sensors

MDPI

Article

Cryptanalysis and Improvement of Several Identity-Based Authenticated and Pairing-Free Key Agreement Protocols for IoT Applications

Haiyan Sun, Chaoyang Li *, Jianwei Zhang, Shujun Liang and Wanwei Huang

College of Software Engineering, Zhengzhou University of Light Industry, Zhengzhou 450001, China; sunhaiyan@zzuli.edu.cn (H.S.)
* Correspondence: lichaoyang2013@163.com

Abstract: Internet of Things (IoT) applications have been increasingly developed. Authenticated key agreement (AKA) plays an essential role in secure communication in IoT applications. Without the PKI certificate and high time-complexity bilinear pairing operations, identity-based AKA (ID-AKA) protocols without pairings are more suitable for protecting the keys in IoT applications. In recent years, many pairing-free ID-AKA protocols have been proposed. Moreover, these protocols have some security flaws or relatively extensive computation and communication efficiency. Focusing on these problems, the security analyses of some recently proposed protocols have been provided first. We then proposed a family of eCK secure ID-AKA protocols without pairings to solve these security problems, which can be applied in IoT applications to guarantee communication security. Meanwhile, the security proofs of these proposed ID-AKA protocols are provided, which show they can hold provable eCK security. Some more efficient instantiations have been provided, which show the efficient performance of these proposed ID-AKA protocols. Moreover, comparisons with similar schemes have shown that these protocols have the least computation and communication efficiency at the same time.

Keywords: AKA; identity-based cryptography; eCK security model; attacks

Citation: Sun, H.; Li, C.; Zhang, J.; Liang, S.; Huang, W. Cryptanalysis and Improvement of Several Identity-Based Authenticated and Pairing-Free Key Agreement Protocols for IoT Applications. *Sensors* **2024**, *24*, 61. https://doi.org/10.3390/s24010061

Academic Editor: Alessandra Rizzardi

Received: 22 November 2023
Revised: 19 December 2023
Accepted: 19 December 2023
Published: 22 December 2023

1. Introduction

Internet of Things (IoT) applications have been increasingly exploited as information technology, operational technology, communication technology, and electronic technology develop. Examples include Smart Grid, Internet of Vehicles, Industrial IoT, and Agriculture IoT, which have enhanced living conditions greatly. With the development of 5G, secure and efficient communication demands have become huge as massive IoT devices are participating in these IoT applications. In general, IoT end devices and IoT servers play important roles in IoT applications. IoT end devices, mounted with sensors, collect useful information and transmit data to the IoT servers over the open network. When these IoT data are transmitted among different IoT devices, the sensitive information inserted in these IoT data needs more secure cryptographic algorithms to protect IoT user privacy and data security.

To protect the security of data transmission, cryptographic means that require secret keys are presented. For IoT user identity authentication, the shared key, public key certificate, and zero-knowledge proof are some common cryptographic methods [1]. For secure IoT data transmission, the digital signature can help confirm data ownership [2], the key agreement can help establish a secure session key [3], and the public key encryption can guarantee IoT data security with ciphertext in the public Internet environment [4]. For IoT data with fine-grained access control, attribute-based encryption can help establish a secure access control strategy with the attributes of IoT users and data [5], and security policy protocol can solve Internet control message protocol (ICMP) attacks [6]. Many

cryptographic algorithms guarantee secure and efficient communication among different IoT applications—there are too many to mention one by one. However, these cryptographic algorithms are public; therefore, the keys are essential for secure data transmission. Moreover, blockchain technology has been applied to IoT applications to solve the centralized management problem [7].

Key agreement (KA) is an important method to protect keys, which supports building a common session key between no less than two different users for subsequent communications. Authenticated key agreement (AKA) not only generates a common session key, but also prevents active attacks and implicitly authenticates the participants simultaneously. In the public-key infrastructure (PKI) setting, the user's long-term public key is matched with the corresponding identity in a certificate, which is derived by a trusted certificate authority. However, managing and transmitting these certificates results in heavy computation and storage costs. Considering IoT end-devices are usually resource-constrained and have limited memory, computation power, storage, and battery life, PKI-based AKA protocols (e.g., [3,8]) are not suitable for IoT applications. Therefore, to avoid the PKI certificate problem, AKA (ID-AKA) protocols with identity are proposed [9]. In ID-AKA protocols, every user owns a unique identity, the long-term public key is constructed by the user's personal identity, and the private key is composed of the identity and a master key, which is created by the trusted key generation center (KGC). Since the introduction of the first ID-AKA protocol in [10], many ID-AKA protocols have been introduced based on bilinear pairings [11–18], which is a high time-complexity operation. Because of the resource restriction of IoT end devices, ID-AKA protocols without pairings are more suitable for IoT applications.

Currently, the modified Bellare and Rogaway (mBR) model [12], the Canetti–Krawczyk (CK) model [19], and the extended Canetti–Krawczyk (eCK) model [20] are some famous security models for AKA protocols. In particular, the eCK model can achieve the most security properties [13]. Meanwhile, to satisfy the communication demands of the high speed, low latency, and large connections in IoT applications, a more secure and efficient AKA protocol is needed. Thus, designing efficient ID-AKA protocols without pairings while maintaining eCK security would be more suitable for IoT applications.

Motivation and Our Contribution

Aiming at considering the aforementioned problems, this paper presents some efficient ID-AKA protocols while holding the eCK security for IoT applications.

- We provide the preliminaries of some Diffie–Hellman assumptions and the forking lemma, provide a detailed description of the eCK-security model for two-party ID-AKA protocols, and draw a figure to show the network model for these protocols.
- We analyze three recently proposed ID-AKA protocols without pairings [21–23], and point out the security flaws against some known attacks.
- We propose a family of pairing-free ID-AKA protocols in the eCK security model. The security proof also considers the case where the public key materials related to the long-term private key can be altered by an active adversary. Furthermore, events in our security proof are complementary.
- We provide some more efficient protocols that need four elliptic curve point multiplication operations. Protocol comparison shows that our efficient protocols have the advantage over similar protocols with the items of security, computation, and communication efficiency.

The paper is organized as follows: Section 2 provides some related work, Section 3 presents the cryptanalysis of several ID-AKA protocols, Section 4 proposes a family of pairing-free ID-AKA protocols, Section 5 shows some more efficient instantiations, Section 6 presents the performance and comparison, and Section 7 gives the conclusion.

2. Related Work

Some related works about the cryptographic methods for IoT applications and developments of KA protocols are provided in the following two subsections.

2.1. Cryptographic Methods for IoT Applications

In IoT applications, the IoT data generally contain plenty of sensitive information about the system users, such as their personal identity, location, and production data. When these IoT data are transmitted among different IoT devices and applications, cryptographic technologies play an essential role in the processes of user identity authentication, IoT data transmission, IoT data fine-grained access control, and so on. Jayabalasamy et al. [2] proposed an aggregate signature scheme to solve the nonrepudiation problem in blockchain ecosystems, which could also confirm the data ownership in the untrusted IoT environment. Li et al. [3] designed a KA protocol based on the SM2 algorithm, which could help establish a secure session key between different system users in smart grid communications. Pu et al. [4] introduced a public key authentication encryption scheme, and added the function of keyword search into this scheme to guarantee IoT data security in industrial IoT applications. Rasori et al. [5] provided a survey of the attributed-based encryption protocols in recent years, and figured out the problems and challenges of IoT data fine-grained access control in most current IoT applications. Onyema et al. [6] presented a security policy protocol for the detection and prevention of Internet control message protocol (ICMP) attacks in software-defined networks. For these different cryptographic algorithms, the key is an essential part of secure IoT communication. Therefore, establishing a secure session key is the first issue that should be taken into consideration for secure communication in IoT applications.

2.2. Developments of KA Protocols

Numerous interesting pairing-free ID-AKA protocols have been introduced in recent years. In 2010, Cao et al. [24] provided a one-round ID-AKA protocol without pairing and presented the security proof in the mBR model. Then, two pairing-free ID-AKA protocols were proposed in [25,26], which utilized the CK model to prove its security. In 2016, Ni et al. [27] showed that the security proof of Sun et al.'s protocol neglected the case where the public key materials related to the long-term private key could be altered by an active adversary, and proposed two eCK-secure ID-AKA protocols without pairings. However, events in the security proof of their protocols were not complementary, which resulted in a mismatch in the freshness definition. Moreover, KGC needed to generate one extra long-term private key, which increased the storage, computation, and communication costs with the increase in the number of users. Bala et al. [21] presented an ID-AKA scheme without pairing for wireless sensor networks and claimed it was secure in the eCK security model. But Dang et al. [28] showed that its security proof had the same problem. Furthermore, the eCK security model was not secure as it suffered from an ephemeral key reveal attack.

In 2018, Dang et al. [28] provided a pairing-free ID-AKA protocol that could achieve eCK security in vehicular ad hoc networks. However, in 2021, Deng et al. [29] showed that it was not eCK-secure and put forward a new scheme that required only four scale multiplication operations. However, we found that the security proof had some flaws, for example, it was inappropriate that the challenger grasped the private key of KGC in the proof Cases CA2, CA3, CA4, and CA6. Mohammadali et al. [22] proposed the NIKE protocol, an ID-AKA protocol without pairing. However, there were still some drawbacks to it. Firstly, we found this scheme had a design flaw, i.e., if an individual was to learn about the long-term private keys of two parties (say $A(Meter)$ and $B(AHE)$), they could easily obtain KGC's master key. Secondly, it could not resist a KCI attack. Thirdly, although the NIKE protocol only needed, at most, three elliptic curve point multiplication operations, it needed three hash-to-point operations, which was a more time-consuming operation than the point multiplication operation. In 2019, Zhang et al. [23] gave two pairing-free and unbalanced ID-AKA protocols for disaster scenarios. Their protocols were actually

unbalanced versions of the protocol in [24]. Their protocols reduced one elliptic curve point multiplication operation for the limited party. However, Zhang et al.'s protocols did not have ephemeral key reveal resistance. In 2020, Daniel et al. [30] pointed out that Bala et al.'s protocol [21] could not resist an ephemeral key reveal attack. They also provided an ID-AKA protocol and presented its security proof in the eCK model [27]. However, its computational cost was still higher, which needed five-point multiplication operations in the elliptic curve. Furthermore, they pointed out that protocol [27] suffered from key offset attacks; however, key offset attacks could be simply avoided by adding a message authentication code using the same method in [30].

In 2021, Kumar and Chand [31] presented an ID-AKA protocol with cloud for the wireless body area network for anonymous health data authentication. However, Rakeei and Moazami [32] pointed out that this protocol could not resist a man-in-the-middle attack and achieve perfect forward secrecy. In 2022, Pu et al. [33] provided a mutual authentication and KA protocol for data privacy preserving in unmanned aerial vehicles (UAVs). Zhang et al. [34] designed a group key agreement (GKA) protocol for user privacy protection and data resource secure sharing in an intelligent IoT system. In 2023, Zhou et al. [35] presented an AGKA protocol for an AI-based automation system, which utilized a semi-trusted authority to perform precomputation operations. Pan et al. [36] focused on the communication security of UAVs to introduce a heterogeneous AKA protocol. Zhang et al. [37] provided a symmetric-key AKA protocol for edge-cloud IIoT, which could achieve perfect forward secrecy based on both authentication and derivation master keys. Abdussami et al. [38] proposed an AKA protocol for secure patient health-related data sharing in IoMT.

The security comparisons of these ID-AKA protocols are shown in Table 1, and the security items of the KGC's master key (MSS), weak perfect forward secrecy (wPFS), key compromise impersonation resilience (KCIR), ephemeral secrets reveal resistance (ESRR), and assumption (AS) were compared. Facing these problems and secure IoT communication demands, a secure ID-AKA protocol is needed to strengthen the security of session keys between different IoT users. The next sections will present the cryptanalysis of several ID-AKA protocols [21–23] first, and then provide the proposed ID-AKA protocols.

Table 1. Security comparisons.

Protocols	Security Model	wPFS	KCIR	ESRR	MSS	AS
CKD [24]	mBR*	Yes	Yes	No	Yes	GDH
FG-I [25]	CK	Yes	Yes	No	Yes	GDH
XW [26]	CK	Yes	Yes	No	Yes	GDH
PF-ID-2PAKA [21]	eCK* (flawed)	Yes	Yes	No	Yes	GDH
DXCLCZF-18 [28]	eCK (flawed)	Yes	No	No	Yes	GDH
ZHWY-19 [23]	mBR* (flawed)	No	No	No	Yes	GDH
NIKE [22]	– (flawed)	Yes	No	Yes	No	–
NCL-16-II [27]	eCK@	Yes	Yes	Yes	Yes	GDH
DRS-20 [30]	eCK	Yes	Yes	Yes	Yes	GDH
DSH-21 [29]	eCK*	Yes	Yes	Yes	Yes	GDH
PWCAS-22 [33]	eCK*	Yes	Yes	Yes	Yes	PUF
ZHVLH-23 [37]	eCK*	Yes	Yes	Yes	Yes	PRF

eCK* and mBR* are without the case where an active adversary may alter all public key materials not only the temporary public key. "–" denotes that there is no formal security proof for the protocol. eCK@ denotes that events in the proof are not complementary.

3. Preliminaries

Some basic concepts including complexity assumptions and the eCK security model for two-party ID-AKA protocols are reviewed in this section.

3.1. Complexity Assumptions

Let $\mathbb{G}$ be an elliptic curve additive group with a large prime order q, and P is a generator of $\mathbb{G}$. Some Diffie–Hellman assumptions over $\mathbb{G}$ are recalled as follows.

