Information Sciences 543 (2021) 143–156

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Certificateless two-party authenticated key agreement scheme for smart grid

Lunzhi Deng *, Ronghai Gao

School of Mathematical Sciences, Guizhou Normal University, Guiyang 550001, China

ARTICLE INFO

Article history:
Received 18 July 2019
Received in revised form 9 July 2020
Accepted 10 July 2020
Available online 23 July 2020

Keywords:
Certificateless cryptography
Authenticated key agreement
Smart grid
Standard model
ECK model

ABSTRACT

Smart grid is a fully automated power transmission network. It monitors and controls each user and grid node to ensure bidirectional flow of information and power between all nodes. It is an important issue that how to achieve secure sharing of information among numerous communicating agents in the smart grid environments. Authenticated key agreement (AKA) is a good option to enable secure communication between the smart meter and utility. In recent years, several AKA schemes have been put forward for smart grid environments. There are two shortcomings in these schemes: First, they are constructed from traditional public key infrastructure (PKI) or identity based cryptography (IBC), so they suffer from certificate management problem or key escrow problem. Second, the security proofs of these schemes were done in the random oracle model (ROM). It is well known, a cryptographic scheme proven to be secure in ROM is not necessarily safe in practical applications. In this paper, we present a certificateless two-party authenticated key agreement (CL2PAKA) scheme for smart grids, then provide the security proofs in the standard model. Our scheme does not require pairing operations and requires only four scale multiplication operations, so it is more efficient than previous ones.

1. Introduction

Smart grid is built on an integrated, high-speed two-way communication network. It achieves the goals of reliability, safety, economy and efficiency through the application of sensing and measuring technology, equipment technology, control methods and decision support system technology. Its main features include self-healing, defending against attacks, providing quality power to meet user needs, allowing access to a variety of different forms of power generation, starting the electricity market, and optimizing the efficient operation of assets.

Smart grid includes four functional systems: sensing, communication, control and drive. The service system consists of communication, control and driver modules. Smart meters (SMs) consists of sensing and communication modules. They are responsible for monitoring energy consumption and providing power price information to consumers. It is not difficult for a malicious attacker to wiretap, amend, and fracture messages delivered between the SMs and the service providers(SPs) because of the wireless property of SMs. In addition, SMs mounted outside the house are ordinarily safeguarded only by entity locks. Therefore, by breaking the physical lock and destroying the physical structure of the SMs, an attacker can easily catch the important information deposited in the SMs. With the prompt growth of smart grid skillfulness, it becomes an aris-

* Corresponding author.
E-mail address: denglunzhi@163.com (L. Deng).

https://doi.org/10.1016/j.ins.2020.07.025
0020-0255/

ing research theme to devise and develop secure and efficient cryptography schemes (such as encryption, authentication and key management).

In smart grids, the data transmitted between the SMs and the SPs may be sensitive and important. If the data is stolen or tampered, it may reveal some personal privacy of the user and even cause significant economic losses to users and service providers. Therefore, these data need to be encrypted before being transmitted. In symmetric encryption, the shared key needs to be set in advance. In public key encryption, there is no need to set a shared key in advance. However, public key encryption consumes more computational cost than symmetric encryption. Therefore, it is an effective method to first generate a shared key through a public key system and then use symmetric encryption to encrypt the data. For smart grids, it is an important issue that how to safely and efficiently generate shared keys between SMs and SPs.

1.1. Related work

Two-parts authenticated key agreement (2PAKA) is an essential cryptographic primitive. A shared session key is generated by the communicating two parties through the open network, and no one other than the communicating parties can know the session key. In 1984, Shamir [34] introduced identity-based cryptography (IBC), which avoids certificate management in the traditional public key infrastructure (PKI). Subsequently, some identity-based 2PAKA schemes [14,15,28–30,42,48] were constructed. All of the above schemes require pairing operation, which is a relatively expensive operation. To reduce the computation cost, researchers designed several identity-based 2PAKA schemes without pairing operations [4,9,11,31,37,45]. In IBC, since the private key generator (PKG) holds the private keys of all users, the user's information will be imperiled if there are potentially dangerous elements inside PKG.

In 2003, Al-Riyami and Paterson presented certificateless cryptography (CLC) [1], which resolves the key escrow problem while avoiding certificate management. Afterwards, many certificateless two-parts authenticated key agreement (CL2PAKA) schemes [13,21,24–26,41,50] were proposed based on bilinear pairings. The computation cost of a pairing operation is about 3 times than that of a scalar multiplication operation on elliptic curve. It is very appealing to design CL2PAKA schemes without pairing operation. In 2009, Geng and Zhang [12] presented the first CL2PAKA scheme without bilinear pairing, and claimed that it is provably secure in the extended Canetti-Krawczyk (eCK) model. In the same year, Hou and Xu [20] also presented a CL2PAKA scheme without bilinear pairing, they did not give the security proofs. However, Yang and Tan [47] pointed out that the two schemes [12,20] are insecure, then constructed a new scheme and showed the security proofs in the eCK model. In 2011, Xiong et al. [44] designed a pairing-free CL2PAKA scheme. In 2012, He et al. [17] and He et al. [18] respective proposed a pairing-free CL2PAKA scheme, and separately declared that the scheme is provably secure in the eCK model. However, Cheng [10] indicated that the scheme [18] is vulnerable against a type I adversary who gets the ephemeral private key of any one party. He et al. [16] showed that the scheme [17] is insecure against a type I adversary, and presented a new scheme. In 2013, Sun et al.[35] pointed out that the two schemes [16,44] are vulnerable, in which the session key can be computed by an adversary who obtained the ephemeral secret keys of communication two parties. They presented a new pairing-free CL2PAKA scheme, and demonstrated the security proofs based on gap Diffie-Hellman (GDH) problem. In the same year, Sun et al. [36] constructed another CL2PAKA scheme without bilinear pairing, and proved it to be secure in the eCK model. Kim et al. [22] presented a pairing-free CL2PAKA scheme and asserted that it is provably secure in the eCK model. However, Bala et al. [6] pointed out that the scheme [22] is vulnerable against Key-Compromise Impersonation attacks. In 2015, Tu et al. [39] put forward a pairing-free CL2PAKA scheme, and showed the security proofs in the eCK model. In 2016, Sun et al. [38] proposed a pairing-free CL2PAKA scheme, which is provably secure based on computation Diffie-Hellman (CDH) problem. Bala et al. [5] presented a pairing-free CL2PAKA scheme for wireless sensor networks, and gave the security proofs in the eCK model.

In 2011, based on elliptic curve and Needham-Schroeder authentication protocol, Wu and Zhou [43] put forward a key distribution scheme for smart grid, and claimed that it is secure. However, Xia and Wang [46] indicated that the scheme [43] is vulnerable against the man-in-the-middle attacks, then constructed a new key distribution scheme. Park et al. [33] pointed out that the scheme [46] is insecure against the impersonation attacks. In 2016, Tsai and Lo [40] designed an anonymous key distribution scheme for smart grid, and asserted that it is provably secure in ROM. However, Odelu et al. [32] indicated that the scheme [40] is vulnerable against the session-state reveal attacks, and presented a new 2PAKA scheme. Yan et al. [49] proposed an authentication and key agreement scheme for smart grid, but did not give the security proofs. In 2018, Mahmooda et al. [27] constructed a 2PAKA scheme for smart grid based on pairings, which supports smart meter anonymity.

1.2. Progress of the security model

In 1993, Bellare and Rogaway [2] proposed the first formal security model (BR model) for authenticated key agreement (AKA) scheme, that can withstand known-key attacks and impersonation attacks. However, it does not study the harm caused by the leakage of long-term private keys, nor does it apply to asymmetric AKA schemes. To make up for the defect, Blake-Wilson et al. [3] introduced the modified BR model (mBR model). However, there are two cases that have not been considered: the leakage of ephemeral private keys and the leakage of intermediate results. In 2001, Canetti and Krawczyk [7] put forward a new security model (CK model). However, it is not weak perfect forward secrecy and does not consider the compromise impersonation attacks. In 2007, LaMacchia et al. [23] proposed the extended CK model (eCK model), which captures all of the security properties.

1.3. Motivation and contributions

Due to smart meters are low-energy devices, CL2PAKA schemes using bilinear mapping are not applicable to smart grids. It was in ROM that the security proofs of known CL2PAKA schemes without requiring pairing operation are given. ROM is a simulation for the hash function, and can not replace the real hash function computation. Canetti et al. [8] stated that a cryptographic scheme is not necessarily safe in real life even if it has been proved to be secure in ROM.

It is attractive to design an efficient CL2PAKA scheme for smart grids and provide the security proofs in the standard model. The contribution of this paper can be summarized as follows:

- We propose a new CL2PAKA scheme. In order to establish a shared key, each of the communication parties only needs to send the message stream once.
- We provide the security proofs in the standard model and eCK model. In the proofs, the adversary can directly calculate the value of the hash function instead of querying the challenger. To the best of our knowledge, it is the first CL2PAKA scheme with provably security in the standard model.
- We make the efficiency comparison between the new scheme and several other schemes. The new scheme does not require pairing operation and requires only four scale multiplication operations for each communication party. So it is more efficient than previous schemes and is suitable for smart grids.

1.4. Roadmap

First, preliminaries are introduced in Section 2. Second, the construction and security proofs for a new CL2PAKA scheme are described in Section 3 and Section 4, respectively. Third, the performance comparisons on several schemes are presented in Section 5. Lastly, some conclusions are proposed in Section 6.

