \epsilon \) and running time \( t \), then an algorithm \( \Gamma \) can be constructed to solve the GDH problem with advantage \( \epsilon' \) and running time \( t \) by interacting with \( \alpha_1 \) only if the game is completed, where events \( E_1 \) and \( E_2 \) occur.

\( E_1 \): \( \alpha_1 \) chooses \( \Pi_{AB}^\# \) for challenge.

\( E_2 \): \( b' = b = 1 \).

Meanwhile, event \( E_1 \) occurs which means all of the events \( E_{1-1} \), \( E_{1-2} \), and \( E_{1-3} \) occur.

\( E_{1-1} \): \( \alpha_1 \) chooses participants A and B for the challenge with \( Create(ID_A) \) and \( Create(ID_B) \) query.

\( E_{1-2} \): \( \alpha_1 \) makes query \( Send(\Pi_{AB}^\#, \{ID_A, T_A, R_A\}) \) or \( Send(\Pi_{BA}^\#, \{ID_B, T_B, R_B\}) \).

\( E_{1-3} \): \( \alpha_1 \) makes \( R_A(\Pi_{AB}^\#) \) query and \( R_B(\Pi_{BA}^\#) \) query.

\( \epsilon' = Pr[E_{1-1}]\cdot Pr[E_{1-2}]\cdot Pr[E_{1-3}]\cdot Pr[E_2] \cdot \epsilon = \frac{C_2^1}{C_q^1}\frac{1}{C_{q_d}^1}\frac{1}{C_{q_t}^1} \epsilon = \frac{4}{q_d q_t (q_d - 1)(q_t - 1)} \epsilon \).

(4)

\( \tau \leq \sum_{i=0}^{1} q_1 t_1 + q_2 t_2 + q_3 t_3 + q_4 t_4 + q_5 t_5 + q_6 t_6 + q_7 t_7 + q_8 t_8 + \frac{q_p t_{piki} + q_{rddh} t_{rddh} + t^t_{CP}}{\epsilon} \).

\( \square \)

**Lemma 8.** The same as Lemma 7 but in case F12.

**Proof.** To interact with \( \alpha_1 \), a GDH solver \( \Gamma \) runs the same steps as that in Lemma 7 to solve the instance of the GDH problem. \( \alpha_1 \) asks \( \Gamma \) for a polynomial number of the queries as shown in Lemma 7; \( \Gamma \) answers the following queries differently:

Create(\( ID_A \)): on receiving \( \{ID_j\} \), \( \Gamma \) performs as follows:

(1) If \( L_C \) contains a tuple of \( \{ID_i, s_i, x_i, P_i, R_i\} \)

(a) If \( i \neq A, B \), \( \Gamma \) returns all the elements of the tuple to \( \alpha_1 \).

(b) Otherwise, \( \Gamma \) returns \( \{ID_i, \bot, x_i, P_i, R_i\} \) to \( \alpha_1 \).

Wireless Communications and Mobile Computing
Lemma 9. The same as Lemma 7 but in case FI3.

Proof. To interact with $\mathcal{A}_1$, a GDH solver $\Gamma$ runs the same steps as that in Lemma 7 to solve the instance of the GDH problem. $\mathcal{A}_1$ asks the $\Gamma$ for a polynomial number of the queries as shown in Lemma 7; $\Gamma$ answers the following queries differently:

Create($ID_i$): on receiving ($ID_i$), $\Gamma$ performs as follows:

1. If $L_C$ contains a tuple of ($ID_i, s_i, x_i, P_i, R_i$)

(a) If $i \neq A, B$, $\Gamma$ randomly chooses $x_i, s_i, h_i$ and then computes $P_i = s_iP$ and $R_i = s_iP - h_iP_{pub}$; $\Gamma$ inserts ($ID_i, s_i, x_i, P_i, R_i$) to $L_C$ and ($ID_i, R_i, P_i, h_i$) to $L_{RHI-1}$ and returns ($ID_i, s_i, x_i, P_i, R_i$) to $\mathcal{A}_1$

(b) $\Gamma$ randomly chooses $x_i, h_i$ and computes $P_i = s_iP$ and $R_i = IP - h_iP_{pub}$ ($I = A$ when $i = A; I = B$ when $i = B$); $\Gamma$ inserts ($ID_i, s_i, x_i, P_i, R_i$) to $L_C$ and ($ID_i, R_i, P_i, h_i$) to $L_{RHI-1}$ and returns ($ID_i, s_i, x_i, P_i, R_i$) to $\mathcal{A}_1$