- **Computational Diffie–Hellman (CDH) Assumption**: Given three points, P, aP, and bP, where $a, b \in \mathbb{Z}_q^*$, the advantage $Adv_{\mathcal{M}}^{CDH}$ to compute abP is negligible for any probabilistic polynomial time (PPT) adversary $\mathcal{M}$.
- **Decision Diffie–Hellman (DDH) Assumption**: Given four points P, aP, bP, and cP, where $a, b, c \in \mathbb{Z}_q^*$, the advantage $Adv_{\mathcal{M}}^{DDH}$ to decide whether $c \equiv ab \bmod q$ is negligible for any PPT adversary $\mathcal{M}$.
- **Gap Diffie–Hellman (GDH) Assumption**: Given four points P, aP, bP, and cP, where $a, b, c \in \mathbb{Z}_q^*$, the advantage $Adv_{\mathcal{M}}^{GDH}$ to compute abP by accessing a DDH oracle is negligible for any PPT adversary $\mathcal{M}$.

3.2. The Forking Lemma

The forking lemma is applied in the security proof of our proposed protocols. Here, we recall it described in [27].

Let Θ_S be a generic digital signature scheme. Given an input message m, Θ_S produces a triple (K, h, v), where K is a randomly selected value in a large set, h is the hash value of (m, K), and v is only dependent on K, m, and h. Assume that a PPT algorithm $\mathcal{M}$ can produce a valid signature (K, h, v) on the message m with non-negligible probability. Then, with non-negligible probability, a replay of this algorithm can output two valid signatures (K, h, v) and $(K, \overline{h}, \overline{v})$ on the same message m, such that $h \neq \overline{h}$.

3.3. eCK-Security Model for Two-Party ID-AKA Protocols

We now recall the eCK-security model for two-party ID-AKA protocols in [13,27].

- **Participants.** Let $\mathcal{U} = \{ID_1, \cdots, ID_L\}$ be a finite set of L honest parties. Each participant $ID_i \in \mathcal{U}$ is modeled as a PPT Turing machine. Any two parties can be involved in a protocol execution. Each party may execute multiple instances (sessions) in parallel. Let $\prod_{i,j}^m$ denote the mth protocol session, which runs at party ID_i (the owner) with intended partner party ID_j. Every session $\prod_{i,j}^m$ has internal state variables $State_{i,j}^m$ and $tran_{i,j}^m$ to record the state of $\prod_{i,j}^m$, and the transcript of messages sent and received by $\prod_{i,j}^m$, respectively. If $\prod_{i,j}^m$ can compute a session key $SK_{i,j}^m$, $State_{i,j}^m = completed$. The messages in $tran_{i,j}^m$ are ordered according to the protocol specification.
- **Adversary Model.** The adversary $\mathcal{M}$ is modeled as a PPT Turing machine and has full control of the communication network. Active attacks are formulated by allowing the adversary $\mathcal{M}$ to perform the following queries:
  - EphemeralKeyReveal ($\prod_{i,j}^m$). The adversary $\mathcal{M}$ is provided the ephemeral private key of $\prod_{i,j}^m$.
  - SessionKeyReveal ($\prod_{i,j}^m$). The session key held by a completed session $\prod_{i,j}^m$ is returned to $\mathcal{M}$.
  - Corrupt (ID_i). The long-term private key of ID_i is returned to the adversary $\mathcal{M}$.
  - KGCStaticKeyReveal. The adversary $\mathcal{M}$ obtains the master key of KGC. This query is used to model master key forward secrecy.
  - RegCT (ID_i). Via this query, the adversary $\mathcal{M}$ is able to register a dishonest party with identity ID_i. Meanwhile, $\mathcal{M}$ obtains ID_i's long-term private key and totally controls ID_i.
  - Send ($\prod_{i,j}^m, M$). Via this query, the adversary $\mathcal{M}$ can send any message M to party ID_i in session $\prod_{i,j}^m$ on behalf of party ID_j. The adversary $\mathcal{M}$ is responded to according to the protocol specification. $\prod_{i,j}^m$ can be initiated by ID_i when $M = \lambda$. In general, for simplicity $ID_i \neq ID_j$ is required, i.e., two identical participants will not run a session. Internal states of $\prod_{i,j}^m$ should be maintained accordingly.

- **Test ($\prod_{i,j}^{m}$)**. The input session $\prod_{i,j}^{m}$ must be fresh. In response to this query, $\prod_{i,j}^{m}$ flips a fair coin $b \in \{0,1\}$, and returns the real session key if $b = 0$, or a random sample from the distribution of the session key if $b = 1$.

- **Security Experiment.** The security experiment between the adversary $\mathcal{M}$ and the challenger $\mathcal{CH}$ consists of the following phases.
  - **Setup.** The challenger $\mathcal{CH}$ generates the system parameters along with the master private key and valid long-term secret keys for each party. The adversary $\mathcal{M}$ is then provided all public data, including the identities of all the honest parties.
  - **The first phase of the game.** Adversary $\mathcal{M}$ is allowed to issue a polynomial number of EphemeralKeyReveal, SessionKeyReveal, Corrupt, KGCStaticKeyReveal, RegCT, and Send queries in any order.
  - **The second phase of the game.** At some point, adversary $\mathcal{M}$ chooses a fresh session $\prod_{i,j}^{m}$ (see Definition 2) and issues a Test($\prod_{i,j}^{m}$) query at most once. After this, adversary $\mathcal{M}$ can keep asking other queries under the condition that the test session must remain fresh.
  - **The end of the game.** $\mathcal{M}$ makes a guess b' for b.

- **Advantage.** $\mathcal{M}$ wins the above security experiment if the test session $\prod_{i,j}^{m}$ is still fresh and $b' = b$. The advantage of $\mathcal{M}$ in winning the above security experiment is defined as $Adv^{AKE}(\mathcal{M}) = |2Pr[\mathcal{M}\ wins] - 1|$, where $\mathcal{M}$ wins refers to $\mathcal{M}$ can distinguish the tested session key from a random string.

In the following, we introduce definitions for Matching Session, Freshness, and eCK Security.

Definition 1 (Matching Session)**.** *If two completed sessions $\prod_{i,j}^{m}$ and $\prod_{j,i}^{n}$ have the same message transcript, they are said to be matching.*

Definition 2 (Freshness)**.** *Let $\prod_{i,j}^{m}$ be a completed session between honest party ID_i and ID_j, $\prod_{i,j}^{m}$ is said to be fresh if none of the following three conditions hold:*

(1) The adversary $\mathcal{M}$ knows the session key of $\prod_{i,j}^{m}$ or its matching session $\prod_{j,i}^{n}$ (if $\prod_{j,i}^{n}$ exists);

(2) $\prod_{i,j}^{m}$ has a matching session $\prod_{j,i}^{n}$, and the adversary $\mathcal{M}$ knows both the long-term private key of participant ID_i and the ephemeral private key of $\prod_{i,j}^{m}$, or both the long-term private key of participant ID_j and the ephemeral private key of $\prod_{j,i}^{n}$.

(3) $\prod_{i,j}^{m}$ has no matching session, and the adversary $\mathcal{M}$ knows both the long-term private key of participant ID_i and the ephemeral private key of $\prod_{i,j}^{m}$, or the long-term private key of participant ID_j.

Remark 1. *The first condition in Definition 2 is to exclude the trivial attack that $\mathcal{M}$ obtains the session key directly. The second and third conditions in Definition 2 are to exclude the trivial attack that $\mathcal{M}$ obtains the long-term private key and the ephemeral private key of one party simultaneously. If $\prod_{i,j}^{m}$ has no matching session, it means that $\mathcal{M}$ has obtained the ephemeral private key of participant ID_j; therefore, $\mathcal{M}$ cannot obtain the long-term private key of ID_j.*

Definition 3 (eCK Security)**.** *We say that an ID-AKA protocol is secure in the eCK model if the following conditions hold:*

(1) If two honest parties successfully complete matching sessions, they both compute the same session key.

(2) For any PPT adversary, $\mathcal{M}$, $Adv^{AKE}(\mathcal{M})$ is negligible in security parameter k.

Remark 2. *If a protocol is secure under Definition 3, then it achieves implicit mutual key authentication and the basic security properties, including weak perfect forward secrecy (wPFS), key compromise impersonation resilience (KCIR), ephemeral secrets reveal resistance (ESRR), known key*

security, no key control, resistance to basic impersonation attack, replay attack resilience, resistance to man-in-the-middle attack, and unknown key share resilience.

4. Cryptanalysis of Several ID-AKA Protocols

Figure 1 shows the network model for ID-AKA protocols considered in our paper. Here, user A, user B, and trusted authority (acts as KGC) are the three main parties of an ID-AKA protocol. A and B obtain their long-term private keys in the extract phase and use them to reach AKA each other. Next, this section provides the cryptanalysis of several ID-AKA protocols.

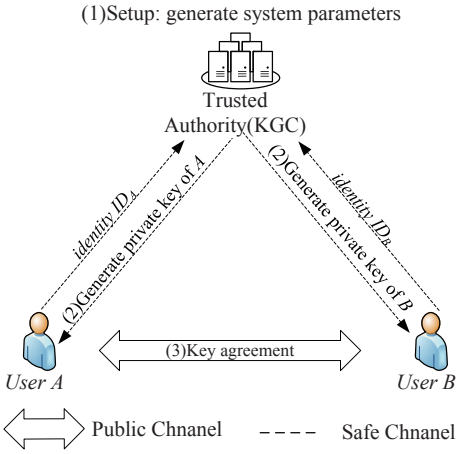

Figure 1. Network model for ID-AKA protocols.

4.1. The PF-ID-2PAKA Protocol

PF-ID-2PAKA [21] is also composed of three stages, i.e., setup, extract, and key agreement. The former two stages are the same as those of the DXCLCZF-18 protocol [28]. Here, we only describe the key agreement stage in Figure 2.

$A(ID_A, R_A, s_A)$		$B(ID_B, R_B, s_B)$
	$M_1 = \{ID_A, R_A, T_A\}$ →	
$e_A \in_R \mathbb{Z}_q^*$		$e_B \in_R \mathbb{Z}_q^*$
$T_A = e_A P$	← $M_2 = \{ID_B, R_B, T_B\}$	$T_B = e_B P$
$K_{AB}^1 = (e_A + s_A)(T_B + H_1(ID_B \Vert R_B)T_B)$		$K_{BA}^1 = (e_B + s_B)(T_A + H_1(ID_A \Vert R_A)T_A)$
$K_{AB}^2 = e_A T_B$		$K_{BA}^2 = e_B T_A$
$sk = H_2(ID_A \Vert ID_B \Vert T_A \Vert T_B \Vert K_{AB}^1 \Vert K_{AB}^2)$		$sk = H_2(ID_A \Vert ID_B \Vert T_A \Vert T_B \Vert K_{BA}^1 \Vert K_{BA}^2)$

Figure 2. The key agreement phrase of the PF-ID-2PAKA protocol [21].

4.1.1. Ephemeral Key Reveal Attack

Suppose that A's ephemeral key e_A is compromised by an adversary. Next, we show that the PF-ID-2PAKA protocol suffers from an ephemeral key reveal attack.

(1) The adversary $\mathcal{M}$ initializes a session $\prod_{A,B}^{\ell}$ through the query Send($\prod_{A,B}^{\ell}, \perp$), and then obtains the message $M_1 = \{ID_A, R_A, T_A\}$, where $T_A = e_A P$ with a random element $e_A \in \mathbb{Z}_q^*$.

(2) Upon receiving the message, M_1, $\mathcal{M}$ randomly picks an ephemeral private key $e_B^{\mathcal{M}} \in \mathbb{Z}_q^*$, calculates $T_B^{\mathcal{M}} = e_B^{\mathcal{M}} P - R_B - H_1(ID_B, R_B)P_{pub}$, and returns $M_2 = \{ID_B, R_B, T_B^{\mathcal{M}}\}$ to $\prod_{A,B}^{\ell}$ impersonating B via the query Send($\prod_{A,B}^{\ell}, M_2$). Note that B's identity ID_B

and correct public key material R_B can be obtained from the response of the query Send($\prod_{B,U}^{t}, \perp$).

(3) Upon the receipt of M_2, A calculates the shared session key and completes this session. Specifically, A calculates $sk = H_2(ID_A||\ ID_B||T_A||T_B^{\mathcal{M}}||K_{AB}^1||K_{AB}^2)$, where $K_{AB}^1 = (s_A + e_A)(R_B + H_1(ID_B, R_B)P_{pub} + T_B^{\mathcal{M}})$ and $K_{AB}^2 = e_A T_B^{\mathcal{M}}$.

(4) Now, $\mathcal{M}$ performs the query EphemeralKeyReveal($\prod_{A,B}^{\ell}$) to reveal an ephemeral private key e_A. With the knowledge of $e_B^{\mathcal{M}}$ and e_A, $\mathcal{M}$ computes $K_{BA}^{1,\mathcal{M}} = e_B^{\mathcal{M}}(R_A + H_1(ID_A, R_A)P_{pub} + T_A)$, $K_{BA}^{2,\mathcal{M}} = e_A T_B^{\mathcal{M}}$, and $sk = H_2(ID_A||\ ID_B||T_A||T_B^{\mathcal{M}}||K_{BA}^{1,\mathcal{M}}||K_{BA}^{2,\mathcal{M}})$.