2. Preliminaries

In this section, some preliminary knowledge are introduced, including elliptic curve group, computation Diffie-Hellman problem and security requirements for a CL2PAKA scheme. The notations used throughout the paper are listed in Table 1.

2.1. Elliptic curve group

Let $\mathbb{E}/F_p$ denote an elliptic curve $\mathbb{E}$ over a prime finite field F_p, defined by an equation:

$$y^2 = x^3 + ax + d \pmod{p},\ a, d \in F_p \text{ and } 4a^3 + 27d^2 \neq 0 \pmod{p}.$$

The points on $\mathbb{E}/F_p$ together with an extra point O called the point at infinity form a group:

$$\mathcal{G} = \{(x, y) : x, y \in F_p, \mathbb{E}(x, y) = 0\} \cup \{O\}.$$

Definition 1 (*Computation Diffie-Hellman (CDH) problem*). Let $G = (P) \leqslant \mathcal{G}$ and P is a point with prime order q. Given the tuple (P, uP, vP), compute uvP.

Table 1
Notations.

Symbol	Meaning
p, q	Two prime numbers
F_p	A prime finite field
Z_q^*	A set consisting of positive integers less than q.
G, P	A addition group with q order and a generator.
x, P_{pub}	The private and public key of system, where $P_{pub} = xP$.
$H_1 \sim H_3$	Three secure hash functions.
ID_i	The identity of the i^{th} smart meter $\mathcal{SM}_i$.
t_i, T_i	The secret value and public key of $\mathcal{SM}_i$.
(d_i, R_i)	The partial private key of $\mathcal{SM}_i$.
ID_j	The identity of the j^{th} service provider $\mathcal{SP}_j$.
t_j, T_j	The secret value and public key of $\mathcal{SP}_j$.
(d_j, R_j)	The partial private key of $\mathcal{SP}_j$.

2.2. Security requirements

There are two types of adversaries against a CL2PAKA scheme. The Type I adversary $\mathcal{A}_1$ plays a dishonest user, who can know the user's secret value and even replace the user's public key, however knows nothing on the user's partial private key. The Type II adversary $\mathcal{A}_2$ plays a malicious KGC that can know the master secret key of the system and the user's private key, however cannot know the user's secret value and cannot replace the user's public key.

Let $\prod_{i,j}^{s}$ denotes the s^{th} session between the smart meter $\mathcal{SM}_i$ and the service provider $\mathcal{SP}_j$. The session $\prod_{i,j}^{s}$ is completed if and only if a session key is computed.

The adversary $\mathcal{A} \in \{\mathcal{A}_1, \mathcal{A}_2\}$ controls entire communication link, so can wiretap, pause, intercept, amend, and infuse messages randomly. $\mathcal{A}$ can interact with many oracles, oracle represents an instance of the scheme being actually executed, and it is generally denoted by $\prod_{i,j}^{s}$. The ability of $\mathcal{A}$ is simulated through following queries.

- Query(PK): $\mathcal{A}$ obtains the public key of a user.
- Query(MK): $\mathcal{A}$ obtains the master secret key.
- Replace(PK): $\mathcal{A}$ replaces the public key of a user.
- Query(SV): $\mathcal{A}$ obtains the secret value of a user whose public key was not be replaced.
- Query(PPK): $\mathcal{A}$ obtains the partial private key of a user.
- Query(ESK): $\mathcal{A}$ obtains the ephemeral secret key of a user.
- Query(SK): $\mathcal{A}$ obtains the session key of a completed session.
- Send $(\prod_{i,j}^{s}, m)$: $\mathcal{A}$ obtains a response generated through the session when he sends a message m to $\prod_{i,j}^{s}$ on behalf of ID_j.
- Test $(\prod_{i,j}^{s})$: $\mathcal{A}$ obtains a session key or a random value when he submits a fresh session.

Definition 2 (*Matching session*). Two sessions $\prod_{j,i}^{e}$ and $\prod_{i,j}^{s}$ are said to be matched if the messages generated in both sessions are the same except for the sequence.

Definition 3 (*Fresh session*). For different adversaries, the definition of freshness is divided into two types. Let $\prod_{i,j}^{s}$ be a completed session between an honest user ID_i and another honest user ID_j.

- Case 1 (Freshness against $\mathcal{A}_1$). $\prod_{i,j}^{s}$ is said to be fresh if none of the three cases occur.
  1. $\mathcal{A}_1$ obtains the session key of $\prod_{i,j}^{s}$ or the session key of the matching session $\prod_{j,i}^{e}$ (if $\prod_{j,i}^{e}$ exists).
  2. The matching session $\prod_{j,i}^{e}$ does exist. $\mathcal{A}_1$ obtains the partial private key and ephemeral secret key of the user ID_i in $\prod_{i,j}^{s}$ or the partial private key and ephemeral secret key of the user ID_j in $\prod_{j,i}^{e}$
  3. The matching session $\prod_{j,i}^{e}$ does not exist. $\mathcal{A}_1$ obtains the partial private key and ephemeral secret key of the user ID_i in $\prod_{i,j}^{s}$ or the partial private key of the user ID_j.
- Case 2 (Freshness against $\mathcal{A}_2$). $\prod_{i,j}^{s}$ is said to be fresh if none of the three cases occur.
  1. $\mathcal{A}_2$ obtains the session key of $\prod_{i,j}^{s}$ or the session key of the matching session $\prod_{j,i}^{e}$ (if $\prod_{j,i}^{e}$ exists).
  2. The matching session $\prod_{j,i}^{e}$ does exist. $\mathcal{A}_2$ obtains the secret value and ephemeral secret key of the user ID_i in $\prod_{i,j}^{s}$ or the secret value and ephemeral secret key of the user ID_j in $\prod_{j,i}^{e}$
  3. The matching session $\prod_{j,i}^{e}$ does not exist. $\mathcal{A}_2$ obtains the secret value and ephemeral secret key of the user ID_i in $\prod_{i,j}^{s}$ or the secret value of the user ID_j.

In the game with parameter v, the adversary $\mathcal{A} \in \{\mathcal{A}_1, \mathcal{A}_2\}$ first makes a series of queries, then makes a Test query for a fresh session $\prod_{i,j}^{s}$. Next, a fair coin $\mu \in \{0, 1\}$ is flipped by the challenger. $\mathcal{A}$ gets the session key if $\mu = 0$, or gets a random value from the distribution of session key if $\mu = 1$. Finally, $\mathcal{A}$ outputs a guess μ' and wins in the game if $\mu' = \mu$. The advantage of $\mathcal{A}$ is defined as: $Adv_{\mathcal{A}}(v) = |\Pr[\mu' = \mu] - \frac{1}{2}|$.

Definition 4 (*Secure scheme*). A CL2PAKA scheme is said to be secure in the eCK model if the following two conditions are satisfied.

- The session keys generated in the two matching sessions are always the same.
- $Adv_{\mathcal{A}}(v)$ is negligible for any $\mathcal{A} \in \{\mathcal{A}_1, \mathcal{A}_2\}$.

3. New scheme

In this section, a new CL2PAPA scheme is constructed for smart grid environment (Fig.1), which is provably secure in the standard model. In the scheme (Fig.2), a key generation center (KGC) acts as the trusted authority.

3.1. Setup phase

With a security parameter v, KGC generates system parameters as below.

1. Chooses an additive group G with the prime order q defined on a curve E/F_p, and picks a generator P of G.
2. Selects the master secret key $x \in Z_q^*$, and computes the systems public key $P_{pub} = xP$.
3. Chooses three secure hash functions
   $H_1 : \{0,1\}^* \times G \times G \to Z_q^*$.
   $H_2 : \{0,1\}^* \times \{0,1\}^* \times G \times G \times G \times G \times G \to Z_q^*$.
   $H_3 : \{0,1\}^* \times \{0,1\}^* \times G \times G \times G \times G \times G \times G \to Z_q^*$.
4. Publishes the system parameters $params = \{G, q, P, P_{pub}, H_1, H_2, H_3\}$ and keeps x secret.

3.2. Registration phase

All users need to be registered with KGC. For a user with identity $ID_k \in \{0,1\}^*$, the long-term private is generated as follow. (Where the user may either be smart meter or service provider).

1. The user chooses at random a secret value $t_k \in Z_q^*$, and publishes $T_k = t_k P$ as public key.
2. The user sends the tuple (ID_k, T_k) to KGC.
3. KGC chooses at random $r_k \in Z_q^*$, computes $R_k = r_k P, h_k = H_2(ID_k, T_k, R_k)$ and $d_k = r_k + h_k x$.
4. KGC sends the partial private key pair (d_k, R_k) to the user by a secure channel.

3.3. Key agreement phase

Suppose that a smart meter $\mathcal{SM}_i$ want to agree a session key with a service provider $\mathcal{SP}_j$, following steps will are carried out.

1. $\mathcal{SM}_i$ chooses at random $a_i \in Z_q^*$, computes $M_i = a_i P$, then sends (R_i, M_i) to $\mathcal{SP}_j$.

   $\mathcal{SM}_i \to \mathcal{SP}_j : R_i, M_i$.

2. In response to the request, $\mathcal{SP}_j$ chooses at random $b_j \in Z_q^*$, computes $M_j = b_j P$, then sends (R_j, M_j) to $\mathcal{SM}_i$.

   $\mathcal{SP}_j \to \mathcal{SM}_i : R_j, M_j$.