$R_5$($ID_i$): on receiving $ID_i$, $\Gamma$ performs as follows:

1. If $i \neq B$, $\Gamma$ returns $s_i$ from $L_C$ to $\mathcal{A}_1$

2. Otherwise, $\Gamma$ returns $\bot$ to $\mathcal{A}_1$

$R_4$($\Pi^j_i$):

1. If $L_j$($\Pi^j_i$) contains a tuple of ($\Pi^j_{i,j}, t_i, T_i$), $\Gamma$ returns $t_i, T_i$ to $\mathcal{A}_1$

2. Otherwise, $\Gamma$ randomly chooses $t_i$, computes $T_i = t_iP$, inserts ($\Pi^j_{i,j}, t_i, T_i$) to $L_j$ ($\Pi^j_i$), and returns $t_i, T_i$ to $\mathcal{A}_1$

Moreover, the first step of (C3) is also different with that of Lemma 7.

(C3) $A$ asks a test ($\Pi^w_{A,B}$) query; the challenger $\zeta$ performs as follows:

1. $\zeta$ gets $x_A, x_B, t_A, t_B, R_A, R_B, P_A, P_B, h_A, h_B, T_A, T_B, h_{TPA}$, and $h_{TPB}$ from $L_C$, $L_{RHI-1, L_{RHI-2}}$, and $L_i$ ($\Pi^j_i$) where $R_A + h_AP_{pub} = AP, R_B + h_BP_{pub} = BP$.

$\Gamma$ gets $C\cdot P$ with

$$C\cdot P = s_A^*x_A^*P = (K_{A}^* - h_{TPA}t_A^*A^*P - h_{TPA}t_A^*A^*B^*P - h_{TPB}t_B^*A^*P - h_{TPB}t_B^*B^*P - x_A^*h_{TPA}T_A - x_B^*h_{TPB}T_B - x_A^*h_{TPA}P - h_{TPB}T_B - x_A^*h_{TPB}P - x_A^*h_{TPA}P).$$

(5)
Table 4: Calculation time of password operation.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{\text{mul}}$</td>
<td>A dot product operation time on an elliptic curve</td>
<td>7.3529 ms</td>
</tr>
<tr>
<td>$T_b$</td>
<td>Hash function operation time</td>
<td>0.0004 ms</td>
</tr>
<tr>
<td>$T_m$</td>
<td>Modular multiplication operation time</td>
<td>0.0147 ms</td>
</tr>
<tr>
<td>$T_{\text{exp}}$</td>
<td>Modular exponential operation time</td>
<td>18.38225 ms</td>
</tr>
<tr>
<td>$T_{\text{add}}$</td>
<td>One point plus operation time on an elliptic curve</td>
<td>0.0613 ms</td>
</tr>
</tbody>
</table>

Γ gets $C \cdot P$ with

$$C \cdot P = s_A t_B \cdot P = (h_{TPA})^{-1} (K_{AB}^* - h_{TPA} h_{TPA} t_A T_B - h_{TPA} t_A \cdot (R_B + h_B P_{pub}) - s_B (R_A + h_A P_{pub}) - s_B (R_A + t_A h_{TPA} T_B - s_A (R_B + h_B P_{pub}) - s_A x_B P)).$$

(7)

Lemma 11. The same as Lemma 7 but in case FI5 and with different $\epsilon'$. $\epsilon' = (1/a_q q_{sd} q_l (q_e - 1)) \cdot \epsilon$

Proof. To interact with $\mathcal{A}_1$, a GDH solver Γ runs the same steps as that in Lemma 7 to solve the instance of the GDH problem. Γ asks the $\Gamma$ for a polynomial number of the queries as shown in Lemma 9; Γ answers the following queries differently:

**Rt_{\Gamma}(I_{T_{\Gamma}}):**

(1) If $i = A$, $j = B$, $f = m$ or $j = A$, $i = B$, and $f = n$

(a) If $i = A$, $j = B$, and $f = m$, Γ sets $T_A = AP$, inserts $(\Pi_{AB}^{m,A,B})$ to $L_i(I_{T_{\Gamma}})$, and returns $T_A = AP$ to $\mathcal{A}_1$

(b) Otherwise, Γ returns ⊥ to $\mathcal{A}_1$

(2) Otherwise,

(a) if $L_i(I_{T_{\Gamma}})$ contains a tuple of $(\Pi_{T_{\Gamma}} t_i, T_{\Gamma})$, Γ returns $t_i, T_{\Gamma}$ to $\mathcal{A}_1$

(b) Γ randomly chooses $t_i$, computes $T_i = t_i P$, inserts $(\Pi_{T_{\Gamma}} t_i, T_{\Gamma})$ to $L_i(I_{T_{\Gamma}})$, and returns $t_i, T_{\Gamma}$ to $\mathcal{A}_1$

Send$(\Pi_{T_{\Gamma}}, M)$:

The matched session $\Pi_{T_{\Gamma}}$ of $\Pi_{T_{\Gamma}}^{m}$ does not exist.