Correctness. The following provides the correctness of the attack process.
As a result of $T_B^{\mathcal{M}} = e_B^{\mathcal{M}} P - R_B - H_1(ID_B, R_B)P_{pub}$, we can obtain

$$\begin{aligned} K_{AB}^1 &= (s_A + e_A)(R_B + H_1(ID_B, R_B)P_{pub} + T_B^{\mathcal{M}}) \\ &= (s_A + e_A)e_B^{\mathcal{M}} P \\ &= e_B^{\mathcal{M}}(s_A P + e_A P) \\ &= e_B^{\mathcal{M}}(R_A + H_1(ID_A, R_A)P_{pub} + T_A) \\ &= K_{BA}^{1,\mathcal{M}}. \end{aligned}$$

Thus, A and $\mathcal{M}$ receive the same session key, which means that the PF-ID-2PAKA protocol suffers from an ephemeral key reveal attack. Note that our construction cannot suffer from the above attack as both of the two shared secrets not only depend on the ephemeral private key, but also depend on the long-term private key, and they are linearly independent. Therefore, it is impossible to remove the long-term private key from the two shared secrets simultaneously.

4.1.2. Flaws in the Security Proof

The PF-ID-2PAKA protocol can not be proved secure under the hardness of the GDH problem in Case 3 (i.e., $\mathcal{M}$ can neither obtain the long-term private key of ID_A nor that of ID_B). Meanwhile, the ephemeral key t_B of ID_B may be created by $\mathcal{M}$, then $\mathcal{CH}$ cannot know t_B. In ignorance of t_B, $\mathcal{CH}$ does not calculate CDH$(U, V) = K_1 - t_A(T_B + V) - t_B U$. Therefore, the challenger $\mathcal{CH}$ cannot solve the GDH instance.

4.2. The ZHWY-19 Protocol

Here, we only describe the key agreement stage of the ZHWY-19-I protocol [23] in Figure 3. For more details, one can refer to [23]. Note that Cheng et al. [39] pointed out that the ZHWY-19-I protocol [23] cannot achieve forward security and resist the key compromise impersonation attack. Here, we point out that this protocol is weak against the ephemeral key reveal attack and flaws in the security proof.

4.2.1. Ephemeral Key Reveal Attack

If e_A and e_B in a past session have been compromised by the adversary $\mathcal{M}$, $\mathcal{M}$ can compute the session key sk.

(1) $\mathcal{M}$ accesses to $M_1 = \{R_A, V_A, u_A\}$ and $M_2 = \{R_B, V_B, T_B, T_A, mac_B\}$ of the session.

(2) Given $\{ID_A, e_A, R_A\}$ and $\{ID_B, e_B, R_B\}$, $\mathcal{M}$ calculates the keys of $PK_A = R_A + H_1(ID_A||\ R_A)P_{pub}$, $PK_B = R_B + H_1(ID_B||R_B)P_{pub}$, $K_1 = e_A PK_B + e_B PK_A$, $K_2 = e_A e_B P$, and $sk = H_2(ID_A||ID_B||T_A||T_B||K_1||K_2)$ as the shared session key.

Thus, the ZHWY-19-I protocol is weak against the ephemeral secret key leakage. Note that they did not claim their protocol holds ephemeral key reveal resistance.

4.2.2. Flaws in the Security Proof

In the ZHWY-19-I protocol, the answer to oracle Corrupt(ID_i) is improper. Actually, Corrupt(ID_i) should return the long-term private key (s_i, R_i, v_i, V_i) rather than s_i to the adversary.

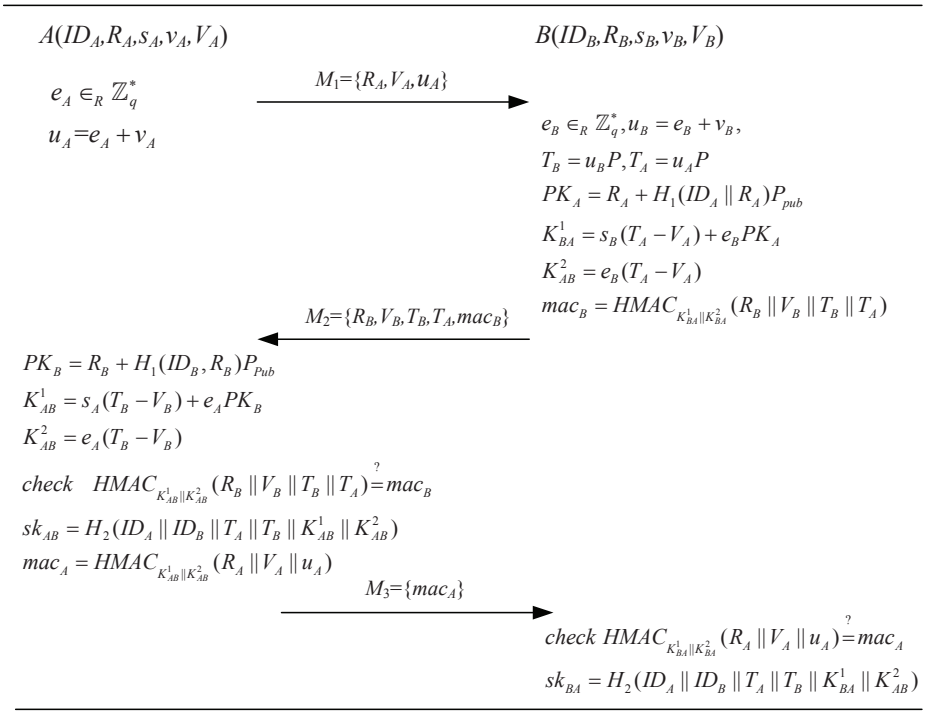

Figure 3. The key agreement phrase of the ZHWY-19-I protocol [23].

4.3. Mohammadali et al.'s Protocols

Mohammadali et al. [22] proposed two protocols, the NIKE protocol and the NIKE$^+$ protocol. These two protocols contain three stages, namely setup, extract, and key agreement. The NIKE protocol is briefly shown in Figure 4. Note that, here, we did not analyze the flaws in the security proof as there was no security proof in [22].

4.3.1. The Insecurity of the KGC's Master Key

If the user $A(Meter)$ and the user $B(AHE)$ launch collusion attacks, they can know the KGC's master key s. As $y_A = H_2(ID_B, Y_B)s$, they can obtain $s = H_2(ID_B, Y_B)^{-1}y_A$ with the knowledge of y_A and Y_B.

4.3.2. Key Compromise Impersonation (KCI) Attack

The NIKE protocol suffers from a key compromise impersonation (KCI) attack, i.e., if the user $B(AHE)$'s long-term private key Y_B is compromised by $\mathcal{M}$, $\mathcal{M}$ can impersonate any user (say $A(Meter)$) with $B(AHE)$'s long-term private key Y_B to communicate with $B(AHE)$. The details are as follows. Note that they did not claim their protocol held KCI resistance.

(1) $\mathcal{M}$ obtains ID_A, R_A by eavesdropping on a connection between $A(Meter)$ and any user. Then, $\mathcal{M}$ picks $e_A^{\mathcal{M}} \in \mathbb{Z}_q^*$ at random, calculates $T_A^{\mathcal{M}} = e_A^{\mathcal{M}}P - H_2(ID_B, Y_B)P_{pub} - R_A$, and sends $\{ID_A, T_A^{\mathcal{M}}, R_A\}$ to $B(AHE)$.
(2) Upon the receipt of $\{ID_A, T_A^{\mathcal{M}}, R_A\}$, $B(AHE)$ generates t_B, T_B, K_{BA}, m_B according to the protocol.
(3) Upon receiving $\{ID_B, T_B, m_B\}$, $\mathcal{M}$ calculates $K_{AB}^{\mathcal{M}} = e_A^{\mathcal{M}}T_B$, $m_A^{\mathcal{M}} = H_1(1, K_{AB}^{\mathcal{M}})$ and $SK = H_1(ID_A || ID_B||, K_{AB}^{\mathcal{M}})$, and finally sends $m_A^{\mathcal{M}}$ to $B(AHE)$.

(4) Upon receiving $m_A^{\mathcal{M}}$, $B(AHE)$ verifies $m_A^{\mathcal{M}}$ is correct and computes the session key according to the protocol.

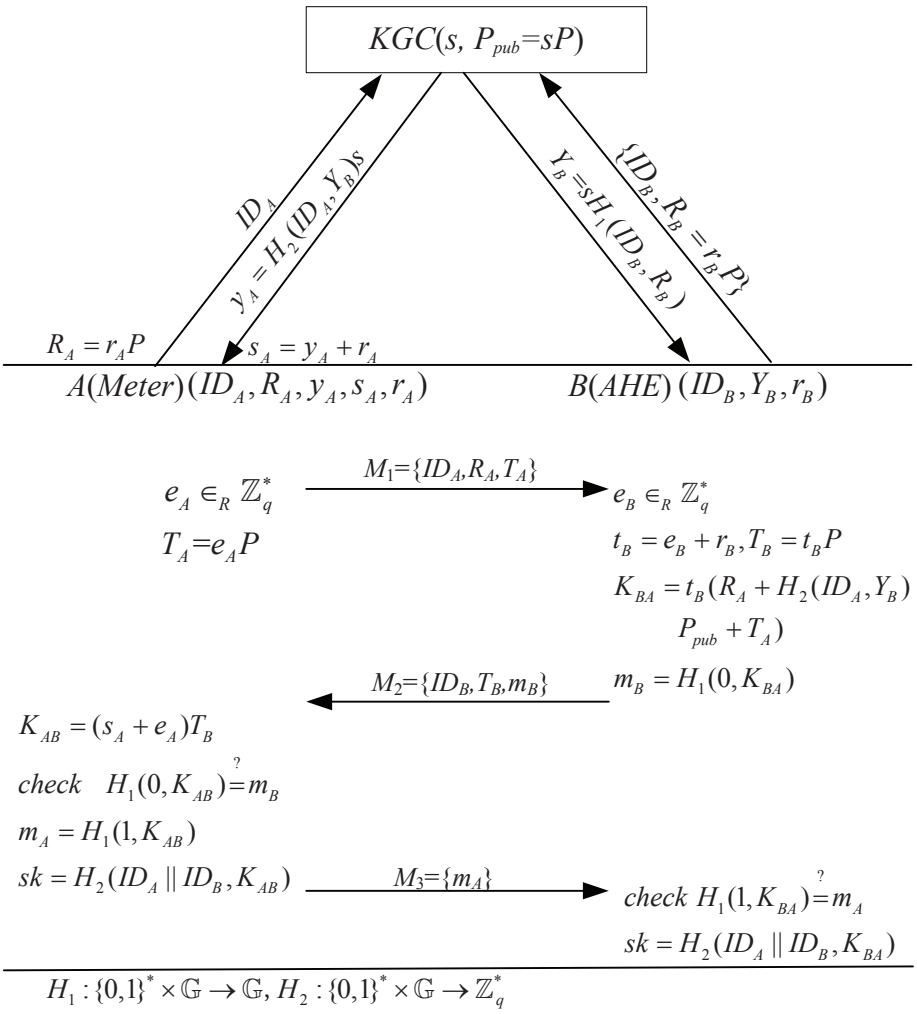

Figure 4. The NIKE protocol [22].

Correctness. The following provides the correctness of the attack process. As $T_A^{\mathcal{M}} = e_A^{\mathcal{M}} P - H_2(ID_B, y_B)P_{pub} - R_A$ and $T_B = t_B P$, we can obtain

$$\begin{aligned} K_{BA} &= t_B(R_A + H_2(ID_A, Y_B)P_{pub} + T_A^{\mathcal{M}}) \\ &= t_B(R_A + H_2(ID_A, Y_B)P_{pub} + e_A^{\mathcal{M}} P \\ &\quad - H_2(ID_B, Y_B)P_{pub} - R_A) \\ &= t_B e_A^{\mathcal{M}} P \\ &= e_A^{\mathcal{M}} T_B \\ &= K_{AB}^{\mathcal{M}}. \end{aligned}$$

Thus, A and $\mathcal{M}$ obtain the same session key, which means that the NIKE protocol suffers from a KCI attack. The KCI attack on the NIKE$^+$ protocol is the same as above.

5. Our General Construction

This section firstly provides the construction $\sum_{C_1,C_2,C_3,C_4}$, secondly show the construction $\sum_{C_1,C_2,C_3,C_4}$ is correct, and thirdly provides the security proof.

5.1. Construction Description

The $\sum_{C_1,C_2,C_3,C_4}$ is composed of three stages, i.e., setup, extract, and key agreement.

- **Setup:** Select security parameter k, KGC performs as follows:
  (1) Pick an elliptic curve $E/\mathbb{F}_p$, where $\mathbb{F}_p$ is a finite filed, and p is a prime number with k bits.
  (2) Create a cyclic additive group $\mathbb{G}$ with the order q, which is generated by a base point P over $E/\mathbb{F}_p$.
  (3) Choose $s \in \mathbb{Z}_q^*$ randomly, and then set the master private key s and the system public key $P_{pub} = sP$.
  (4) Pick $H_1 : \{0,1\}^* \to \mathbb{Z}_q^*$ and $H_2 : \{0,1\}^* \to \{0,1\}^k$.
  (5) Expose $\langle E/\mathbb{F}_p, \mathbb{G}, q, P, P_{pub}, H_1, H_2 \rangle$, and meanwhile retain s unrevealed.
- **Extract:** KGC derives the long-term private key for user $ID_i \in \{0,1\}^*$ as below.
  (1) KGC randomly selects $r_i \in \mathbb{Z}_q^*$, and calculates $R_i = r_iP$ and $h_i = H_1(ID_i||R_i)$.
  (2) KGC computes $s_i = r_i + h_is \bmod q$ and derives (s_i, R_i) as the user's long-term private key.
  (3) KGC sends (s_i, R_i) to the user securely.