3. $\mathcal{SM}_i$ computes

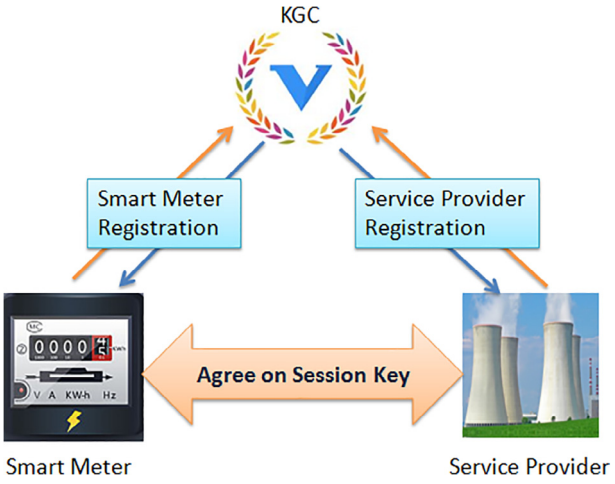

Fig. 1. Certificateless key agreement.

Smart Meter $\mathcal{SM}_i$		Service Provider $\mathcal{SP}_j$
Chooses at random $a_i \in Z_q^*$		
Computes $M_i = a_i P$	$\xrightarrow{R_i,\ M_i}$	
		Chooses at random $b_j \in Z_q^*$
	$\xleftarrow{R_j,\ M_j}$	Computes $M_j = b_j P$
$h_j = H_1(ID_j, T_j, R_j)$		$h_i = H_1(ID_i, T_i, R_i)$
$l_{ij} = H_2(ID_i, ID_j, T_i, T_j, R_i, R_j, M_i, M_j)$		$l_{ij} = H_2(ID_i, ID_j, T_i, T_j, R_i, R_j, M_i, M_j)$
$K_{ij} = (l_{ij}a_i + t_i + d_i)(l_{ij}M_j + T_j + R_j + h_j P_{pub})$		$K_{ji} = (l_{ij}b_j + t_j + d_j)(l_{ij}M_i + T_i + R_i + h_i P_{pub})$
$sk = H_3(ID_i, ID_j, T_i, T_j, R_i, R_j, M_i, M_j, K_{ij})$		$sk = H_3(ID_i, ID_j, T_i, T_j, R_i, R_j, M_i, M_j, K_{ji})$

Fig. 2. New CL2PAKA scheme.

$h_j = H_1(ID_j, T_j, R_j)$.
$l_{ij} = H_2(ID_i, ID_j, T_i, T_j, R_i, R_j, M_i, M_j)$.
$K_{ij} = (l_{ij}a_i + t_i + d_i)(l_{ij}M_j + T_j + R_j + h_j P_{pub})$.
$sk = H_3(ID_i, ID_j, T_i, T_j, R_i, R_j, M_i, M_j, K_{ij})$.

4. $\mathcal{SP}_j$ computes.

$h_i = H_1(ID_i, T_i, R_i)$.
$l_{ij} = H_2(ID_i, ID_j, T_i, T_j, R_i, R_j, M_i, M_j)$.
$K_{ji} = (l_{ij}b_j + t_j + d_j)(l_{ij}M_i + T_i + R_i + h_i P_{pub})$.
$sk = H_3(ID_i, ID_j, T_i, T_j, R_i, R_j, M_i, M_j, K_{ji})$.

- Consistency

$$\begin{aligned} K_{ij} &= (l_{ij}a_i + t_i + d_i)(l_{ij}M_j + T_j + R_j + h_j P_{pub}) \\ &= (l_{ij}a_i + t_i + d_i)(l_{ij}b_j + t_j + r_j + h_j x)P \\ &= (l_{ij}b_j + t_j + d_j)(l_{ij}a_i + t_i + r_i + h_i x)P \\ &= (l_{ij}b_j + t_j + d_j)(l_{ij}a_i P + t_i P + r_i P + h_i x P) \\ &= (l_{ij}b_j + t_j + d_j)(l_{ij}M_i + T_i + R_i + h_i P_{pub}) \\ &= K_{ji} \end{aligned}$$

$$\begin{aligned} sk &= H_3(ID_i, ID_j, T_i, T_j, R_i, R_j, M_i, M_j, K_{ij}) \\ &= H_3(ID_i, ID_j, T_i, T_j, R_i, R_j, M_i, M_j, K_{ji}) \end{aligned}$$

4. Security analysis

In this section, the scheme is proved to be secure in the eCK model, and the secure proofs are also done in the standard model. In other words, the adversary can get hash function value by computing the hash function instead of querying the challenger. In the security proofs, three lists L_i $(i = 1, 2, 3)$ are set to store the input–output pairs for hash function H_i $(i = 1, 2, 3)$, respectively. Once again stressed, the outputs of the hash functions are not randomly selected by the challenger $\mathcal{C}$, but are computed by the real hash function. In order to complete the security proofs, $\mathcal{C}$ needs to record the correspondence between the input and output of the hash functions.

Theorem 1. The session key generated in two matching sessions is the same.

Proof. The session key generated in two matching sessions is the same according to the consistency analysis.

Before further discussion, it is necessary to describe some important settings. Suppose that $\mathcal{A} \in \{\mathcal{A}_1, \mathcal{A}_2\}$ activates no more than n_1 honest parties, and each party is engaged in no more than n_2 sessions. Assume that $\mathcal{A}$ selects the $\prod_{I,J}^{S}$ as the test session. There are three ways for $\mathcal{A}$ to distinguish the test session key from the random string.

1. Guessing attack: $\mathcal{A}$ guesses the session key rightly.
2. Key replication attack: $\mathcal{A}$ establishes a non-matching session, which have the same session key with $\prod_{I,J}^{S}$. So $\mathcal{A}$ can get the session key by querying the non-matching session.
3. Forging attack: At some point, $\mathcal{A}$ computes the value $H_3(ID_I, ID_J, T_I, T_J, R_I, R_J, M_I^S, M_J^S, K_{IJ}^S)$. Namely, $\mathcal{A}$ obtains K_{IJ}^S itself.

Theorem 2. *The advantage of a Type I adversary against the new scheme is negligible in the standard model if the CDH problem is intractable.*

Proof. Let v be a secure parameter. Suppose that there exists a Type I adversary $\mathcal{A}_1$, that can win in the game with non-negligible advantage $Adv_{\mathcal{A}_1}(v)$. Then there exists a challenger $\mathcal{C}$, which can solve the CDH problem with non-negligible probability.

For Guessing attack, since the session key $sk \in Z_q^*$, the probability of correctly guessing the value of sk is $\frac{1}{q-1}$. For Key replication attack, the hash function H_3 needs to output the same result for two different input values. Since H_3 is a secure hash function, the probability of success of this attack is negligible.

Forging attack is analyzed as below. The eCK game is performed between a challenger $\mathcal{C}$ and a adversary $\mathcal{A}_1$. Let the tuple (P, uP, vP) be an instance of CDH problem. If $\mathcal{A}_1$ is successful in forging attack with non-negligible probability $Adv_{\mathcal{A}_1}(v)$, then the challenger $\mathcal{C}$ can utilize $\mathcal{A}_1$ to compute the value uvP with non-negligible probability. Before the game starts, $\mathcal{C}$ picks at random two integers $I, J \in \{1, n_1\}$ $(I \neq J)$, selects at random an integers $S \in \{1, n_2\}$, sets $\prod_{I,J}^{S}$ as the test session. So the probability that $\mathcal{C}$ guesses correctly the test session $\prod_{I,J}^{S}$ does not exceed $\frac{1}{n_1^2 \cdot n_2}$. Let $\prod_{J,I}^{E}$ be the matching session of $\prod_{I,J}^{S}$. There are six cases that need to be considered.

- CA1-1. $\prod_{J,I}^{E}$ exists, $\mathcal{A}_1$ does not obtain the ephemeral secret keys for ID_I and ID_J.
- CA1-2. $\prod_{J,I}^{E}$ exists, $\mathcal{A}_1$ does not obtain the ephemeral secret key of ID_I and does not obtain the partial private key of ID_J.
- CA1-3. $\prod_{J,I}^{E}$ exists, $\mathcal{A}_1$ does not obtain the partial private key of ID_I and does not obtain the ephemeral secret key of ID_J.
- CA1-4. $\prod_{J,I}^{E}$ exists, $\mathcal{A}_1$ does not obtain the partial private key for ID_I and ID_J.
- CA1-5. $\prod_{J,I}^{E}$ does not exist, $\mathcal{A}_1$ does not obtain the ephemeral secret key of ID_I and does not obtain the partial private key of ID_J.
- CA1-6. $\prod_{J,I}^{E}$ does not exist, $\mathcal{A}_1$ does not obtain the partial private keys for ID_I and ID_J.

In the following, the six cases will be analyzed separately.

The analysis of CA1-1.

Setup. Same as that in the new scheme. $\mathcal{C}$ then sends $params = \{G, q, P, P_{pub} = xP, H_1, H_2, H_3\}$ to the $\mathcal{A}_1$ and keeps x secret.

Query. $\mathcal{A}_1$ will query the public key before an identity is used in any other queries, and all queries are different. Several lists are set to store the queries and answers, them are initially empty. $\mathcal{C}$ answers the queries issued by $\mathcal{A}_1$ as follow.