(1) This query should be asked after Create$(ID_A)$ and $R_t(I_{T_{\Gamma}})$ when $\mathcal{A}_1$ gets $T_i$ and $R_i$. $\mathcal{A}_1$ gets $\{ID_A, T_i, R_i\}$ from $L_C$ and $L_i(I_{T_{\Gamma}})$ and sets $M = \{ID_A, T_i, R_i\}$; Γ returns $\{ID_{A}', \bot, R_i\}$ to $\mathcal{A}_1$ as response

(2) If $M = \bot$, Γ initiates the session $\Pi_{T_{\Gamma}}^{m}$ with $\{ID_A, T_i, R_i\}$

(a) If $i = A$, $j = B$, and $f = m$, this query should be asked after Create$(ID_B)$. $\mathcal{A}_1$ gets $x_B, P_B$ from $L_C$, sets $t'_B = x_B$, and sets $T_B = P_B$, which responds with $\{ID_{B}', T'_B, R_B\}$

(b) Otherwise, $\mathcal{A}_1$ responds with $\{ID_{A}', \bot, R_i\}$

Moreover, the first two steps of (C3) is also different with that of Lemma 9.

(C3) $\mathcal{A}_1$ asks a test $(\Pi_{T_{\Gamma}}^{m,A,B})$ query; the challenger ζ performs as follows:

(1) ζ gets $s_A, x_A, x_B, R_A, R_B, P_A, P_B, T_A, h_A, h_B, h_{TPA}$, and $h_{TPB}$ from $L_C$, $L_{RH1-3}$, $L_{RH2-3}$, and $L_i(\Pi_{T_{\Gamma}})$ where $T_A = AP, R_B + h_B P_{pub} = BP$. ζ gets $T'_B$ from the response message from $\mathcal{A}_1$

(2) Let $T'_B = B' P$; ζ computes $K_{AB}^*$ with the candidate solution of $ABP$ and $AB' P$ and gets $h_{2AB}$ from $L_{RH2}$ with $(ID_A, ID_B, (T_A, m)\Pi_{T_{\Gamma}}^{m}(B)\Pi_{T_{\Gamma}}^{m,A}(K_{AB}))$

As the proof in Lemma 7, if $\mathcal{A}_1$ win the game in case FI5, an algorithm Γ can be constructed to solve the GDH problem only if the game is completed, where events $E_1$ and $E_2$ occur. In case FI5, event $E_1$ occurs which means all of the events $E_1-1, E_1-4$, and $E_1-5$ occur.

$E_{1-1}: \mathcal{A}_1$ chooses participant A and B for challenge with Create$(ID_A)$ and Create$(ID_B)$ query