  Upon receiving (s_i, R_i), the user can verify $s_iP \stackrel{?}{=} R_i + H_1(ID_i||R_i)P_{pub}$. If this verification succeeds, the key pair (s_i, R_i) is correct and valid. s_iP serves as the real public key in relation to ID_i.
- **Key Agreement:** Assume that user A with identity ID_A hopes to compute a key with user B with identity ID_B.
  (1) A randomly picks an ephemeral secret key $e_A \in \mathbb{Z}_q^*$, calculates $T_A = e_AP$, and returns $M_1 = \{ID_A, R_A, T_A\}$ to B. The agreement process in Figure 5.
  (2) When $\{ID_A, R_A, T_A\}$ is received, B picks an ephemeral secret key $e_B \in \mathbb{Z}_q^*$, calculates $T_B = e_BP$, and returns $\{ID_B, R_B, T_B\}$ to A. Next, B calculates $sk_{BA} = H_2(ID_A||ID_B||R_A||R_B||T_A||T_B||K_{BA}^1||K_{BA}^2)$ as the shared session key, where $K_{BA}^1 = (s_B + C_2e_B)(PK_A + C_1T_A)$, $K_{BA}^2 = (s_B + C_4e_B)(PK_A + C_3T_A)$, $PK_A = R_A + H_1(ID_A||R_A)P_{pub}$, $C_i(i = 1,2,3,4) \in \mathbb{Z}_q^*$ and $C_1 \neq C_3, C_2 \neq C_4$. Finally, B sends $M_2 = \{ID_B, R_B, T_B\}$ to A.
  (3) When $M_2 = \{ID_B, R_B, T_B\}$ is received, A calculates $PK_B = R_B + H_1(ID_B||R_B)P_{pub}$, and two shared secrets $K_{AB}^1 = (s_A + C_1e_A)(PK_B + C_2T_B)$ and $K_{AB}^2 = (s_A + C_3e_A)(PK_B + C_4T_B)$, where $C_i(i = 1,2,3,4) \in \mathbb{Z}_q^*$ and $C_1 \neq C_3, C_2 \neq C_4$. Finally, A calculates the shared session key $sk_{AB} = H_2(ID_A||ID_B||R_A||R_B||T_A||T_B||K_{AB}^1||K_{AB}^2)$.

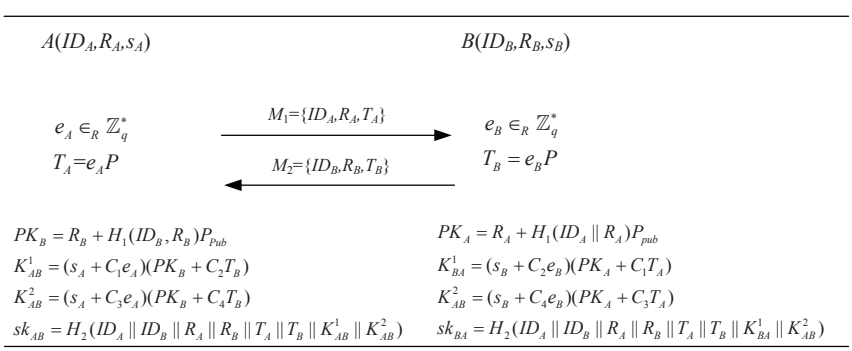

Figure 5. The key agreement phrase of our proposed construction $\sum_{C_1,C_2,C_3,C_4}$.

Note that our construction provides a method to construct eCK secure ID-AKA protocols; however, parameters C_1, C_2, C_3, and C_4 should be fixed in the real execution environment. One can choose a concrete and efficient protocol derived from our construction to

execute in the real environment, e.g., protocol $\sum_{1,1,-1,-1}$, protocol $\sum_{1,-1,-1,1}$ and protocol $\sum_{1,1,2,2}$ described in Section 6.

5.2. Construction Correctness

The following provides the correctness of our construction. As $PK_B = R_B + H_1(ID_B||R_B)$ $P_{pub} = s_B P$, $PK_A = R_A + H_1(ID_A||R_A)P_{pub} = s_A P$, $T_A = e_A P$ and $T_B = e_B P$, we can obtain:

$$\begin{aligned} K_{AB}^1 &= (s_A + C_1 e_A)(PK_B + C_2 T_B) \\ &= (s_A + C_1 e_A)(s_B P + C_2 e_B P) \\ &= (s_A + C_1 e_A)(s_B + C_2 e_B)P \\ &= (s_B + C_2 e_B)(s_A + C_1 e_A)P \\ &= (s_B + C_2 e_B)(PK_A + C_1 T_A) = K_{BA}^1 = K_1; \end{aligned}$$

$$\begin{aligned} K_{AB}^2 &= (s_A + C_3 e_A)(PK_B + C_4 T_B) \\ &= (s_A + C_3 e_A)(s_B P + C_4 e_B P) \\ &= (s_A + C_3 e_A)(s_B + C_4 e_B)P \\ &= (s_B + C_4 e_B)(s_A + C_3 e_A)P \\ &= (s_B + C_4 e_B)(PK_A + C_3 T_A) = K_{BA}^2 = K_2. \end{aligned}$$

Thus both A and B compute $sk_{AB} = sk_{BA} = sk = H_2(ID_A||ID_B||R_A||R_B||T_A||T_B||K_1||\ K_2)$ as their session key. Hence the correctness holds.

5.3. Security Proof

Here, the events in our security proof are complementary, while they are not complementary in [27], and the security proof can be reduced to the following theorems.

Theorem 1. *Provide two random oracles, H_1 and H_2, the proposed ID-AKA protocol is secure in the eCK model based on the GDH assumption over the elliptic curve group.*

Proof. This theorem is under the condition that the two conditions shown in Definition 3 hold. The correctness analysis shows that the first condition stands. The second condition would be proven by contradiction, i.e., there is an adversary who can execute a PPT algorithm to win the game with non-negligible probability, we can use $\mathcal{M}$ to create a GDH solver $\mathcal{CH}$ who can find a solution for the GDH instance. □

Assume $\mathcal{M}$ is a polynomially (in security parameter k) bounded adversary whose advantage is $Adv_{\mathcal{M}}(k)$. Suppose that $\mathcal{M}$ activates no more than $n_p(k)$ different honest parties, and each party can take part in no more than $n_s(k)$ sessions. Suppose that $\mathcal{M}$ chooses $\prod_{a,b}^{n}$, the nth protocol session which executes between party ID_a (the owner) and the target party ID_b (the peer) as the test session. Assume that $\mathcal{M}$ performs, at most, $n_h(k)$ H_2 queries.

According to $Adv^{AKE}(\mathcal{M}) = |2Pr[\mathcal{M}\ wins] - 1|$, we can derive that $Pr[\mathcal{M}\ wins]$ is non-negligible as $Adv^{AKE}(\mathcal{M})$ is non-negligible. As H_2 is modeled as a random oracle, $\mathcal{M}$ can make a clear distinction between a random string and the tested session key in the following three ways:

A1. Guessing attack: $\mathcal{M}$ directly guesses the correct session key.
A2. Key replication attack: $\mathcal{M}$ successfully creates a session that cannot match the test session while holding the same session key. Here, $\mathcal{M}$ can obtain the test session key by querying the non-matching session key.
A3. Forging attack: Sometimes, $\mathcal{M}$ makes H_2 queries on $(ID_a, ID_b, R_a, R_b, T_a, T_b, K_1, K_2)$ in the test session. Here, $\mathcal{M}$ calculates K_1 and K_2 itself.

The guessing of H_2's output is with the negligible probability $\mathcal{O}(1/2^k)$. If two sessions are different, H_2 has the same input by probability $\mathcal{O}(n_s(k)^2/2^k)$, which is also negligible. Then, $\mathcal{M}$ can provide the difference between a random string and the tested session key only by *forging attack*.

Next, a reduction approach is applied to analyze the *forging attack*. This approach reduces the protocol security to the hardness of mathematical problems in the GDH assumption. By making assumptions about the adversary, a challenger can solve a GDH instance with the queried data and forged session key derived by a query-respond game between them. As the GDH instance cannot be solved with the current computation ability in polynomial time, the assumptions about the adversary are invalid, and the proposed ID-AKA protocols are secure.

Now, the detailed descriptions of the reduction proofs are as follows.

If $\mathcal{M}$ can successfully execute *forging attack* with non-negligible probability $Adv^F_{\mathcal{M}}(k)$, we will use $\mathcal{M}$ to create a GDH solver $\mathcal{CH}$ to find a solution for the GDH instance with $Adv^{GDH}_{\mathcal{S}}(k)$. Here, GDH instance is $(U = uP, V = vP)$, and $u, v \in \mathbb{Z}^*_q$, $P \in \mathbb{G}$, $\mathcal{CH}$ plans to calculate GDH$(U, V) = uvP$ performing the DDH oracle. $\mathcal{CH}$ acts as a challenger that performs the eCK game with $\mathcal{M}$ and makes response for $\mathcal{M}$'s queries.

Before the game starts, $\mathcal{CH}$ guesses the test session that $\mathcal{M}$'s choices is $\prod^n_{a,b}$ with a correct probability at least $1/n_p(k)^2 n_s(k)$. Next, $\mathcal{CH}$ needs to guess the strategy that $\mathcal{M}$ adopts. Then, according to Definition 2, test session $\prod^n_{a,b}$ has the matching session $\prod^l_{b,a}$, then $\mathcal{M}$ can only passively forward messages between participant ID_a and participant ID_b, i.e., messages including public key materials and ephemeral keys of $\prod^n_{a,b}$ and $\prod^l_{b,a}$ are selected by $\mathcal{CH}$. Test session $\prod^n_{a,b}$ has no matching session, then $\mathcal{M}$ alters some messages at its own will, i.e., messages including the public key material and the ephemeral key of $\prod^n_{a,b}$ are chosen by $\mathcal{CH}$, and another one of ID_b is chosen by $\mathcal{M}$, and, thus, for ID_b, $\mathcal{CH}$ can only consider the long-term private key of ID_b. With the former analysis and the freeness definition, $\mathcal{CH}$ guesses the operation that $\mathcal{M}$ selects one of the following six complementary choices. Note that, strictly speaking, the ephemeral private key of ID_a refers to $\prod^n_{a,b}$'s ephemeral private key, and the ephemeral private key of ID_b refers to the matching session $\prod^l_{b,a}$'s ephemeral private key.

S1: $\prod^n_{a,b}$ has $\prod^l_{b,a}$, and $\mathcal{M}$ obtains neither the long-term private key of ID_a nor the ephemeral private key of ID_b.
S2: $\prod^n_{a,b}$ has $\prod^l_{b,a}$, and $\mathcal{M}$ cannot obtain any information about the ephemeral private keys of ID_a and ID_b.
S3: $\prod^n_{a,b}$ has $\prod^l_{b,a}$, and $\mathcal{M}$ knows neither the ephemeral private key of ID_a nor the long-term private key of ID_b.
S4: $\prod^n_{a,b}$ has $\prod^l_{b,a}$, and $\mathcal{M}$ cannot obtain any information about the long-term private keys of ID_a and ID_b.
S5: $\prod^n_{a,b}$ does not have a matching session, and $\mathcal{M}$ knows neither the ephemeral private key of ID_a nor the long-term private key of ID_b.
S6: $\prod^n_{a,b}$ does not have a matching session, and $\mathcal{M}$ does not know any information about the long-term private keys of ID_a and ID_b.

One of the former operation successes is if $\mathcal{M}$ succeeds in a forging attack with non-negligible probability. Therefore, the assumption about adversary $\mathcal{M}$ is invalid, and the proposed ID-AKA protocol is secure in the eCK model.

5.3.1. The Analysis of Strategy S1

In this subsection, we analyze strategy S1.

- **Setup:** $\mathcal{CH}$ initializes a list $Setup^{list}$ with entries of $(ID_i, (d_i, R_i), PK_i)$. $\mathcal{CH}$ creates the system parameters and long-term private keys of all parties as follows.
  - $\mathcal{CH}$ picks $P_{pub} \in \mathbb{G}$ at random, and exposes $\langle E/\mathbb{F}_p, \mathbb{G}, q, P, P_{pub}, H_1, H_2 \rangle$. Thus, $\mathcal{CH}$ cannot obtain any information about KGC's master key.

- For ID_a, $\mathcal{CH}$ sets the long-term private key $(\perp, R_a)$, where $h_a \in_R \mathbb{Z}_q^*$, $R_a = U - h_a P_{pub}$. Thus, $PK_a = R_a + h_a P_{pub} = U$.
- For $ID_i (i \neq a)$, $\mathcal{CH}$ sets the long-term private key (s_i, R_i), where $h_i, s_i \in_R \mathbb{Z}_q^*$, $R_i = s_i P - h_i P_{pub}$. Thus, $PK_i = R_i + h_i P_{pub} = s_i P$.
- For every participant, $\mathcal{CH}$ transfers (ID_i, R_i) to $\mathcal{M}$, and stores the tuple $(ID_i, (d_i, R_i), PK_i)$ and (ID_i, R_i, h_i) in $Setup^{list}$ and ${H_1}^{list}$ (described later), respectively.