- Query(PK). $\mathcal{C}$ maintains a list L_U of tuple $(ID_k, t_k, T_k, r_k, R_k)$. $\mathcal{A}_1$ submits an identity ID_k, $\mathcal{C}$ picks at random $t_k, r_k \in Z_q^*$, computes $T_k = t_kP$ and $R_k = r_kP$, then returns T_k and adds $(ID_k, t_k, T_k, r_k, R_k)$ to the list L_U.
- Replace(PK): $\mathcal{A}_1$ submits a tuple $T_k' = x_k'P$ for ID_k, $\mathcal{C}$ replaces T_k with T_k', and update $(ID_k, t_k, T_k, r_k, R_k)$ to $(ID_k, *, T_k', r_k, R_k)$ in the list L_U. Where $*$ can be the secret value t_k' or be the symbol $\perp$. Namely, $\mathcal{A}_1$ may submit the secret value or not.
- Query(SV): $\mathcal{A}_1$ submits an identity ID_k, $\mathcal{C}$ finds $(ID_k, t_k, T_k, r_k, R_k)$ in the list L_U and returns t_k. If $\mathcal{A}_1$ has replaced the public key T_k and did not submit the new secret value, then $\mathcal{C}$ may refuse to reply.
- Query(PPK): $\mathcal{A}_1$ submits an identity ID_k, $\mathcal{C}$ finds $(ID_k, t_k, T_k, r_k, R_k)$ in the list L_U, computes $h_k = H_1(ID_k, T_k, R_k), d_k = r_k + h_kx$ and returns d_k.
- Query(ESK): $\mathcal{C}$ maintains a list L_W of tuple $(ID_i, ID_j, s, a_i, b_j)$. $\mathcal{A}_1$ submits a session $\prod_{i,j}^{s}$, then $\mathcal{C}$ does as follow.
  1. If $\prod_{i,j}^{s} = \prod_{I,J}^{S}$ or $\prod_{i,j}^{s} = \prod_{J,I}^{E}$, then $\mathcal{C}$ failures and stops.
  2. Otherwise, $\mathcal{C}$ picks at random $a_i, b_j \in Z_q^*$, returns (a_i, b_j) and adds $(ID_i, ID_j, s, a_i, b_j)$ to the list L_W.
- Query(SK): $\mathcal{A}_1$ submits a session $\prod_{i,j}^{s}$, $\mathcal{C}$ does as follow. If $\mathcal{A}_1$ has replaced the public key T_i (or T_j) and did not submit the new secret value t_i' (or t_j'), then $\mathcal{C}$ may refuse to reply.
  1. If $\prod_{i,j}^{s} = \prod_{I,J}^{S}$ or $\prod_{i,j}^{s} = \prod_{J,I}^{E}$, then $\mathcal{C}$ failures and stops.

Table 2
Comparison of several CL2PAKA schemes.

Scheme	Message size (byte)	Running Time (ms)	Security model
Goya [13]	$2\|G_1\|$ (128)	$6B_P + 14S_{G_1}$ (383.948)	ROM
Lin [26]	$\|G_1\|$ (64)	$3B_P + 2S_{G_1} + E_{G_2}$ (127.198)	ROM
Sun [38]	$4\|G\|$ (80)	$12S_G$ (40.02)	ROM
Tu [39]	$2\|G\|$ (40)	$5S_G$ (16.675)	ROM
New scheme	$2\|G\|$ (40)	$4S_G$ (13.340)	Standard model

2. If $\mathcal{A}_1$ has made Query(ESK) for $\prod_{i,j}^{s}$, $\mathcal{C}$ finds $(ID_i, ID_j, s, a_i, b_j)$ in the list L_W, finds $(ID_i, t_i, T_i, r_i, R_i)$ or $(ID_j, t_j, T_j, r_j, R_j)$ in list L_U, computes $h_k = H_1(ID_k, T_k, R_k), d_k = r_k + h_k x$ for $k = i$ or j, then computes session key by performing the Key Agreement algorithm.
3. Otherwise, $\mathcal{C}$ picks at random $a_i, b_j \in Z_q^*$ and adds $(ID_i, ID_j, s, a_i, b_j)$ to the list L_W, then does as that in case 2).

- Send($\prod_{i,j}^{s}, m$). $\mathcal{C}$ responds to queries as follow.
  1. If $(\prod_{i,j}^{s}, m) = (\prod_{I,J}^{S}, \perp)$, $\mathcal{C}$ finds $(ID_I, t_I, T_I, r_I, R_I)$ in the list L_U, then returns (R_I, uP).
  2. If $(\prod_{i,j}^{s}, m) = (\prod_{J,I}^{E}, \perp)$, $\mathcal{C}$ finds $(ID_J, t_J, T_J, r_J, R_J)$ in the list L_U, then returns (R_J, vP).
  3. If $(\prod_{i,j}^{s}, m) = (\prod_{i,j}^{s}, \perp)$, where $\prod_{i,j}^{s} \neq \prod_{I,J}^{S}$ and $\prod_{i,j}^{s} \neq \prod_{J,I}^{E}$. $\mathcal{C}$ finds $(ID_i, t_i, T_i, r_i, R_i)$ in the list L_U and does as follows.
  (a) If $\mathcal{A}_1$ has made Query(ESK) for $\prod_{i,j}^{s}$, $\mathcal{C}$ finds $(ID_i, ID_j, s, a_i, b_j)$ in the list L_W, then returns $(R_i, a_i P)$.
  (b) Otherwise, $\mathcal{C}$ picks at random $a_i, b_j \in Z_q^*$, returns $(R_i, a_i P)$ and adds $(ID_i, ID_j, s, a_i, b_j)$ to the list L_W.
  4. If m is the second message in the session, namely $m = (R_j, *)$, $\mathcal{C}$ accepts the session.
- Test($\prod_{i,j}^{s}$). $\mathcal{A}_1$ must submit the new secret value t_i' (or t_j') to $\mathcal{C}$ if the public key T_i (or T_j) has been replaced to T_i' (or T_j'). This is a reasonable request, since $\mathcal{C}$ cannot generate the session key if he does not know the secret values for ID_i and ID_j. $\mathcal{C}$ responds to query as follow.
  1. If $\prod_{i,j}^{s} \neq \prod_{I,J}^{S}$, $\mathcal{C}$ failures and stops.
  2. If $\prod_{i,j}^{s} = \prod_{I,J}^{S}$, $\mathcal{C}$ picks at random a number $sk \in Z_q^*$ and returns it to $\mathcal{A}_1$.

Solve CDH problem. If $\mathcal{A}_1$ wins in the game by forging attack, he must compute $H_3(ID_I, ID_J, T_I, T_J, R_I, R_J, M_I^S, M_J^S, K_{IJ}^S)$, where $M_I^S = uP, M_J^S = vP$ and $K_{IJ}^S = (l_{IJ} \cdot u + t_I + r_I + h_I \cdot x)(l_{IJ} \cdot v + t_J + r_J + h_J \cdot x)P$. $\mathcal{C}$ finds $(ID_I, ID_J, T_I, T_J, R_I, R_J, M_I^S, M_J^S, K_{IJ}^S)$ in the list L_3, and finds (t_I, r_I) and (t_J, r_J) in the list L_U, then computes $h_I = H_1(ID_I, T_I, R_I), h_J = H_1(ID_J, T_J, R_J)$, $l_{IJ} = H_2(ID_I, ID_J, T_I, T_J, R_I, R_J, M_I^S, M_J^S)$, $Z_1 = (t_I + r_I + h_I x)(t_J + r_J + h_J x)P, Z_2 = l_{IJ}(t_I + r_I + h_I x) \cdot vP$ and $Z_3 = l_{IJ}(t_J + r_J + h_J x) \cdot uP$, final solves CDH problem by computing: $uvP = l_{IJ}^{-2}(K_{IJ}^S - Z_1 - Z_2 - Z_3)$.

Probability. If $\mathcal{C}$ correctly guesses the test session $\prod_{I,J}^{S}$, then he will not fail during the query phase. So $\mathcal{C}$ can compute the value uvP with probability $\frac{1}{n_1^2 n_2} Adv_{\mathcal{A}_1}(v)$ if $\mathcal{A}_1$ wins in the game with advantage $Adv_{\mathcal{A}_1}(v)$.

The analysis of CA1-2.

Setup. Same as that in the analysis of CA1-1.

Query. $\mathcal{C}$ responds to the queries from $\mathcal{A}_1$ as those in the analysis of CA1-1 except for the Query(PK), Query(PPK), Query (ESK) and Send($\prod_{i,j}^{s}, m$).