$E_{1-4}: \mathcal{A}_1$ makes query Send$(\Pi_{T_{\Gamma}}^{m,A,B})$

$E_{1-5}: \mathcal{A}_1$ makes query Rtn$_A(\Pi_{T_{\Gamma}}^{m,A,B})$

$$e' = Pr[E_{1-1}] \cdot Pr[E_{1-4}] \cdot Pr[E_{1-5}] \cdot Pr[E_2] \cdot e = \frac{1}{C_{q_e}^2} \cdot \frac{1}{C_{q_{sd}}^2} \cdot \frac{1}{C_{q_l}^2} \cdot \frac{1}{q_e, q_{sd} q_l (q_e - 1)} \cdot e.$$
<table>
<thead>
<tr>
<th>Protocols</th>
<th>Number of rounds</th>
<th>Low-power mobile node</th>
<th>Powerful node</th>
<th>Join phase</th>
<th>Removal phase</th>
<th>Total computation cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chuang and Tseng [28]</td>
<td>3</td>
<td>$2T \exp + 3T h$</td>
<td>$(m + 1)T \exp + (2m + 1)T h$</td>
<td>$2T \exp + (2m + 3)T h$</td>
<td>$(2m + 2)T h$</td>
<td>$(m + 5)T \exp + (6m + 9)T h$</td>
</tr>
<tr>
<td>Wu et al. [29]</td>
<td>2</td>
<td>$2T \exp + 2T h$</td>
<td>$(m + 1)T \exp + (m + 1)T h$</td>
<td>$2T \exp + (m + 4)T h$</td>
<td>$(m + 4)T h$</td>
<td>$(m + 5)T \exp + (3m + 11)T h$</td>
</tr>
<tr>
<td>Islam et al. [30]</td>
<td>2</td>
<td>$4T \text{mul} + 4T h$</td>
<td>$(5m + 4)T \text{mul} + (2m + 4)T h$</td>
<td>$(2m + 5)T \text{mul} + 6T h$</td>
<td>$(2m + 4)T \text{mul} + 5T h$</td>
<td>$(7m + 15)T \text{mul} + (2m + 17)T h$</td>
</tr>
<tr>
<td>Mandal et al. [31]</td>
<td>2</td>
<td>$5T \text{mul} + 9T h$</td>
<td>$(5m + 2)T \text{mul} + (5m + 7)T h$</td>
<td>$(3m + 4)T \text{mul} + (2m + 11)T h$</td>
<td>$(3m + 4)T \text{mul} + (2m + 11)T h$</td>
<td>$(10m + 15)T \text{mul} + (9m + 38)T h$</td>
</tr>
<tr>
<td>Luo et al. [21]</td>
<td>1</td>
<td>$5T \text{mul} + T \text{add} + 4T h$</td>
<td>$(5m + 1)T \text{mul} + 3mT \text{add} + 4mT h$</td>
<td>$(3m + 10)T \text{mul} + (m + 5)T \text{add} + (m + 8)T h$</td>
<td>$(3m - 1)T \text{mul} + (m + 2)T \text{add} + mT h$</td>
<td>$(11m + 15)T \text{mul} + (5m + 8)T \text{add} + (6m + 12)T h$</td>
</tr>
<tr>
<td>Proposed scheme</td>
<td>2</td>
<td>$5T \text{mul} + 3T h$</td>
<td>$4(m - 1)T \text{mul} + 6(m - 1)T h$</td>
<td>$(4m + 5)T \text{mul} + (6m + 3)T h$</td>
<td>$(4m - 3)T \text{mul} + (6m - 9)T h$</td>
<td>$(12m + 3)T \text{mul} + (18m - 9)T h$</td>
</tr>
</tbody>
</table>
computes $C \cdot P$ with

$$C \cdot P = t_s \cdot P = (h_{TPA})^{-1}(K_{AB}^* - h_{TPA}K_{HA}x_s A^*T_A - x_s h_{TPA}T_A)$$

$$- h_{TPA}x_s A^* - x_s h_{TPA}T_B - x_s (R_B + h_B P_{pub}) - x_s x_h P$$

$$= (h_{TPA})^{-1}(K_{AB}^* - h_{TPA}K_{HA}x_s A^*T_A - x_s h_{TPA}T_A - h_{TPA}x_s A^* - h_{TPA}x_s A^* - h_{TPA}x_s A^*)$$

$$- x_s h_{TPA}T_B - x_s (R_B + h_B P_{pub}) - x_s x_h P.$$  (9)

\[\square\]

**Lemma 12.** The same as Lemma 11 but in case F15.

**Proof.** To interact with $A$, a GDH solver $\Gamma$ runs the same steps as that in Lemma 7 to solve the instance of the GDH problem. $A$ asks the $\Gamma$ for a polynomial number of the queries as shown in Lemma 8; $\Gamma$ answers the following queries differently:

- (1) If $j = A$, $i = B$, and $f = n$, $\Gamma$ returns $\bot$ to $A$.
- (2) Otherwise,
  - (a) if $L_i(\Pi_{I,i}^f)$ contains a tuple of $(\Pi_{I,i}^f, t_i, t_i')$, $\Gamma$ returns $t_i, t_i'$ to $A$.
  - (b) $\Gamma$ randomly chooses $t_i'$, computes $t_i = t_i' P$, inserts $(\Pi_{I,i}^f, t_i, t_i')$ to $L_i(\Pi_{I,i}^f)$, and returns $t_i, t_i'$ to $A$.

Send$(\Pi_{I,i}^f, M)$: the same as that of Lemma 11. Moreover, the first two steps of (C3) is also different with that of Lemma 8.