- **Queries:** $\mathcal{CH}$ maintains four lists, ${H_1}^{list}$, ${H_2}^{list}$, $Send^{list}$, and R^{list}, which are initially empty and used to record H_1, H_2, Send, and SessionKeyReveal oracles, respectively. $\mathcal{CH}$ starts $\mathcal{M}$ by answering $\mathcal{M}$'s queries, as follows.
  - $H_1(ID_i, R_i)$: If an entry (ID_i, R_i, h_i) is recorded in ${H_1}^{list}$, $\mathcal{CH}$ responds with h_i. Then, $\mathcal{CH}$ randomly selects $h_i \in \mathbb{Z}_q^*$, appends (ID_i, R_i, h_i) to ${H_1}^{list}$, and sends h_i back to $\mathcal{M}$.
  - $H_2(ID_i, ID_j, R_i, R_j, T_i, T_j, K_1, K_2)$: List ${H_2}^{list}$ is with $(ID_i, ID_j, R_i, R_j, T_i, T_j, K_1, K_2, h_2)$.
    - If a matching entry $(ID_i, ID_j, R_i, R_j, T_i, T_j, K_1, K_2, h_2)$ is stored in ${H_2}^{list}$, $\mathcal{CH}$ replies with h_2.
    - Else, $\mathcal{CH}$ seeks $(*, ID_i, ID_j, R_i, R_j, T_i, T_j, *)$ in R^{list}. Then, if such an entry exists, $\mathcal{CH}$ sees if K_1 and K_2 are produced correctly by validating $\mathrm{DDH}(PK_i + C_1 T_i, PK_j + C_2 T_j, K_1) \stackrel{?}{=} 1$ and $\mathrm{DDH}(PK_i + C_3 T_i, PK_j + C_4 T_j, K_2) \stackrel{?}{=} 1$, respectively, where $PK_i = R_i + H_1(ID_i, R_i) P_{pub}$ and $PK_j = R_j + H_1(ID_j, R_j) P_{pub}$. If both verifications pass, $\mathcal{CH}$ receives the corresponding $SK_{i,j}^m$ and sets $h_2 \leftarrow SK_{i,j}^m$. Otherwise (at least one verification fails or none), $\mathcal{CH}$ picks $h_2 \in \{0,1\}^k$ at random. Finally, $\mathcal{CH}$ inserts the tuple $(ID_i, ID_j, T_i, T_j, K_1, K_2, h_2)$ into ${H_2}^{list}$ and provides h_2 as the answer.
  - Corrupt(ID_i): If $ID_i = ID_a$, $\mathcal{CH}$ discontinues. Otherwise, $\mathcal{CH}$ responds with s_i.
  - KGCStaticKeyReveal: $\mathcal{CH}$ discontinues.
  - EphemeralKeyReveal($\prod_{i,j}^m$): If $\prod_{i,j}^m = \prod_{b,a}^l$, $\mathcal{CH}$ discontinues. Otherwise, $\mathcal{CH}$ provides the stored ephemeral key $r_{i,j}^m$ as the answer.
  - Send($\prod_{i,j}^m$,M): List $Send^{list}$ is with $(\prod_{i,j}^m, tran_{i,j}^m, r_{i,j}^m, State_{i,j}^m)$, where $tran_{i,j}^m$, $r_{i,j}^m$ and $State_{i,j}^m$ are the transcript by now, the ephemeral secret key, and the state by now, respectively.
    - If M is the second message on the transcript, $\mathcal{CH}$ sets $State_{i,j}^m = completed$ and updates $Send^{list}$.
    - Else $\mathcal{CH}$ executes as follows.
      - If $\prod_{i,j}^m = \prod_{b,a}^l$, $\mathcal{CH}$ sets $r_{i,j}^m = \perp$, gets R_b from $Setup^{list}$ and replies with $\{ID_b, R_b, V\}$.
      - Else $\mathcal{CH}$ randomly chooses $r_{i,j}^m \in \mathbb{Z}_q^*$, obtains R_i from $Setup^{list}$ and replies with $\{ID_i, R_i, r_{i,j}^m P\}$.
      - Finally, $\mathcal{CH}$ updates $Send^{list}$, and updates $State_{i,j}^m$ to $completed$ if the newly generated message is the second message on the transcript.
  - SessionKeyReveal($\prod_{i,j}^m$): List R^{list} is of the form $(\prod_{i,j}^m, ID_{ini}, ID_{resp}, R_{ini}, R_{resp}, T_{ini}, T_{resp}, SK_{i,j}^m)$, where $ini \in \{i,j\}$ and $resp \in \{i,j\}$ denote the initiator and the responder of $\prod_{i,j}^m$, respectively.
    - $\mathcal{CH}$ receives $State_{i,j}^m$ from $Send^{list}$. If $State_{i,j}^m \neq completed$, $\mathcal{CH}$ returns $\perp$.
    - Else if $\prod_{i,j}^m = \prod_{a,b}^n$ or $\prod_{i,j}^m = \prod_{b,a}^l$, $\mathcal{CH}$ aborts.
    - Else if the session key $SK_{i,j}^m$ already exists, $\mathcal{CH}$ responds with $SK_{i,j}^m$.
    - Else $\mathcal{CH}$ obtains $\{ID_{ini}, R_{ini}, T_{ini}\}$ and $\{ID_{resp}, R_{resp}, T_{resp}\}$ from $Send^{list}$, and looks up ${H_2}^{list}$ to see if there is a tuple $(*, ID_{ini}, ID_{resp}, R_{ini}, R_{resp}, T_{ini}, T_{resp}, *)$. Then, if it exists, $\mathcal{CH}$ sees if K_1 and K_2 are produced correctly by validating

$\mathrm{DDH}(PK_{ini} + C_1T_{ini}, PK_{resp} + C_3T_{resp}, K_1) \stackrel{?}{=} 1$ and $\mathrm{DDH}(PK_{ini} + C_3T_{ini}, PK_{resp} + C_4T_{resp}, K_2) \stackrel{?}{=} 1$, respectively, where $PK_{ini} = R_{ini} + H_1(ID_{ini}, R_{ini})P_{pub}$ and $PK_{resp} = R_{resp} + H_1(ID_{resp}, R_{resp})P_{pub}$. If both verifications pass, $\mathcal{CH}$ receives the corresponding h_2 and sets $SK^m_{i,j} \leftarrow h_2$. Otherwise (at least one verification fails or no such a tuple exists), $\mathcal{CH}$ picks $SK^m_{i,j} \in \{0,1\}^k$ at random. Finally, $\mathcal{CH}$ inserts the tuple $(\prod^m_{i,j}, ID_{ini}, ID_{resp}, R_{ini}, R_{resp}, T_{ini}, T_{resp}, SK^m_{i,j})$ into R^{list} and returns $SK^m_{i,j}$.

- Test($\prod^m_{i,j}$): If $\prod^m_{i,j} = \prod^n_{a,b}$, $\mathcal{CH}$ picks $\xi \in \{0,1\}^k$ at random and sends ξ back to $\mathcal{M}$. Otherwise, $\mathcal{CH}$ aborts.

- **Analysis:** If $\mathcal{M}$ can successfully execute a forging attack in Strategy S1 with non-negligible probability, the following conditions should be met.

  (1) $\mathcal{CH}$ continues following the above simulation. If $\mathcal{M}$ chooses Strategy S1, with $\prod^n_{a,b}$ and $\prod^l_{b,a}$ as the test session and its corresponding matching session, respectively, this condition can be met.

  (2) For the test session $\prod^n_{a,b}$, adversary $\mathcal{M}$ must have conducted H_2 queries on the values $\{ID_a, ID_b, R_a, R_b, T_a, V, K_1, K_2\}$, where R_a and R_b are the public key materials of ID_a and ID_b picked by the challenger $\mathcal{CH}$, respectively, T_a and V are the outgoing messages of ID_a and ID_b picked by the challenger $\mathcal{CH}$, respectively, and K_1 and K_2 are correctly formed.

     * If $\prod^n_{a,b}$ is an initiator, the correct input of H_2 should be $(ID_a, ID_b, R_a, R_b, T_a, V, K_1, K_2)$, where $K_1 = (DLOG(U) + C_1r^n_{a,b})(R_b + h_bP_{pub} + C_2V)$, $K_2 = (DLOG(U) + C_3r^n_{a,b})(R_b + h_bP_{pub} + C_4V)$, and $h_b = H_1(ID_b, R_b)$.
     * If $\prod^n_{a,b}$ is a responder, the correct input of H_2 should be $(ID_b, ID_a, R_b, R_a, V, T_a, K_1, K_2)$, where $K_1 = (DLOG(U) + C_2r^n_{a,b})(R_b + h_bP_{pub} + C_1V)$, $K_2 = (DLOG(U) + C_4r^n_{a,b})(R_b + h_bP_{pub} + C_3V)$, and $h_b = H_1(ID_b, R_b)$.

  Finally, $\mathcal{CH}$ receives the item in ${H_2}^{list}$ and outputs $\mathrm{GDH}(U, V) = (C_2 - C_4)^{-1}\Big(K_1 - K_2 + r^n_{a,b}(C_3 - C_1)(R_b + h_bP_{pub}) + r^n_{a,b}(C_3C_4 - C_1C_2)V\Big)$ if $\prod^n_{a,b}$ is an initiator or $\mathrm{GDH}(U, V) = (C_1 - C_3)^{-1}\Big(K_1 - K_2 + r^n_{a,b}(C_4 - C_2)(R_b + h_bP_{pub}) + r^n_{a,b}(C_3C_4 - C_1C_2)V\Big)$ if $\prod^n_{a,b}$ is a responder by the knowledge of $r^n_{a,b}$. Note that since $C_i (i = 1, 2, 3, 4) \in \mathbb{Z}^*_q$ and $C_1 \neq C_3, C_2 \neq C_4$, the solution of $\mathrm{GDH}(U, V)$ is correct. The $\mathcal{CH}$ success rate is at least

$$Adv^{GDH}_{\mathcal{S}}(k) >= \frac{Adv^{F}_{\mathcal{M}}(k)}{6n_h(k)n_p(k)^2n_s(k)^2}.$$

  As $Adv^F_{\mathcal{M}}(k)$ is non-negligible, $Adv^{GDH}_{\mathcal{S}}(k)$ can also be seen as non-negligible. Now, it derives the contradiction of the GDH assumption.

5.3.2. The Analysis of Strategy S2

In this subsection, we analyze strategy S2.

- **Setup:** $Setup^{list}$ is an initially empty list with $(ID_i, (d_i, R_i), PK_i)$. $\mathcal{CH}$ creates the system parameters and all parties' long-term private keys as follows.
  - $\mathcal{CH}$ picks $s \in \mathbb{Z}^*_q$ at random, computes $P_{pub} = sP$, and exposes system parameters $\langle E/\mathbb{F}_p, \mathbb{G}, q, P, P_{pub}, H_1, H_2\rangle$. Thus, $\mathcal{CH}$ cannot obtain any information about KGC's master key.
  - For ID_i, $\mathcal{CH}$ sets the long-term private key (s_i, R_i), where $h_i, s_i \in_R \mathbb{Z}^*_q$, $R_i = s_iP - h_iP_{pub}$. Thus, $PK_i = R_i + h_iP_{pub} = s_iP$.
  - For every participant, $\mathcal{CH}$ transfers (ID_i, R_i) to $\mathcal{M}$, and stores the tuple $(ID_i, (d_i, R_i), PK_i)$ and (ID_i, R_i, h_i) in $Setup^{list}$ and ${H_1}^{list}$ (described later), respectively.

- **Queries:** $\mathcal{CH}$ maintains four lists ${H_1}^{list}$, ${H_2}^{list}$, $Send^{list}$, and R^{list} to store H_1, H_2, Send, and SessionKeyReveal oracles, respectively. $\mathcal{CH}$ performs the queries game with $\mathcal{M}$ as follows:
  - $H_1(ID_i, R_i)$, SessionKeyReveal($\prod_{i,j}^m$), Test($\prod_{i,j}^m$), and $H_2(ID_i, ID_j, R_i, R_j, T_i, T_j, K_1, K_2)$: These four queries are described in the same as those in Strategy S1.
  - Corrupt(ID_i): $\mathcal{CH}$ responds with (s_i, R_i).
  - KGCStaticKeyReveal: $\mathcal{CH}$ responds with s to $\mathcal{M}$.
  - EphemeralKeyReveal($\prod_{i,j}^m$): If $\prod_{i,j}^m = \prod_{a,b}^n$ or $\prod_{i,j}^m = \prod_{b,a}^l$, $\mathcal{CH}$ discontinues. Otherwise, $\mathcal{CH}$ provides the stored ephemeral key $r_{i,j}^m$ as the answer.
  - Send($\prod_{i,j}^m$,M): List $Send^{list}$ has $(\prod_{i,j}^m, tran_{i,j}^m, r_{i,j}^m, State_{i,j}^m)$, where $tran_{i,j}^m$, $r_{i,j}^m$ and $State_{i,j}^m$ are the transcript by now, the ephemeral secret key, and the state by now, respectively.
    - If M is the second message on the transcript, $\mathcal{CH}$ sets $State_{i,j}^m = completed$ and updates $Send^{list}$.
    - Else $\mathcal{CH}$ performs the following steps.
      - If $\prod_{i,j}^m = \prod_{a,b}^n$, $\mathcal{CH}$ sets $r_{i,j}^m = \perp$, receives R_a from $Setup^{list}$, and replies with $\{ID_a, R_a, U\}$.
      - Else If $\prod_{i,j}^m = \prod_{b,a}^l$, $\mathcal{CH}$ sets $r_{i,j}^m = \perp$, receives R_b from $Setup^{list}$, and replies with $\{ID_b, R_b, V\}$.
      - Else $\mathcal{CH}$ randomly chooses $r_{i,j}^m \in \mathbb{Z}_q^*$, obtains R_i from $Setup^{list}$, and replies with $\{ID_i, R_i, r_{i,j}^m P\}$.
      - Finally, $\mathcal{CH}$ updates $Send^{list}$, and updates $State_{i,j}^m$ to $completed$ if the newly generated message is the second message on the transcript.
- **Analysis:** Here, we assume that $\prod_{a,b}^n$ is an initiator here. If $\mathcal{M}$ indeed chooses Strategy S2, $\prod_{a,b}^n$ and $\prod_{b,a}^l$ as the test session and its matching session, respectively, then $\mathcal{CH}$ continues this simulation. If $\mathcal{M}$ successfully executes the forging attack, it must have queried oracle $H_2(ID_a, ID_b, R_a, R_b, U, V, K_1, K_2)$, where R_a, R_b, U, and V are all picked by the challenger $\mathcal{CH}$, $K_1 = (s_a + C_1 DLOG(U))(R_b + h_b P_{pub} + C_2 V)$, $K_2 = (s_a + C_3 DLOG(U))(R_b + h_b P_{pub} + C_4 V)$, and $h_b = H_1(ID_b, R_b)$.
  Finally, $\mathcal{CH}$ receives the item in ${H_2}^{list}$, and outputs $\mathrm{GDH}(U, V) = (C_1 C_3 (C_2 - C_4))^{-1}\Big(C_3 K_1 - C_1 K_2 + s_a(C_1 - C_3)(R_b + h_b P_{pub}) + s_a(C_1 C_4 - C_2 C_3)V\Big)$ by the knowledge of s_a. Note that as $C_i (i = 1, 2, 3, 4) \in \mathbb{Z}_q^*$ and $C_1 \neq C_3, C_2 \neq C_4$, the solution of $\mathrm{GDH}(U, V)$ is correct. $\mathcal{CH}$'s success probability is at least