- Query(PK): $\mathcal{C}$ maintains a list L_U of tuple $(ID_k, t_k, T_k, r_k, R_k)$. $\mathcal{A}_1$ submits an identity ID_k, $\mathcal{C}$ does as follow.
  1. If $ID_k = ID_J$, $\mathcal{C}$ picks at random $t_J \in Z_q^*$ and sets $T_J = t_J P$ and $R_J = vP$, then returns T_J and adds $(ID_J, t_J, T_J, \perp, R_J)$ to the list L_U.
  2. Otherwise, $\mathcal{C}$ picks at random $t_k, r_k \in Z_q^*$, sets $T_k = t_k P$ and $R_k = r_k P$, then returns T_k and adds $(ID_k, t_k, T_k, r_k, R_k)$ to the list L_U.
- Query(PPK): $\mathcal{A}_1$ submits an identity ID_k, if $ID_k = ID_J$, then $\mathcal{C}$ failures and stops. Otherwise, $\mathcal{C}$ finds $(ID_k, t_k, T_k, r_k, R_k)$ in the list L_U, computes $h_k = H_1(ID_k, T_k, R_k), d_k = r_k + h_k x P$ and returns d_k.
- Query(ESK): $\mathcal{C}$ maintains a list L_W of tuple $(ID_i, ID_j, s, a_i, b_j)$. $\mathcal{A}_1$ submits a session $\prod_{i,j}^{s}$, $\mathcal{C}$ does as follow.
  1. If $\prod_{i,j}^{s} = \prod_{I,J}^{S}$ or $\prod_{i,j}^{s} = \prod_{J,I}^{E}$, $\mathcal{C}$ picks at random $b_J \in Z_q^*$, returns $(\perp, b_J)$ and adds $(ID_I, ID_J, S, \perp, b_J)$ to the list L_W.
  2. Otherwise, $\mathcal{C}$ picks at random $a_i, b_j \in Z_q^*$, returns (a_i, b_j) and adds $(ID_i, ID_j, s, a_i, b_j)$ to the list L_W.
- Send($\prod_{i,j}^{s}, m$). $\mathcal{C}$ responds to queries as follow.
  1. If $(\prod_{i,j}^{s}, m) = (\prod_{I,J}^{S}, \perp)$, $\mathcal{C}$ finds $(ID_I, t_I, T_I, r_I, R_I)$ in the list L_U, then returns (R_I, uP).
  2. If $(\prod_{i,j}^{s}, m) = (\prod_{J,I}^{E}, \perp)$, $\mathcal{C}$ finds $(ID_J, t_J, T_J, *, R_J)$ in the list L_U and finds $(ID_I, ID_J, S, \perp, b_J)$ in the list L_W, returns $(R_J, b_J P)$.
  3. Otherwise, same as that in the analysis of CA1-1.

Solve CDH problem. If $\mathcal{A}_1$ wins in the game by forging attack, he must compute $H_3(ID_I, ID_J, T_I, T_J, R_I, R_J, M_I^S, M_J^S, K_{IJ}^S)$, where $M_I^S = uP, R_J = vP$ and $K_{IJ}^S = (l_{IJ} \cdot u + t_I + r_I + h_I \cdot x)(l_{IJ} \cdot b_J + t_J + v + h_J \cdot x)P$. $\mathcal{C}$ finds $(ID_I, ID_J, T_I, T_J, R_I, R_J, M_I^S, M_J^S, K_{IJ}^S)$ in the list L_3, finds b_J in the list L_W and finds (t_I, r_I) and $(t_J, \perp)$ in the list L_U, computes $h_I = H_1(ID_I, T_I, R_I), h_J = H_1(ID_J, T_J, R_J)$, $l_{IJ} = H_2(ID_I, ID_J, T_I, T_J, R_I, R_J, M_I^S, M_J^S)$, $Z_1 = (t_I + r_I + h_I x)(l_{IJ} b_J + t_J + h_J x)P, Z_2 = (t_I + r_I + h_I x)vP$ and $Z_3 = l_{IJ}(l_{IJ} b_J + t_J + h_J x) \cdot uP$, final solves CDH problem by computing: $uvP = l_{IJ}^{-1}(K_{IJ}^S - Z_1 - Z_2 - Z_3)$.

Probability. Same as that in the analysis of CA1-1.

The analysis of CA1-3.

$\mathcal{C}$ exchanges the roles of ID_I and ID_J in the CA1-2, then does as that in the analysis of CA1-2.

The analysis of CA1-4.

Setup. Same as that in the analysis of CA1-1.

Query. $\mathcal{C}$ responds to the queries from $\mathcal{A}_1$ as those in the analysis of CA1-1 except for the Query(PK), Query(PPK), Query (ESK), Query(SK) and Send($\prod_{i,j}^{s}, m$).

- Query(PK): $\mathcal{C}$ maintains a list L_U of tuple $(ID_k, t_k, T_k, r_k, R_k)$. $\mathcal{A}_1$ submits an identity $ID_k, \mathcal{C}$ does as follow.
  1. If $ID_k = ID_I, \mathcal{C}$ picks at random $t_I \in Z_q^*$, sets $T_I = t_I P$ and $R_I = uP$, then returns T_I and adds $(ID_I, t_I, T_I, \perp, R_I)$ to the list L_U.
  2. If $ID_k = ID_J, \mathcal{C}$ picks at random $t_J \in Z_q^*$, sets $T_J = t_J P$ and $R_J = vP$, then returns T_J and adds $(ID_J, t_J, T_J, \perp, R_J)$ to the list L_U.
  3. Otherwise, $\mathcal{C}$ picks at random $t_k, r_k \in Z_q^*$, returns $T_k = t_k P$ and $R_k = r_k P$, then returns T_k and adds $(ID_k, t_k, T_k, r_k, R_k)$ in the list L_U.
- Query(PPK): $\mathcal{A}_1$ submits an identity ID_k, if $ID_k = ID_I$ or $ID_k = ID_J$, then $\mathcal{C}$ failures and stops. Otherwise, $\mathcal{C}$ finds $(ID_k, t_k, T_k, r_k, R_k)$ in the list L_U, computes $h_k = H_1(ID_k, T_k, R_k), d_k = r_k + h_k x$ and returns d_k.
- Query(ESK): $\mathcal{C}$ maintains a list L_W of tuple $(ID_i, ID_j, s, a_i, b_j)$. $\mathcal{A}_1$ submits a session $\prod_{i,j}^{s}, \mathcal{C}$ picks at random $a_i, b_j \in Z_q^*$, returns (a_i, b_j) and adds $(ID_i, ID_j, s, a_i, b_j)$ to the list L_W.
- Query(SK): $\mathcal{A}_1$ submits a session $\prod_{i,j}^{s}, \mathcal{C}$ does as follow. If $\mathcal{A}_1$ has replaced the public key T_i (or T_j) and did not submit the new secret value t_i' (or t_j'), then $\mathcal{C}$ may refuse to reply.
  1. If $\{ID_i, ID_j\} = \{ID_I, ID_J\}$, then $\mathcal{C}$ failures and stops.
  2. Otherwise, same as that in the analysis of CA1-1.
- Send($\prod_{i,j}^{s}, m$). $\mathcal{C}$ finds $(ID_i, t_i, T_i, r_i, R_i)$ in the list L_U, then responds to queries as follow.
  1. If $(\prod_{i,j}^{s}, m) = (\prod_{i,j}^{s}, \perp), \mathcal{C}$ does as follows.
  (a) $\mathcal{A}_1$ has made Query(ESK) for $\prod_{i,j}^{s}$, $\mathcal{C}$ finds $(ID_i, ID_j, s, a_i, b_j)$ in the list L_W, returns $(R_i, a_i P)$.
  (b) Otherwise, $\mathcal{C}$ picks at random $a_i, b_j \in Z_q^*$, returns $(R_i, a_i P)$ and adds $(ID_i, ID_j, s, a_i, b_j)$ to the list L_W.
  2. m is the second message in the session, namely $m = (R_j, *)$, $\mathcal{C}$ accepts the session.

Solve CDH problem. If $\mathcal{A}_1$ wins in the game by forging attack, he must compute $H_3(ID_I, ID_J, T_I, T_J, R_I, R_J, M_I^S, M_J^S, K_{IJ}^S)$, where $R_I = uP, R_J = vP$ and $K_{IJ}^S = (l_{IJ} \cdot a_I + t_I + u + h_I \cdot x)(l_{IJ} \cdot b_J + t_J + v + h_J \cdot x)P$. $\mathcal{C}$ finds $(ID_I, ID_J, T_I, T_J, R_I, R_J, M_I^S, M_J^S, K_{IJ}^S)$ in the list L_3, finds (a_I, b_J) in the list L_W and finds $(t_I, \perp)$ and $(t_J, \perp)$ in the list L_U, computes $h_I = H_1(ID_I, T_I, R_I), h_J = H_1(ID_J, T_J, R_J), l_{IJ} = H_2(ID_I, ID_J, T_I, T_J, R_I, R_J, M_I^S, M_J^S)$,
$Z_1 = (l_{IJ}a_I + t_I + h_I x)(l_{IJ}b_J + t_J + h_J x)P, Z_2 = (l_{IJ}a_I + t_I + h_I x)vP$ and $Z_3 = (l_{IJ}b_J + t_J + h_J x) \cdot uP$, final solves CDH problem by computing: $uvP = K_{IJ}^S - Z_1 - Z_2 - Z_3$.

Probability. If $\mathcal{C}$ correctly guesses the test session $\prod_{I,J}^{S}$, then he will not fail during the query phase except for the Query (SK). The probability of $\mathcal{C}$ did not fail in Query(SK) is $1 - \frac{2}{n_1(n_1-1)} > \frac{1}{2}$. So $\mathcal{C}$ can compute the value uvP with probability $\frac{1}{2n_1^2 n_2} Adv_{\mathcal{A}_1}(v)$ if $\mathcal{A}_1$ wins in the game with advantage $Adv_{\mathcal{A}_1}(v)$.

The analysis of CA1-5 is similar to that of CA1-2.

The only difference between CA1-2 and CA1-5 is that there is an matching session $\prod_{J,I}^{E}$ for the test session $\prod_{I,J}^{S}$ in CA1-2, but there is not matching session for $\prod_{I,J}^{S}$ in CA1-5. So the analysis of CA1-5 is similar to that of CA1-2.

The analysis of CA1-6 is similar to that of CA1-4.

The only difference between CA1-4 and CA1-6 is that there is an matching session $\prod_{J,I}^{E}$ for the test session $\prod_{I,J}^{S}$ in CA1-4, but there is not matching session for $\prod_{I,J}^{S}$ in CA1-6. So the analysis of CA1-6 is similar to that of CA1-4.