(C3) $A$ asks a test $(\Pi_{A,B}^m)$ query; the challenger $C$ performs as follows:

- (1) $C$ gets $x_A, x_B, t_A, R_A, R_B, P_A, P_B, h_A, h_B, T_A, h_{TPA}$, and $h_{TPB}$ from $L_C$, $L_{R1}^H$, $L_{R2}^H$, and $L_i(\Pi_{I,i}^f)$ where $R_A + h_A P_{pub} = AP, R_B + h_B P_{pub} = BP$. $C$ gets $T_A'$ from the response message from $A$.
- (2) Let $T_B' = B' P$; $C$ computes $K_{AB}^*$ with the candidate solution of $ADB$ and $AB'$ and gets $h_{2AB}^*$ from $L_{R1}^H$ with $(ID_A, ID_B, T_A)\Pi_{A,B}^m, (T_B)\Pi_{B,A}^m, K_{AB}^*$.

$\Gamma$ gets $C \cdot P$ with

$$C \cdot P = t_s \cdot P = (K_{AB}^* - h_{TPA}K_{HA}x_s A^*T_A - x_s h_{TPA}T_A)$$

$$- h_{TPA}x_s A^* - x_s h_{TPA}T_B - x_s (R_B + h_B P_{pub}) - x_s x_h P.$$

$$= (K_{AB}^* - h_{TPA}K_{HA}x_s A^*T_A - x_s h_{TPA}T_A - h_{TPA}x_s A^* - h_{TPA}x_s A^*)$$

$$- x_s h_{TPA}T_B - x_s (R_B + h_B P_{pub}) - x_s x_h P.$$  (10)

With the similar proof, Lemma 13 to 18 can be derived for case F11 to F16, which need not be given in this paper. Then, according to Lemma 7 to 18, if $A$ wins the game in polynomial time, $\Gamma$ can solve the GDH problem, which is contradictory with the security assumption of GDH problem. Then, we conclude that $A$ cannot win the Game and $Adv_A(k)$ is negligible. Therefore, our protocol is secure under the eCK security model with the GDH assumption. $\Box$

### 5. Performance Analysis

According to the research work in literatures [26, 27], the calculation time of relevant operation in protocol execution is shown in Table 4. The cross-domain authentication scheme proposed in this paper is compared and analyzed with existing schemes of the same type, and the cost comparison of scheme calculation is shown in Table 5. The parameters used in the table are as follows:

- $m$: number of participants in each domain $Di$
- $n$: the number of domains for cross-domain key negotiation
- $N$: the number of participants in a negotiation

In this scheme, we use the dot product of elliptic curve operation and hash function to encryption scheme. Chuang and Tseng and Wu et al.'s scheme [28, 29] used modular exponential operation. Table 4 shows that the operation time of modular finger is longer than a dot product operation time on elliptic curve. Compared with that, our calculation cost is smaller and the number of rounds negotiated is also less. Though the computation cost of [30] is lower than our proposed model, however Tan's scheme lacks perfect forward secrecy [32], and the schemes do not have a signature mechanism that supports confidentiality, integrity, authentication, and nonrepudiation in a logical step. Mandal et al.'s scheme [31] used signature verification with a signature verification time of 2.005 s, but they did not include the signature verification time into the calculation cost. So compared with Mandal et al.'s scheme, our computational cost is still lower. Although in Luo et al.'s scheme [21], the group session key is calculated in one-round communication and the total computation cost is lower than our proposed protocol when $m$ is large, our proposed protocol has lower computation cost for low-power mobile node and powerful node when there is no member joining or removing from the group. In addition, Luo et al.'s scheme only had a secondary security level, the malicious KGC could collude with some malicious users to attack the protocol indicated by Ren et al. [24].

### 6. The Conclusion

In view of the shortage and complexity of cross-domain group authentication communication schemes, this paper proposes a certificateless cross-domain group key management scheme based on ECC. In the proposed scheme, key negotiation is divided into two parts: intradomain key negotiation and interdomain key negotiation. On the basis of ensuring security, group cross-domain communication is realized. It avoids the complex certification path construction and verification process and reduces the length of the
trust path. The scheme is proved to be secure in random oracle model with low computational cost and is suitable for users’ group communication requirements across multiple domains.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no competing interests.

Acknowledgments

This work was supported by the National Key R&D Program of China on Blue Granary Technology Innovation (2019YFDF000805), the Program for Shanghai Collaborative Innovation Center for Cultivating Elite Breeds and Green-Culture of Aquaculture Animals (Project No. 2021-KJ-02-12), and the Science and Technology on Near-Surface Detection Laboratory (Project No. 6142414200410). The authors would also like to acknowledge the support of Shang Hai Ocean University throughout the research process.

References