$$Adv_{\mathcal{S}}^{GDH}(k) >= \frac{Adv_{\mathcal{M}}^{F}(k)}{6 n_h(k) n_p(k)^2 n_s(k)^2}.$$

  As $Adv_{\mathcal{M}}^{F}(k)$ is non-negligible, $Adv_{\mathcal{S}}^{GDH}(k)$ can also be seen as also non-negligible. Now, it derives the contradiction of the GDH assumption.

5.3.3. The Analysis of Strategy S3

Here, we omit the detailed analysis of Strategy S3 as the analysis is almost the same as that for Strategy S1.

5.3.4. The Analysis of Strategy S4

In this subsection, we analyze strategy S4.

- **Setup:** $\mathcal{CH}$ initializes a list $Setup^{list}$ with $(ID_i, (d_i, R_i), PK_i)$. $\mathcal{CH}$ creates the system parameters and all parties' long-term private keys.

- $\mathcal{CH}$ picks $P_{pub} \in \mathbb{G}$ at random, and exposes $\langle E/\mathbb{F}_p, \mathbb{G}, q, P, P_{pub}, H_1, H_2 \rangle$. Thus, $\mathcal{CH}$ does not obtain any information about KGC's master key.
- For ID_a, $\mathcal{CH}$ sets the long-term private key $(\perp, R_a)$, where $h_a \in_R \mathbb{Z}_q^*$, $R_a = U - h_a P_{pub}$. Thus, $PK_a = R_a + h_a P_{pub} = U$.
- For ID_b, $\mathcal{CH}$ sets the long-term private key $(\perp, R_b)$, where $h_b \in_R \mathbb{Z}_q^*$, $R_b = V - h_b P_{pub}$. Thus, $PK_b = R_b + h_b P_{pub} = V$.
- For $ID_i (i \neq a, i \neq b)$, $\mathcal{CH}$ sets the long-term private key (s_i, R_i), where $h_i, s_i \in_R \mathbb{Z}_q^*$, $R_i = s_i P - h_i P_{pub}$. Thus, $PK_i = R_i + h_i P_{pub} = s_i P$.
- For every participant, $\mathcal{CH}$ transfers (ID_i, R_i) to $\mathcal{M}$, and stores $(ID_i, (d_i, R_i), PK_i)$ and (ID_i, R_i, h_i) in $Setup^{list}$ and ${H_1}^{list}$ (described later), respectively.

- **Queries:** $\mathcal{CH}$ maintains four lists ${H_1}^{list}$, ${H_2}^{list}$, $Send^{list}$, and R^{list}, which are initially empty and used for recording H_1, H_2, Send, and SessionKeyReveal oracles, respectively. $\mathcal{CH}$ performs the queries game with $\mathcal{M}$ as follows:
  - $H_1(ID_i, R_i)$, SessionKeyReveal($\prod_{i,j}^m$), Test($\prod_{i,j}^m$), $H_2(ID_i, ID_j, R_i, R_j, T_i, T_j, K_1, K_2)$, and KGCStaticKeyReveal: These five queries are described in the same way as those in Strategy S1.
  - Corrupt(ID_i): If $ID_i = ID_a$ or $ID_i = ID_b$, $\mathcal{CH}$ discontinues. Otherwise, $\mathcal{CH}$ responds with (s_i, R_i).
  - EphemeralKeyReveal($\prod_{i,j}^m$): $\mathcal{CH}$ responses with $r_{i,j}^m$.
  - Send($\prod_{i,j}^m$,M): List $Send^{list}$ is of the form $(\prod_{i,j}^m, tran_{i,j}^m, r_{i,j}^m, State_{i,j}^m)$, where $tran_{i,j}^m$, $r_{i,j}^m$, and $State_{i,j}^m$ are the transcript by now, the ephemeral secret key, and the state by now, respectively.
    - If M is the second message on the transcript, $\mathcal{CH}$ sets $State_{i,j}^m = completed$ and updates $Send^{list}$.
    - Else $\mathcal{CH}$ randomly chooses $r_{i,j}^m \in \mathbb{Z}_q^*$, obtains R_i from $Setup^{list}$, and replies with $\{ID_i, R_i, r_{i,j}^m P\}$. Then, $\mathcal{CH}$ updates $Send^{list}$, and updates $State_{i,j}^m$ to *completed* if the newly generated message is the second message on the transcript.
- **Analysis:** Here, we assume that $\prod_{a,b}^n$ is an initiator. If $\mathcal{M}$ selects Strategy S4, $\prod_{a,b}^n$ and $\prod_{b,a}^l$ as the test session and its matching session, then $\mathcal{CH}$ does not abort in the simulation. If $\mathcal{M}$ successfully performs the forging attack, it must have queried oracle $H_2(ID_a, ID_b, U - h_a P_{pub}, V - h_b P_{pub}, T_a, T_b, K_1, K_2)$, where T_a, T_b, U, and V are all picked by the challenger $\mathcal{CH}$, $K_1 = (DLOG(U) + C_1 r_{a,b}^n)(V + C_2 T_b)$, $K_2 = (DLOG(U) + C_3 r_{a,b}^n)(V + C_4 T_b)$, $h_a = H_1(ID_a, R_a)$, and $h_b = H_1(ID_b, R_b)$. Finally, $\mathcal{CH}$ receives the item in ${H_2}^{list}$, and outputs $\mathrm{GDH}(U, V) = (C_4 - C_2)^{-1}\Big(C_4 K_1 - C_2 K_2 + r_{a,b}^n (C_2 C_3 - C_1 C4) V + r_{a,b}^n C_2 C_4 (C_3 - C_1) T_b\Big)$ by the knowledge of $r_{a,b}^n$. Note that as $C_i (i = 1, 2, 3, 4) \in \mathbb{Z}_q^*$ and $C_1 \neq C_3, C_2 \neq C_4$, the solution of $\mathrm{GDH}(U, V)$ is correct. $\mathcal{CH}$'s success probability is at least

$$Adv_{\mathcal{S}}^{GDH}(k) >= \frac{Adv_{\mathcal{M}}^{F}(k)}{6 n_h(k) n_p(k)^2 n_s(k)^2}.$$

  As $Adv_{\mathcal{M}}^{F}(k)$ is non-negligible, $Adv_{\mathcal{S}}^{GDH}(k)$ can also be seen as non-negligible. Now, it derives the contradiction of the GDH assumption.

5.3.5. The Analysis of Strategy S5

$\prod_{a,b}^n$ has no matching session in strategy S5, thus at least one of ID_b's public key material R_b and ID_b's ephemeral private key is chosen by $\mathcal{M}$. If the adversary selects R_b themselves, then the change in R_b means the change in the ID_b's long-term private key s_b. Hence, a GDH instance cannot be embedded in the long-term private key in strategy S5.

- **Setup:** $Setup^{list}$ is an initially empty list and $(ID_i, (d_i, R_i), PK_i)$ is needed in this phase. $\mathcal{CH}$ creates the system parameters and all parties' long-term private keys.
  - $\mathcal{CH}$ sets V as the system public key P_{pub} and exposes the system parameters $\langle E/\mathbb{F}_p, \mathbb{G}, q, P, P_{pub}, H_1, H_2 \rangle$. Thus $\mathcal{CH}$ cannot know KGC's master key.
  - For ID_i, $\mathcal{CH}$ sets the long-term private key (s_i, R_i), where $h_i, s_i \in_R \mathbb{Z}_q^*$, $R_i = s_i P - h_i P_{pub}$. Thus, $PK_i = R_i + h_i P_{pub} = s_i P$.
  - For every participant, $\mathcal{CH}$ transfers (ID_i, R_i) to $\mathcal{M}$, and stores $(ID_i, (d_i, R_i), PK_i)$ and (ID_i, R_i, h_i) in $Setup^{list}$ and ${H_1}^{list}$ (described later), respectively.
- **Queries:** $\mathcal{CH}$ maintains four lists, ${H_1}^{list}$, ${H_2}^{list}$, $Send^{list}$, and R^{list} to store H_1, H_2, Send, and SessionKeyReveal oracles, respectively. $\mathcal{CH}$ performs the queries game with $\mathcal{M}$ as follows:
  - $H_1(ID_i, R_i)$, KGCStaticKeyReveal, Test($\prod_{i,j}^m$), and $H_2(ID_i, ID_j, R_i, R_j, T_i, T_j, K_1, K_2)$ are described the same as those in Strategy S1.
  - Corrupt(ID_i): If $ID_i = ID_b$, $\mathcal{CH}$ discontinues. Otherwise, $\mathcal{CH}$ responds with (s_i, R_i).
  - EphemeralKeyReveal($\prod_{i,j}^m$): If $\prod_{i,j}^m = \prod_{a,b}^n$, $\mathcal{CH}$ discontinues. Otherwise, $\mathcal{CH}$ provides the stored ephemeral key $r_{i,j}^m$ as the answer.
  - Send($\prod_{i,j}^m$,M): List $Send^{list}$ is with $(\prod_{i,j}^m, tran_{i,j}^m, r_{i,j}^m, State_{i,j}^m)$, where $tran_{i,j}^m$, $r_{i,j}^m$, and $State_{i,j}^m$ are the transcript by now, the ephemeral secret key, and the state by now, respectively.
    - If M is the second message on the transcript, $\mathcal{CH}$ sets $State_{i,j}^m = completed$ and updates $Send^{list}$.
    - Else $\mathcal{CH}$ performs the following steps.
      - If $\prod_{i,j}^m = \prod_{a,b}^n$, $\mathcal{CH}$ sets $r_{i,j}^m = \perp$, receives R_a from $Setup^{list}$, and replies with $\{ID_a, R_a, U\}$.
      - Else $\mathcal{CH}$ randomly chooses $r_{i,j}^m \in \mathbb{Z}_q^*$, obtains R_i from $Setup^{list}$, and replies with $\{ID_i, R_i, r_{i,j}^m P\}$.
      - Finally, $\mathcal{CH}$ updates $Send^{list}$, and updates $State_{i,j}^m$ to $completed$ if the newly generated message is the second message on the transcript.
  - SessionKeyReveal($\prod_{i,j}^m$): This query is the same as that in Strategy S1, except that "Else if $\prod_{i,j}^m = \prod_{a,b}^n$ or $\prod_{i,j}^m = \prod_{b,a}^l$" is modified to "Else if $\prod_{i,j}^m = \prod_{a,b}^n$". This is because the matching session $\prod_{b,a}^l$ certainly exists in Strategy S1, while $\prod_{b,a}^l$ does not exist in Strategy S5.
- **Analysis:** Here, we assume that $\prod_{a,b}^n$ is an initiator. If $\mathcal{M}$ selects Strategy S5 and $\prod_{a,b}^n$ as the test session, then $\mathcal{CH}$ continues using the above simulation. If $\mathcal{M}$ successfully performs the forging attack with non-negligible probability, it should execute the H_2 query on $(ID_a, ID_b, R_a, R_b, U, T_b, K_1, K_2)$, where $K_1 = (s_a + C_1 DLOG(U))(R_b + h_b V + C_2 T_b)$, $K_2 = (s_a + C_3 DLOG(U))(R_b + h_b V + C_4 T_b)$, and $h_b = H_1(ID_b, R_b)$. Note that ID_a's public key material R_a and outgoing message U are both picked by the challenger $\mathcal{CH}$, and at least one of ID_b's public key material R_b and outgoing message T_b is chosen by $\mathcal{M}$.
  By the forking lemma [27], $\mathcal{CH}$ replays $\mathcal{M}$ with the same input and tossing coins. Here, $\mathcal{CH}$ only changes the query results of $H_1(ID_b, R_b)$, i.e., $\mathcal{CH}$ sets $H_1(ID_b, R_b)$ to $\overline{h_b}$, where $\overline{h_b} \in \mathbb{Z}_q^*$ and $\overline{h_b} \neq h_b$. Then, if $\mathcal{M}$ succeeds, it should perform a query on H_2 with $(ID_a, ID_b, R_a, R_b, U, T_b, \overline{K_1}, \overline{K_2})$, where $\overline{K_1} = (s_a + C_1 DLOG(U))(R_b + \overline{h_b} V + C_2 T_b)$, $\overline{K_2} = (s_a + C_3 DLOG(U))\ (R_b + \overline{h_b} V + C_4 T_b)$.
  Finally, $\mathcal{CH}$ receives the item in ${H_2}^{list}$, and outputs $\text{GDH}(U, V) = C_1^{-1}\Big((h_b - \overline{h_b})^{-1}(K_1 - \overline{K_1}) - s_a V\Big)$ using the knowledge of s_a. Note that as $C_1 \in \mathbb{Z}_q^*$, the solution of $\text{GDH}(U, V)$

is correct. Let λ be a factor from the forking lemma for Strategy S5. $\mathcal{CH}$'s success probability is at least

$$Adv_{\mathcal{S}}^{GDH}(k) >= \frac{\lambda Adv_{\mathcal{M}}^{F}(k)}{6n_h(k)^2 n_p(k)^2 n_s(k)}.$$

As $Adv_{\mathcal{M}}^{F}(k)$ is non-negligible, $Adv_{\mathcal{S}}^{GDH}(k)$ can also be seen as non-negligible. Now, it derives the contradiction of the GDH assumption.