Theorem 3. *The advantage of a Type II adversary against the new scheme is negligible if the CDH problem is intractable.*

Proof. Suppose that v is a secure parameter and there exist a Type II adversary $\mathcal{A}_2$, that can win in the game with non-negligible advantage $Adv_{\mathcal{A}_2}(v)$. Then there exists a challenger $\mathcal{C}$, which can solve the CDH problem with non-negligible probability.

For guessing attack and key replication attack, the analysis are same as that in the Theorem 2.

Forging attack is analyzed as below. The eCK game with secure parameter v is performed between a challenger $\mathcal{C}$ and a adversary $\mathcal{A}_2$. Let the tuple (P, uP, vP) be an instance of CDH problem. If $\mathcal{A}_2$ is successful in forging attack with non-negligible probability $Adv_{\mathcal{A}_2}(v)$, then the challenger $\mathcal{C}$ can utilize $\mathcal{A}_2$ to compute the value uvP with non-negligible probability. Before the game starts, $\mathcal{C}$ picks at random two integers $I, J \in \{1, n_1\}$ $(I \neq J)$, selects at random an integers $S \in \{1, n_2\}$, sets $\prod_{I,J}^{S}$ as the test session. So the probability that $\mathcal{C}$ guesses correctly the test session $\prod_{I,J}^{S}$ does not exceed $\frac{1}{n_1^2 \cdot n_2}$. Let $\prod_{J,I}^{E}$ be the matching session of $\prod_{I,J}^{S}$. There are six cases that need to be considered.

- CA2-1. $\prod_{J,I}^{E}$ exists, $\mathcal{A}_2$ does not obtain the ephemeral secret key for ID_I and ID_J.
- CA2-2. $\prod_{J,I}^{E}$ exists, $\mathcal{A}_2$ does not obtain the ephemeral secret key of ID_I and does not obtain the secret value of ID_J.
- CA2-3. $\prod_{J,I}^{E}$ exists, $\mathcal{A}_2$ does not obtain the secret value of ID_I and does not obtain the ephemeral secret key of ID_J.
- CA2-4. $\prod_{J,I}^{E}$ exists, $\mathcal{A}_2$ does not obtain the secret values for ID_I and ID_J.
- CA2-5. $\prod_{J,I}^{E}$ does not exist, $\mathcal{A}_2$ does not obtain the ephemeral secret key of ID_I and does not obtain the secret value of ID_J.
- CA2-6. $\prod_{J,I}^{E}$ does not exists, $\mathcal{A}_2$ does not obtain the secret values for ID_I and ID_J.

In the following, the six cases will be analyzed separately.

The analysis of CA2-1.

Setup. Same as that in the new scheme. $\mathcal{C}$ then sends $params = \{G, q, P, P_{pub} = xP, H_1, H_2, H_3\}$ and $msk = \{x\}$ to $\mathcal{A}_2$.

The analysis of the rest is exactly the same as that in the analysis of CA1-1.

The analysis of CA2-2.

Setup. Same as that in the analysis of CA2-1.

$\mathcal{C}$ responds to the queries from $\mathcal{A}_2$ as those in the analysis of CA1-1 except for the Query(PK), Query(SV), Query(ESK) and Send($\prod_{i,j}^{s}, m$).

- Query(PK): $\mathcal{C}$ maintains a list L_U of tuple $(ID_k, t_k, T_k, r_k, R_k)$. $\mathcal{A}_2$ submits an identity ID_k, $\mathcal{C}$ does as follow.
  1. If $ID_k = ID_J$, $\mathcal{C}$ picks at random $r_J \in Z_q^*$ and returns $T_J = vP$ and $R_J = r_JP$, then returns T_J and adds $(ID_J, \perp, T_J, r_J, R_J)$ to the list L_U.
  2. Otherwise, $\mathcal{C}$ picks at random $t_k, r_k \in Z_q^*$, returns $T_k = t_kP$ and $R_k = r_kP$ then returns T_k and adds $(ID_k, t_k, T_k, r_k, R_k)$ in the list L_U.
- Query(SV): $\mathcal{A}_2$ submits an identity ID_k, if $ID_k = ID_J$, then $\mathcal{C}$ failures and stops. Otherwise, $\mathcal{C}$ finds $(ID_k, t_k, T_k, r_k, R_k)$ in the list L_U and returns t_k.
- Query(ESK): $\mathcal{C}$ maintains a list L_W of tuple $(ID_i, ID_j, s, a_i, b_j)$. $\mathcal{A}_2$ submits a session $\prod_{i,j}^{s}, \mathcal{C}$ does as follow.
  1. If $\prod_{i,j}^{s} = \prod_{I,J}^{S}$ or $\prod_{i,j}^{s} = \prod_{J,I}^{E}$, $\mathcal{C}$ picks at random $b_J \in Z_q^*$, returns $(\perp, b_J)$ and adds $(ID_I, ID_J, S, \perp, b_J)$ to the list L_W.
  2. Otherwise, $\mathcal{C}$ picks at random $a_i, b_j \in Z_q^*$, returns (a_i, b_j) and adds $(ID_i, ID_j, s, a_i, b_j)$ to the list L_W.
- Send($\prod_{i,j}^{s}, m$). $\mathcal{C}$ responds to queries as follow.
  1. If $(\prod_{i,j}^{s}, m) = (\prod_{I,J}^{S}, \perp)$, $\mathcal{C}$ finds $(ID_I, t_I, T_I, r_I, R_I)$ in the list L_U, then returns (R_I, uP).
  2. If $(\prod_{i,j}^{s}, m) = (\prod_{J,I}^{E}, \perp)$, $\mathcal{C}$ finds $(ID_J, t_J, T_J, r_J, R_J)$ in the list L_U and finds $(ID_I, ID_J, S, \perp, b_J)$ in the list L_W, then returns (R_J, b_JP).
  3. Otherwise, same as that in the analysis of CA1-1.

Solve CDH problem. If $\mathcal{A}_2$ wins in the game by forging attack, he must compute $H_3(ID_I, ID_J, T_I, T_J, R_I, R_J, M_I^S, M_J^S, K_{IJ}^S)$, where $M_I^S = uP, T_J = vP$ and $K_{IJ}^S = (l_{IJ} \cdot u + t_I + r_I + h_I \cdot x)(l_{IJ} \cdot b_J + v + r_J + h_J \cdot x)P$. $\mathcal{C}$ finds $(ID_I, ID_J, T_I, T_J, R_I, R_J, M_I^S, M_J^S, K_{IJ}^S)$ in the list L_3, finds b_J in the list L_W and finds (t_I, r_I) and $(\perp, r_J)$ in the list L_U, computes $h_I = H_1(ID_I, T_I, R_I), h_J = H_1(ID_J, T_J, R_J), l_{IJ} = H_2(ID_I, ID_J, T_I, T_J, R_I, R_J, M_I^S, M_J^S)$, $Z_1 = (t_I + r_I + h_Ix)(l_{IJ}b_J + r_J + h_Jx)P, Z_2 = (t_I + r_I + h_Ix)vP$ and $Z_3 = l_{IJ}(l_{IJ}b_J + r_J + h_Jx) \cdot uP$, final solves CDH problem by computing: $uvP = l_{IJ}^{-1}(K_{IJ}^S - Z_1 - Z_2 - Z_3)$.

Probability. Same as that in the analysis of CA1-1.

The analysis of CA2-3.

$\mathcal{C}$ exchanges the roles of ID_I and ID_J in the CA2-2, then does as that in the analysis of CA2-2.

The analysis of CA2-4.

Setup. Same as that in the analysis of CA2-1.

Query. $\mathcal{C}$ responds to the queries from $\mathcal{A}_2$ as those in the analysis of CA1-1 except for the Query(PK), Query(SV), Query (ESK), Query(SK) and Send($\prod_{i,j}^{s}, m$).

- Query(PK): $\mathcal{C}$ maintains a list L_U of tuple $(ID_k, t_k, T_k, r_k, R_k)$. $\mathcal{A}_2$ submits an identity ID_k, $\mathcal{C}$ does as follow.
  1. If $ID_k = ID_I$, $\mathcal{C}$ picks at random $r_I \in Z_q^*$, sets $T_I = uP$ and $R_I = r_IP$, then returns T_I and adds $(ID_I, \perp, T_I, r_I, R_I)$ to the list L_U.
  2. If $ID_k = ID_J$, $\mathcal{C}$ picks at random $r_J \in Z_q^*$, sets $T_J = vP$ and $R_J = r_JP$, then returns T_J and adds $(ID_J, \perp, T_J, r_J, R_J)$ to the list L_U.
  3. Otherwise, $\mathcal{C}$ picks at random $t_k, r_k \in Z_q^*$, sets $T_k = t_kP$ and $R_k = r_kP$, then returns T_k and adds $(ID_k, t_k, T_k, r_k, R_k)$ in the list L_U.
- Query(SV): $\mathcal{A}_2$ submits an identity ID_k, if $ID_k = ID_I$ or $ID_k = ID_J$, then $\mathcal{C}$ failures and stops. Otherwise, $\mathcal{C}$ finds $(ID_k, t_k, T_k, r_k, R_k)$ in the list L_U and returns t_k.
- Query(ESK): $\mathcal{C}$ maintains a list L_W of tuple $(ID_i, ID_j, s, a_i, b_j)$. $\mathcal{A}_2$ submits a session $\prod_{i,j}^{s}, \mathcal{C}$ picks at random $a_i, b_j \in Z_q^*$, returns (a_i, b_j) and adds $(ID_i, ID_j, s, a_i, b_j)$ to the list L_W.
- Query(SK). Same as that in the analysis of CA1-4.