5.3.6. The Analysis of Strategy S6

In this subsection, we will analyze strategy S6. A GDH instance cannot be embedded in the long-term private key in Strategy S6.

- **Setup:** $Setup^{list}$ is an initially empty list, and $(ID_i, (d_i, R_i), PK_i)$ is needed in this phase. $\mathcal{CH}$ creates the system parameters and all parties' long-term private keys.
  - $\mathcal{CH}$ sets V as the system public key P_{pub} and exposes $\langle E/\mathbb{F}_p, \mathbb{G}, q, P, P_{pub}, H_1, H_2 \rangle$. Thus, $\mathcal{CH}$ does not obtain any information about KGC's master key.
  - For ID_a, $\mathcal{CH}$ sets the long-term private key $(\perp, R_a)$, where $h_a \in_R \mathbb{Z}_q^*$, $R_a = U - h_a P_{pub}$. Thus, $PK_a = R_a + h_a P_{pub} = U$.
  - For $ID_i (i \neq a)$, $\mathcal{CH}$ sets (s_i, R_i) as the long-term private key, where $h_i, s_i \in_R \mathbb{Z}_q^*$, $R_i = s_i P - h_i P_{pub}$. Thus, $PK_i = R_i + h_i P_{pub} = s_i P$.
  - For every participant, $\mathcal{CH}$ transfers (ID_i, R_i) to $\mathcal{M}$, and stores $(ID_i, (d_i, R_i), PK_i)$ and (ID_i, R_i, h_i) in $Setup^{list}$ and ${H_1}^{list}$ (described later), respectively.
- **Queries:** $\mathcal{CH}$ maintains four lists, ${H_1}^{list}$, ${H_2}^{list}$, $Send^{list}$, and R^{list}, which are initially empty and used for recording H_1, H_2, Send, and SessionKeyReveal oracles, respectively. $\mathcal{CH}$ starts $\mathcal{M}$ by answering $\mathcal{M}$'s queries as follows:
  - The five queries $H_1(ID_i, R_i)$, SessionKeyReveal($\prod_{i,j}^{m}$), Test($\prod_{i,j}^{m}$), KGCStaticKeyRevea, and $H_2(ID_i, ID_j, R_i, R_j, T_i, T_j, K_1, K_2)$ are described in the same way as those in Strategy S5.
  - Corrupt(ID_i), EphemeralKeyReveal($\prod_{i,j}^{m}$) and Send($\prod_{i,j}^{m}$,M): These three queries are described to be the same as those in Strategy S4.
- **Analysis:** Here, we assume that $\prod_{a,b}^{n}$ is an initiator. If $\mathcal{M}$ selects Strategy S6 and $\prod_{a,b}^{n}$ as the test session, then $\mathcal{CH}$ continues this simulation. If $\mathcal{M}$ successfully performs the forging attack, it should execute H_2 query on $(ID_a, ID_b, R_a, R_b, T_a, T_b, K_1, K_2)$, where $K_1 = (DLOG(U) + C_1 r_{a,b}^{n})(R_b + h_b V + C_2 T_b)$,$K_2 = (DLOG(U) + C_3 r_{a,b}^{n})(R_b + h_b V + C_4 T_b)$, and $h_b = H_1(ID_b, R_b)$. Note that ID_a's public key material R_a and outgoing message T_a are both picked by the challenger $\mathcal{CH}$, and at least one of ID_b's public key material R_b and outgoing message T_b is chosen by $\mathcal{M}$.
  By the forking lemma [27], $\mathcal{CH}$ replays $\mathcal{M}$ with the same input and tossing coins. Here, $\mathcal{CH}$ only changes the query results of $H_1(ID_b, R_b)$, i.e., $\mathcal{CH}$ sets $H_1(ID_b, R_b)$ to $\overline{h_b}$, where $\overline{h_b} \in \mathbb{Z}_q^*$ and $\overline{h_b} \neq h_b$. Then, if $\mathcal{M}$ succeeds, it should perform a query on H_2 with $(ID_a, ID_b, R_a, R_b, T_a, T_b, \overline{K_1}, \overline{K_2})$, where $\overline{K_1} = (DLOG(U) + C_1 r_{a,b}^{n})(R_b + \overline{h_b} V + C_2 T_b)$, $\overline{K_2} = (DLOG(U) + C_3 r_{a,b}^{n})\ (R_b + \overline{h_b} V + C_4 T_b)$.
  Here, $\mathcal{CH}$ receives the item in ${H_2}^{list}$, and outputs $\mathrm{GDH}(U, V) = (h_b - \overline{h_b})^{-1}(K_1 - \overline{K_1}) - C_1 r_{a,b}^{n} V$ using the knowledge of $r_{a,b}^{n}$. Note that as $\overline{h_b} \neq h_b$, the solution of $\mathrm{GDH}(U, V)$ is correct. Let λ be a factor from the forking lemma in Strategy S6. $\mathcal{CH}$'s success probability is at least

$$Adv_{\mathcal{S}}^{GDH}(k) >= \frac{\lambda Adv_{\mathcal{M}}^{F}(k)}{6n_h(k)^2 n_p(k)^2 n_s(k)}.$$

As $Adv_{\mathcal{M}}^{F}(k)$ is non-negligible, $Adv_{\mathcal{S}}^{GDH}(k)$ can also be seen as non-negligible. Now, it derives the contradiction of the GDH assumption.

The former formal security proof has proven that the proposed ID-AKA protocol $\sum_{C_1,C_2,C_3,C_4}$ is secure against some comment attacks of guessing attacks, key replication attacks, and forging attacks. Its security can be reduced to the hardness of GDH assumption over the elliptic curve group in the eCK model.

6. More Efficient Instantiations

As $C_i (i = 1, 2, 3, 4) \in \mathbb{Z}_q^*$, our construction needs six scalar multiplications (here, we ignore less time-consuming point additions and general hash function outputs), which is a bit higher than the NCL-16-II protocol [27] at the same security level. However, the NCL-16-II protocol is only a special protocol, while our construction will result in different special protocols with different C_i values, for example, protocol $\sum_{1,1,-1,-1}$ and protocol $\sum_{1,1,H_1(R_A||R_B||T_A||T_B),H_1(R_B||R_A||T_B||T_A)}$. How should the values of C_1, C_2, C_3, and C_4 be chosen in the real execution environment? It would be better to select values that result in more efficient instantiation as different protocols have different computation costs. The following provides some efficient instantiations of our construction.

Protocol 1 ($\sum_{1,1,-1,-1}$). *In this protocol,* $C_1 = C_2 = 1, C_3 = C_4 = -1$. *A computes the shared secrets* $K_{AB}^1 = (s_A + e_A)(PK_B + T_B)$ *and* $K_{AB}^2 = (s_A - e_A)(PK_B - T_B)$. *B compute the shared secrets* $K_{BA}^1 = (s_B + e_B)(PK_A + T_A)$ *and* $K_{BA}^2 = (s_B - e_B)(PK_A - T_A)$. *This protocol reduces two scalar multiplications compared with the general construction.*

Protocol 2 ($\sum_{1,-1,-1,1}$). *In this protocol,* $C_1 = C_4 = 1, C_2 = C_3 = -1$. *A computes the shared secrets* $K_{AB}^1 = (s_A + e_A)(PK_B - T_B)$ *and* $K_{AB}^2 = (s_A - e_A)(PK_B + T_B)$. *B computes the shared secrets* $K_{BA}^1 = (s_B - e_B)(PK_A + T_A)$ *and* $K_{BA}^2 = (s_B + e_B)(PK_A - T_A)$. *This protocol has the same efficiency as* $\sum_{1,1,-1,-1}$.

Protocol 3 ($\sum_{1,1,2,2}$). *In this protocol,* $C_1 = C_2 = 1, C_3 = C_4 = 2$. *A computes the shared secrets* $K_{AB}^1 = (s_A + e_A)(PK_B + T_B)$ *and* $K_{AB}^2 = (s_A + 2e_A)(PK_B + 2T_B)$. *B computes the shared secrets* $K_{BA}^1 = (s_B + e_B)(PK_A + T_A)$ *and* $K_{BA}^2 = (s_B + 2e_B)(PK_A + 2T_A)$. *This protocol only adds a point addition operation compared with the protocol of* $\sum_{1,1,-1,-1}$.

7. Performance and Comparison

This section presents the efficiency and security comparison between our Protocols 1 and 2 with other competitive ID-AKA protocols. Note that only the HC protocols [13] were pairings-based ID-AKA protocols, the other ID-AKA protocols [21–28] and ours were all pairing-free.

7.1. Comparison of Computation Overheads

To evaluate the computational overhead, Table 2 lists the same execution time of different cryptographic operations, reported in [40]. The execution time was calculated using the MIRACL library on a Samsung Galaxy S5 smartphone, equipped with a 2.5 GHz ARM Krait processor with 2GB RAM memory running the Android 4.4.2 operating system.

Table 2. Execution time on a Samsung Galaxy S5.

Notation	Explanation (The Execution Time of)	Time (ms)
T_p	A bilinear pairing e: $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$	32.713
T_{psm}	A pairing-based scalar multiplication in $\mathbb{G}_1$	13.405
T_{ppa}	A pairing-based point addition in $\mathbb{G}_1$	0.056
T_{pexp}	An exponentiation operation in $\mathbb{G}_2$	2.249
T_{mtph}	A hash-to-point in $\mathbb{G}_1$	33.582
T_{esm}	An ECC-based scalar multiplication in $\mathbb{G}$	3.350
T_{epa}	An ECC-based point addition in $\mathbb{G}$	0.014
T_{emtph}	A hash-to-point in $\mathbb{G}$	8.250
T_h	A general hash function	0.006

Next, the total execution times of these two protocols and the competitive ID-AKA protocols [13,21–28] were computed, which are shown in Table 3. In our Protocols 1 and 2, to agree on a session key, each party needed to compute four ECC-based scalar multiplications, three ECC-based point additions, and two general hash function outputs. Therefore, the total computation time at each party was about $4T_{esm} + 3T_{epa} + 2T_h \approx 13.454$ ms. Similarly, the communication costs of protocols in [13,21,24–28] were computed. In protocols ZHWY-19 [23] and NIKE [22], two parties had unbalanced computation costs, i.e., one party had a lower computation cost than the other party. Here, we adopted the lower computation cost of one party. According to Table 2, our Protocols 1 and 2 were nearly 80% of protocols NCL-16-II [27] and CKD [24], 100% of protocols [25] and [21,23,28], 72% of protocol XW [26], and 8% of the HC protocol [13] with relation to the computation cost. That is to say, our Protocols 1 and 2 almost had the lowest computation cost. The comparison results are shown in Figure 6.

Table 3. Comparative computation overheads.

Protocols	Computations	Computation Cost (ms)	Energy Consumption (mj)
PF-ID-2PAKA [21]	$4T_{esm} + 2T_{epa} + 2T_h$	13.44	322.56
DXCLCZF-18 [28]	$4T_{esm} + 2T_{epa} + 2T_h$	13.44	322.56
NIKE [22]	$2T_{esm} + 3T_{emtph}$	31.45	754.8
ZHWY-19 [23]	$4T_{esm} + 3T_{epa} + 3T_h$	13.46	323.04
NCL-16-II [27]	$5T_{esm} + 4T_{epa} + 3T_h$	16.824	403.776
DRS-20 [30]	$5T_{esm} + 3T_{epa} + 5T_h$	16.822	403.728
DSH-21 [29]	$4T_{esm} + 3T_{epa} + 3T_h$	13.460	323.04
PWCAS-22 [33]	$4T_{esm} + 6T_{epa} + 5T_h$	13.514	324.336
ZHVLH-23 [37]	$5T_{esm} + T_{epa} + 16T_h$	16.860	404.64
Our protocols	$4T_{esm} + 3T_{epa} + 2T_h$	13.454	322.896

Energy consumption is one essential item for IoT communication, and the energy consumption of ID-AKA protocol decides the energy efficiency of IoT communication as it is executed by the IoT device. As shown in the former comparative computation overheads in Table 3, the computation costs were calculated. To compute the energy consumption of these key agreement algorithms, the IoT devices equipped with 3.0 V and 8.0 mA were selected. This parameter was set according to the power level of MICA 2 [41]. For the proposed ID-AKA protocol, the energy consumption was $3.0 * 8.0 * 13.454 = 322.896$ mj, and the comparison results with similar protocols are shown in Figure 7. Therefore, the low computation overheads led to low energy consumption, and the proposed ID-AKA protocol had more advantages than similar protocols in relation to the costs of computation and energy.