- Send($\prod_{i,j}^{s}, m$). Same as that in the analysis of CA1-4.

Solve CDH problem. If $\mathcal{A}_2$ wins in the game by forging attack, he must compute $H_3(ID_I, ID_J, T_I, T_J, R_I, R_J, M_I^S, M_J^S, K_{IJ}^S)$, where $T_I = uP, T_J = vP$ and $K_{IJ}^S = (l_{IJ} \cdot a_I + u + r_I + h_I \cdot x)(l_{IJ} \cdot b_J + v + r_J + h_J \cdot x)P$. $\mathcal{C}$ finds $(ID_I, ID_J, T_I, T_J, R_I, R_J, M_I^S, M_J^S, K_{IJ}^S)$ in the list L_3, finds (a_I, b_J) in the list L_W and finds $(\perp, r_I)$ and $(\perp, r_J)$ in the list L_U, computes $h_I = H_1(ID_I, T_I, R_I), h_J = H_1(ID_J, T_J, R_J), l_{IJ} = H_2(ID_I, ID_J, T_I, T_J, R_I, R_J, M_I^S, M_J^S)$, $Z_1 = (l_{IJ}a_I + r_I + h_I x)(l_{IJ}b_J + r_J + h_J x)P, Z_2 = (l_{IJ}a_I + r_I + h_I x)vP$ and $Z_3 = (l_{IJ}b_J + r_J + h_J x) \cdot uP$, final solves CDH problem by computing: $uvP = K_{IJ}^S - Z_1 - Z_2 - Z_3$.

Probability. Same as that in the analysis of CA1-4.

The analysis of CA2-5.

The only difference between CA2-2 and CA2-5 is that there is an matching session $\prod_{J,I}^{E}$ for the test session $\prod_{I,J}^{S}$ in CA2-2, but there is not matching session for $\prod_{I,J}^{S}$ in CA2-5. So the analysis of CA2-5 is similar to that of CA2-2.

The analysis of CA2-6.

The only difference between CA2-4 and CA2-6 is that there is an matching session $\prod_{J,I}^{E}$ for the test session $\prod_{I,J}^{S}$ in CA2-4, but there is not matching session for $\prod_{I,J}^{S}$ in CA2-6. So the analysis of CA2-6 is similar to that of CA2-4.

5. Efficiency and comparison

In this section, we make a performance comparison between the new scheme and four CL2PAKA schemes from the last three years. Several notations are listed in Table 3.

For fairness and credibility, third-party data is used to analyze several CL2PAKA schemes. Performing the related operations on a Samsung Galaxy S5 (Google Android 4.4.2 operating system, Quad-core 2.45G processor and 2G bytes memory). He et al. [19] obtains the time overhead on basic cryptographic operations (Table 4). To achieve 1024-bit RSA level security, an Ate pairing $\hat{e} : G_1 \times G_1 \rightarrow G_2$ was used, where G_1 is an additive group with the prime order q, which is defined on a super singular elliptic curve $y^2 = x^3 + 1$ over a finite field F_p, and the sizes of q and p are 160 bits and 512 bits, respectively. An additive group G with the prime order q was used, which is defined on a non-singular elliptic curve over a prime field F_p, where both sizes of p and q are 160 bits.

The two schemes [13,26] require bilinear pairing operations, resulting in higher computation costs. In the scheme [38], the private key of the user includes 4 points in Z_q^* and 2 points in G, the public key of the user includes 2 points in G. In the scheme [39] and the new scheme, the private key of the user includes 2 points in Z_q^* and 1 point in G, the public key of the user includes only 1 point in G. In the calculation of the shared secret, each entity in the scheme [39] needs to compute 2 values in G. Each entity in the new scheme needs to compute only 1 value in G. A detailed comparison of the five CL2PAPA schemes is as follows.

A simple method is used to evaluate the computation cost (Fig.3. In Goya et al.'s scheme [13], each entity in the communication needs to perform 6 pairing operations and 14 scale multiplication operations in G_1, so the running time is $32.713 \times 6 + 13.405 \times 14 = 383.948$ (ms). In Lin's scheme [26], each entity in the communication needs to perform 3 pairing operations, 2 scale multiplication operations in G_1 and 1 exponentiation operation in G_2, so the running time is $32.713 \times 3 + 13.405 \times 2 + 2.249 = 127.198$ (ms). In Sun et al.'s scheme [38], each entity in the communication needs to perform 12 scale multiplication operations in G, so the running time is $3.335 \times 12 = 40.02$ (ms). In Tu et al.'s scheme [39], each entity in the communication needs to perform 5 scale multiplication operations in G, so the running time is $3.335 \times 5 = 16.675$ (ms). In the new scheme, each entity in the communication needs to perform 4 scale multiplication operations in G, so the running time is $3.335 \times 4 = 13.340$ (ms).

Next, the size of message transmitted in the communication are computed (Fig.4. In Goya et al.'s scheme [13], each entity needs to send 2 points from G_1, so the size of message is $(512 \times 2)/8 = 128$ (bytes). In Lin's scheme [26], each entity needs to send 1 point from G_1, so the size of message is $512/8 = 64$ (bytes). In Sun et al.'s scheme [38], each entity needs to send 4 points from G, so the size of message is $(160 \times 4)/8 = 80$ (bytes). In Tu et al.'s scheme [39], each entity needs to send 2 points

Table 3
Cryptographic operation symbols.

Symbol	Meaning
B_P	A bilinear pairing operation.
S_{G_1}	A scale multiplication operation in G_1.
E_{G_2}	An exponentiation operation in G_2.
S_G	A scale multiplication operation in G.
$\|G_1\|$	An element in G_1.
$\|G\|$	An element in G.
$\|Z_q^*\|$	An element in Z_q^*.

Table 4
Cryptographic operation time (in milliseconds).

B_p	S_{G_1}	E_{G_2}	S_G
32.713	13.405	2.249	3.335

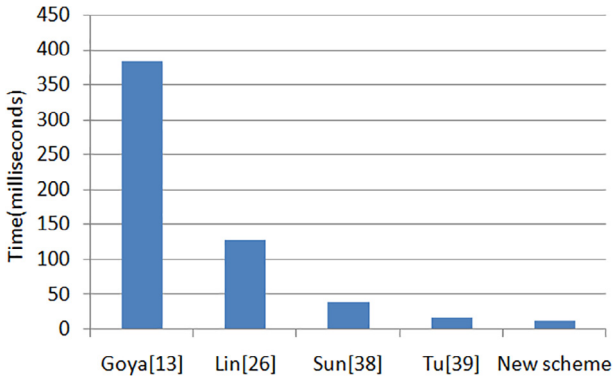

Fig. 3. Computation cost.

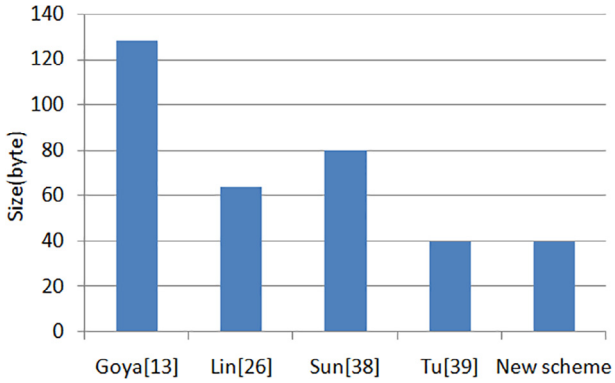

Fig. 4. Message size.

from G, so the size of message is $(160 \times 2)/8 = 40$ (bytes). In the new scheme, each entity needs to send 2 points from G, so the size of message is $(160 \times 2)/8 = 40$ (bytes). The detailed comparison results of five CL2PAKA schemes are illustrated in Table 2.

According to the analysis above, it can be concluded that the computation cost of each entity in the new scheme is 3.47% of that in the scheme [13], 10.49% of that in the scheme [26], 33.33% of that in the scheme [38], 80% of that in the scheme [39]. The size of the message to be transmitted in new scheme is 31.25% of that of the scheme [13], 62.5% of that in the scheme [26], 50% of that in the scheme [38], same as that in the scheme [39].

6. Conclusion

Authentication key agreement is an important technology to achieve secure communication between smart meters and service providers. Pairing is a relatively expensive operation and does not apply to smart meters with low-power. In recent years, several pairing-free CL2PAKA schemes were proposed, however, it was in ROM that the security proofs of these schemes were shown. It is well known, a cryptographic scheme is not necessarily safe in real life even though it has been proven to be secure in ROM. In this paper, we construct a new CL2PAKA scheme and show the security proofs in eCK model and the standard model. Our scheme does not require pairing operation and requires only four scalar multiplication operations on elliptic curve. Performance analysis shows that the computation costs and storage costs of our scheme are lower than that of previous schemes. So it is suitable for smart grids.

CRediT authorship contribution statement

Lunzhi Deng: Conceptualization, Methodology, Formal analysis, Writing - original draft. **Ronghai Gao:** Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This research is supported by the National Natural Science Foundation of China under Grants No. 61962011, the Innovation Group Major Research Projects of Department of Education of Guizhou Province under Grant No. KY[2016]026, the Guizhou Provincial Science and Technology Foundation under Grant No. [2014]1434.

References

[1] S.S. Al-Riyami, K.G. Paterson, Certificateless public cryptography. Advances in Cryptology-Asiacrypt 2003, LNCS, vol. 2894, 2003, pp. 452–473.
[2] M. Bellare, P. Rogaway, Entity authentication and key distribution, Advances in Cryptology-Crypto'93, LNCS, vol.773, 1993, pp. 232–249.
[3] S. Blake-Wilson, D. Johnson, A. Menezes, Key agreement protocols and their security analysis, in: 6th IMA International Conference on Cryptography and Coding, LNCS, vol. 1355, 1997, pp. 30–45.
[4] S. Bala, G. Sharma, A.K. Verma, PF-ID-2PAKA: pairing free identity-based two-party authenticated key agreement protocol for wireless sensor networks, Wireless Pers. Commun. 87 (3) (2016) 995–1012.
[5] S. Bala, G. Sharma, A.K.A. Verma, non-interactive certificateless two-party authenticated key agreement protocol for wireless sensor networks, Int. J. Ad Hoc Ubiq Co. 21 (2) (2016) 140–155.
[6] S. Bala, G. Sharma, A.K. Verma, Impersonation attack on certificateless key agreement protocol, Int. J. Ad Hoc Ubiq Co. 27 (2) (2018) 108–120.
[7] R. Canetti, H. Krawczyk, Analysis of key-exchange protocols and their use for building secure channels, Advances in Cryptology-Eurocrypt 2001, LNCS, vol. 2045, 2001, pp. 453–474.
[8] R. Canetti, O. Goldreich, T. Halev, The random oracle methodology revisited, J. ACM 51 (4) (2004) 557–594.
[9] X. Cao, W. Kou, X.A. Du, Pairing-free identity-based authenticated key agreement protocol with minimal message exchanges, Inf. Sci. 180 (2010) 2895–2903.
[10] Q. Cheng, Cryptanalysis of an efficient certificateless two-party authenticated key agreement protocol,https://eprint.iacr.org/2012/725.pdf.
[11] L. Dang, J. Xu, X. Cao, H. Li, J. Chen, Y. Zhang, X. Fu, Efficient identity-based authenticated key agreement protocol with provable security for vehicular ad hoc networks, Int. J. Distrib. Sens. N 14 (4) (2018) 1–16.
[12] M. Geng, F. Zhang, Provably secure certificateless two-party authenticated key agreement protocol without pairing, in: 2009 International Conference on Computational Intelligence and Security, 2009, pp. 208–212.
[13] D. Goya, D. Nakamura, R. Terada, Certificateless key agreement protocols under strong models, IEICE T. Fund Electr. E99A (10) (2016) 1822–1832.
[14] H. Guo, Z. Li, Y. Muc, X. Zhang, Provably secure identity-based authenticated key agreement protocols with malicious private key generators, Inf. Sci. 181 (2011) 628–647.
[15] M. Holbl, T. Welzer, B. Brumen, An improved two party identity-based authenticated key agreement protocol using pairings, J. Comput. Syst. Sci. 78 (1) (2012) 142–150.
[16] D. He, Y. Chen, J. Chen, R. Zahng, W. Han, A new two-round certificateless authenticated key agreement protocol without bilinear pairings, Math. Comput. Model. 54 (2011) 3143–3152.
[17] D. He, Y. Chen, J. Hu, A pairing-free certificateless authenticated key agreement protocol, Int. J. Commun. Syst. 25 (2012) 221–230.
[18] D. He, S. Padhye, J. Chen, An efficient certificateless two-party authenticated key agreement protocol, Comput. Math. Appl. 64 (2012) 1914–1926.
[19] D. He, H. Wang, L. Wang, J. Shen, X. Yang, Efficient certificateless anonymous multi-receiver encryption scheme for mobile devices, Soft Comput. 21 (22) (2017) 6801–6810.
[20] M. Hou, Q. Xu, A two-party certificateless authenticated key agreement protocol without pairing, in: 2nd IEEE International Conference on Computer Science and Information Technology, 2009, pp. 412–416.
[21] B. Huang, H. Tu, Strongly secure certificateless one-pass authenticated key agreement scheme, Kuwait J. Sci. 42 (1) (2015) 91–108.
[22] Y. Kim, Y. Choe, O. Hyong, An efficient bilinear pairing-free certificateless two-party authenticated key agreement protocol in the eck model, J. Theor. Phys. Cryptogr. 3 (2013) 1–10.
[23] B. LaMacchia, K. Lauter, A. Mityagin, Stronger security of authenticated key exchange, in: 1st International Conference on Provable Security, LNCS, vol. 4784, 2007, pp. 1–16.
[24] M. Luo, H. Zhao, An authentication and key agreement mechanism for multi-domain wireless networks using certificateless public-key cryptography, Wireless Pers. Commun. 81 (2015) 779–798.
[25] X. Li, Y. Zhang, G.A. Zhang, new certificateless authenticated key agreement protocol for SIP with different KGCs, Secur. Commun. Netw. 6 (2013) 631–643.
[26] H. Lin, Secure certificateless two-party key agreement with short message, Inf. Technol. Control 45 (2016) 71–76.
[27] K. Mahmooda, X. Li, S. Chaudhry, H. Naqvi, S. Kumari, A. Sangaiah, J. Rodrigues, Pairing based anonymous and secure key agreement protocol for smart grid edge computing infrastructure, Future Gener. Comput. Syst. 88 (2018) 491–500.
[28] L. Ni, G. Chen, J. Li, Y. Hao, Strongly secure identity-based authenticated key agreement protocols, Comput. Electr. Eng. 37 (2) (2011) 205–217.
[29] L. Ni, G. Chen, J. Li, Escrowable identity-based authenticated key agreement protocol with strong security, Comput. Math. Appl. 65 (9) (2013) 1339–1349.
[30] L. Ni, G. Chen, J. Li, Y. Hao, Strongly secure identity-based authenticated key agreement protocols in the escrow mode, Sci. China Inform. Sci. 56 (8) (2013) 1–14.
[31] L. Ni, G. Chen, J. Li, Y. Hao, Strongly secure identity-based authenticated key agreement protocols without bilinear pairings, Inf. Sci. 367 (2016) 176–193.
[32] V. Odelu, A. Das, M. Wazid, M. Conti, Provably secure authenticated key agreement scheme for smart grid, IEEE T. Smart Grid 9 (3) (2018) 1900–1910.
[33] J.H. Park, M. Kim, D. Kwon, Security weakness in the smart grid key distribution scheme proposed by Xia and Wang, IEEE T. Smart Grid 4 (3) (2013) 1613–1614.
[34] A. Shamir, Identity-based cryptosystem and signature scheme, Advances in Cryptology-Crypto 1984, LNCS, vol. 196, 1984, pp. 47–53.

[35] H. Sun, Q. Wen, H. Zhang, Z. Jin, A strongly secure pairing-free certificateless authenticated key agreement protocol for low-power devices, Inf. Technol. Control 42 (2013) 113–123.
[36] H. Sun, Q. Wen, H. Zhang, Z. Jin, A novel pairing-free certificateless authenticated key agreement protocol with provable security, Front Comput. Sci. 7 (2013) 544–557.
[37] H. Sun, Q. Wen, H. Zhang, et al, A strongly secure identity based authenticated key agreement protocol without pairings under the GDH assumption, Secur. Commun. Netw. 8 (17) (2015) 3167–3179.
[38] H. Sun, Q. Wen, W. Li, A strongly secure pairing-free certificateless authenticated key agreement protocol under the CDH assumption, Sci. China Inf. Sci. 59 (2016) 032109:1–032109:16.
[39] H. Tu, N. Kumar, J. Kim, J.A. Seo, strongly secure pairing-free certificateless authenticated key agreement protocol suitable for smart media and mobile environments, Multimed. Tools Appl. 74 (2015) 6365–6377.
[40] J. Tsai, N. Lo, Secure anonymous key distribution scheme for smart grid, IEEE T. Smart Grid 7 (2) (2016) 906–914.
[41] F. Wang, Y.A. Zhang, new provably secure authentication and key agreement mechanism for SIP using certificateless public-key cryptography, Comput. Commun. 31 (2008) 2142–2149.
[42] S. Wang, Z. Cao, K. Choo, et al, An improved identitybased key agreement protocol and its security proof, Inf. Sci. 179 (3) (2009) 307–318.
[43] D. Wu, C. Zhou, Fault-tolerant and scalable key management for smart grid, IEEE T. Smart Grid 2 (2) (2011) 375–381.
[44] H. Xiong, Q. Wu, Z. Chen, Toward pairing-free certificateless authenticated key exchanges, in: 14th International Conference on Information Security, LNCS, vol. 7001, 2011, pp. 79–94.
[45] M. Xie, L. Wang, One-round identity-based key exchange with perfect forward security, Theor. Comput. Sci. 112 (14) (2012) 587–591.
[46] J. Xia, Y. Wang, Secure key distribution for the smart grid, IEEE T. Smart Grid 3 (3) (2012) 1437–1443.
[47] G. Yang, C. Tan, Strongly secure certificateless key exchange without pairing, in: 6th ACM Symposium on Information Computer and Communications Security, 2011, pp. 71–79.
[48] L. Ynag, C. Ding, M. Wu, TinyIBAK: design and prototype implementation of an identity-based authenticated key agreement scheme for large scale sensor networks, KSII T. Internet Inf. 7 (11) (2013) 2769–2792.
[49] L. Yan, Y. Chang, S.A. Zhang, lightweight authentication and key agreement scheme for smart grid, Int. J. Distrib. Sens. Netw. 13 (2) (2017) 1–7.
[50] L. Zhang, F. Zhang, Q. Wu, J. Domingo-Ferrer, Simulatable certificateless two party authenticated key agreement protocol, Inf. Sci. 180 (6) (2010) 1020–1030.