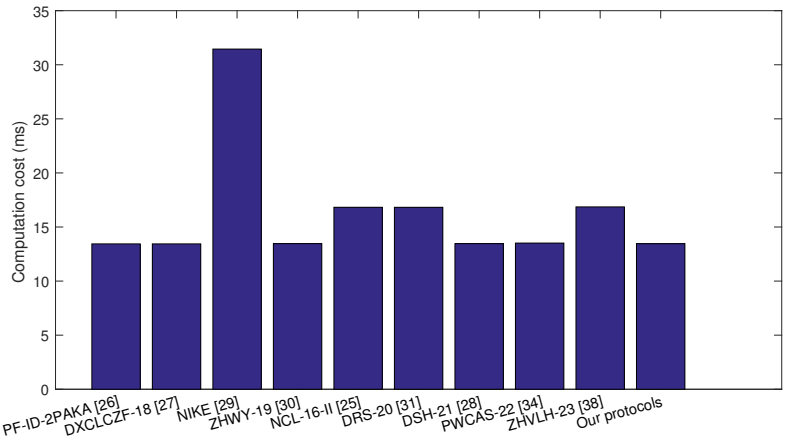

Figure 6. Comparison of computation overheads.

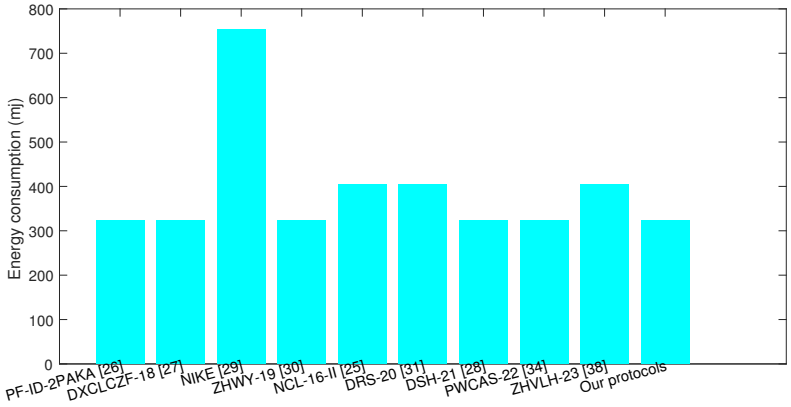

Figure 7. Comparison of energy consumption.

7.2. Comparison of Communication Overheads

Let $|\mathbb{G}_1|$, $|\mathbb{G}_2|$, $|\mathbb{G}|$, and $|\mathbb{Z}_q^*|$ represent elements sizes of $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}$, and $\mathbb{Z}_q^*$, respectively. Furthermore, assume $|ID|$ and $|H|$ represent the length of an identifier and a general hash output, respectively. Considering the Ate pairing and elliptic curves, $|\mathbb{G}_1|$, $|\mathbb{G}_2|$, $|\mathbb{G}|$, $|\mathbb{Z}_q^*|$, and $|H|$ are 1024, 1024, 320, 160, and 160 bits, respectively. We assumed $|ID|$ is 32 bits in length.

Table 4 demonstrates the communication cost comparison of the key agreement phase. Note that in our Protocol 1 (Protocol 2), party A sends $\{ID_A, R_A, T_A\}$ to party B, where $R_A, T_A \in \mathbb{G}$ and ID_A is the identity of A. Party B symmetrically sends $\{ID_B, R_B, T_B\}$ to party A, where $R_B, T_B \in \mathbb{G}$ and ID_B is the identity of B. Therefore, the communication cost of our protocol 1 (protocol 2) is $2 * (|ID|+2|\mathbb{G}|) = 2 * (32 + 2 * 320) = 1344$ bits. The results show that Protocols 1 and 2 have the lowest communication cost.

Table 4. Comparative communication overheads.

Protocols	Messages No.	Communication Cost	Cost (bits)
PF-ID-2PAKA [21]	2	$2\|ID\| + 4\|\mathbb{G}\|$	1344
DXCLCZF-18 [28]	2	$2\|ID\| + 6\|\mathbb{G}\|$	1984
ZHWY-19 [23]	3	$6\|\mathbb{G}\| + \|\mathbb{Z}_q^*\| + 2\|H\|$	2400
NIKE [22]	3	$2\|ID\| + 5\|\mathbb{G}\|$	1664
NCL-16-II [27]	2	$2\|ID\| + 6\|\mathbb{G}\|$	1984
DRS-20 [30]	2	$2\|ID\| + 4\|\mathbb{G}\| + \|H\|$	1504
DSH-21 [29]	2	$2\|ID\| + 4\|\mathbb{G}\|$	1344
PWCAS-22 [33]	2	$2\|ID\| + 4\|\mathbb{G}\| + 2\|H\|$	1664
ZHVLH-23 [37]	2	$3 * (2\|ID\| + \|\mathbb{G}\| + \|H\|)$	1632
Our protocols	2	$2\|ID\| + 4\|\mathbb{G}\|$	1344

7.3. Security Comparisons

As shown in Table 1, some related ID-AKA protocols can capture other security attributes, including known key security, no key control, resistance to basic impersonation attacks, replay attack resilience, resistance to man-in-the-middle attacks, and unknown key share resilience. But, for the proposed IA-AKA protocols, we did not consider explicit mutual authentication, as it can be easily achieved for all one-round protocols [13,21,24–28] by adding a key confirmation. Here, the protocol PWCAS-22 [33] is based on physical unclonable function (PUF), and ZHVLH-23 [37] is based on pseudo-random permutation (PRF). The HC protocol [13], the NCL-16-II protocol [27], the DRS-20 protocol [30], and our protocols are provably secure in the eCK security model. But events in the security proof of NCL-16-II [27] are not complementary, which mismatches the freshness definition. Table 3 shows that our Protocols 1 and 2 can reach the best computation efficiency.

Compared with similar ID-AKA protocols, the proposed Protocols 1 and 2 presented the lowest computation and communication overheads, which could improve IoT com-

munication efficiency in IoT applications. Meanwhile, with the increase in the number of devices, these ID-AKA protocols could also maintain high efficiency, as the key agreement process was executed between two different IoT users. The key agreement between the two parties was less affected by the number of devices in IoT applications and only affected by the hardware, software, and communication protocol in the public internet environment. Although the key agreement times will increase more, this can be ignored with the increasing IoT computation ability.

8. Conclusions

This paper first reviews several ID-AKA protocols without pairings in terms of security and efficiency. We carefully studied them and pointed out the security weaknesses against the ephemeral key reveal attack, key compromise impersonation attack, and launch collusion attack. We also proposed a family of ID-AKA protocols without pairings and proven the security in the eCK security model, a widely accepted security model for AKA protocols. Six strategy analyses were provided, and these ID-AKA protocols were proven to be secure in the eCK model based on the GDH assumption over the elliptic curve group. Then, the instantiations, performance and comparison were presented, and the results show that the proposed ID-AKA protocols were more efficient than other protocols in similar literature. In addition, these ID-AKA protocols no only improved communication security and efficiency in IoT applications, but also saved energy consumption for the communication process.

In the future, with the increasing amount of IoT devices, some security issues of identity authentication, data fine-grained access control, and user privacy protection should still be taken into consideration. Especially with the development of quantum computers and quantum computation, the anti-quantum attack security ID-AKA protocol will be a hot research direction. Meanwhile, many customized ID-AKA schemes should be designed to meet the special requirements of future IoT applications.

Author Contributions: Methodology, H.S. and C.L.; Validation, W.H.; Formal analysis, J.Z.; Investigation, S.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Natural Science Foundation of China (62072416), the Key Research and Development Special Project of Henan Province (221111210500), the Science and Technology Program of Henan Province (212102210107, 232102210125), the Doctor Scientific Research Fund of Zhengzhou University of Light Industry (2021BSJJ033) and the Foundation of State Key Laboratory of Public Big Data (No. PBD2023-25).

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Khan, M.A.; Din, I.U.; Majali, T.E.; Kim, B.S. A survey of authentication in Internet of things-enabled healthcare systems. *Sensors* **2022**, *22*, 9089. [CrossRef]
2. Jayabalasamy, G.; Koppu, S. High-performance Edwards curve aggregate signature (HECAS) for nonrepudiation in IoT-based applications built on the blockchain ecosystem. *J. King Saud Univ.-Comput. Inf. Sci.* **2022**, *34*, 9677–9687. [CrossRef]
3. Li, W.; Li, R.; Wu, K.; Cheng, R.; Su, L.; Cui, W. Design and implementation of an SM2-based security authentication scheme with the key agreement for smart grid communications. *IEEE Access* **2018**,*6*, 71194–71207. [CrossRef]
4. Pu, L.; Lin, C.; Chen, B.; He, D. User-friendly public-key authenticated encryption with keyword search for industrial Internet of things. *IEEE Internet Things J.* **2023**, *10* 13544–13555. [CrossRef]
5. Rasori, M.; La Manna, M.; Perazzo, P.; Dini, G. A survey on attribute-based encryption schemes suitable for the Internet of things. *IEEE Internet Things J.* **2022**, *9*, 8269–8290. [CrossRef]
6. Onyema, E.M.; Kumar, M.A.; Balasubaramanian, S.; Bharany, S.; Rehman, A.U.; Eldin, E.T.; Shafiq, M. A security policy protocol for detection and prevention of internet control message protocol attacks in software defined networks. *Sustainability* **2022**, *14*, 11950. [CrossRef]
7. Alam, S.; Shuaib, M.; Ahmad, S.; Jayakody, D.N.K.; Muthanna, A.; Bharany, S.; Elgendy, I.A. Blockchain-based solutions supporting reliable healthcare for fog computing and Internet of medical things (IoMT) integration. *Sustainability* **2022**, *14*, 15312. [CrossRef]

8. Sun, F.; He, S.; Zhang, X.; Zhang, J.; Li, Q.; He, Y. A fully authenticated Diffie-Hellman protocol and its application in WSNs. *IEEE Trans. Inf. Forensics Secur.* **2022**, *17*, 1986–1999. [CrossRef]
9. Shamir, A. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology: Proceedings of CRYPTO 84 4*; Springer: Berlin/Heidelberg, Germany, 1985; pp. 47–53.
10. Smart, N.P. Identity-based authenticated key agreement protocol based on Weil pairing. *Electron. Lett.* **2002**, *38*, 630–632. [CrossRef]
11. Wang, S.; Cao, Z.; Choo, K.K.R.; Wang, L. An improved identity-based key agreement protocol and its security proof. *Inf. Sci.* **2009**, *179*, 307–318. [CrossRef]
12. Chen, L.; Cheng, Z.; Smart, N.P. Identity-based key agreement protocols from pairings. *Int. J. Inf. Secur.* **2007**, *6*, 213–241. [CrossRef]
13. Huang, H.; Cao, Z. An ID-based authenticated key exchange protocol based on bilinear Diffie-Hellman problem. In Proceedings of the 4th International Symposium on Information, Computer, and Communications Security, Sydney, Australia, 10–12 March 2009; pp. 333–342.
14. Choo, K.K.R.; Nam, J.; Won, D. A mechanical approach to derive identity-based protocols from Diffie-Hellman-based protocols. *Inf. Sci.* **2014**, *281*, 182–200. [CrossRef]
15. Wu, L.; Wang, J.; Choo, K.K.R.; Li, Y.; He, D. An efficient provably-secure identity-based authentication scheme using bilinear pairings for Ad hoc network. *J. Inf. Secur. Appl.* **2017**, *37*, 112–121. [CrossRef]
16. Odelu, V.; Das, A.K.; Wazid, M.; Conti, M. Provably secure authenticated key agreement scheme for smart grid. *IEEE Trans. Smart Grid* **2016**, *9*, 1900–1910. [CrossRef]
17. Gupta, D.S.; Islam, S.H.; Obaidat, M.S.; Vijayakumar, P.; Kumar, N.; Park, Y. A provably secure and lightweight identity-based two-party authenticated key agreement protocol for IIoT environments. *IEEE Syst. J.* **2020**, *15*, 1732–1741. [CrossRef]
18. Lian, H.; Pan, T.; Wang, H.; Zhao, Y. Identity-Based Identity-Concealed Authenticated Key Exchange. In *Computer Security-ESORICS 2021: 26th European Symposium on Research in Computer Security, Darmstadt, Germany, 4–8 October 2021*; Proceedings, Part II 26; Springer International Publishing: Berlin/Heidelberg, Germany, 2021; pp. 651–675.
19. Canetti, R.; Krawczyk, H. Analysis of key-exchange protocols and their use for building secure channels. In Proceedings of the International conference on the theory and applications of cryptographic techniques, Innsbruck, Austria, 6–10 May 2001; Springer: Berlin/Heidelberg, Germany, 2001; pp. 453–474.
20. LaMacchia, B.; Lauter, K.; Mityagin, A. Stronger security of authenticated key exchange. In Proceedings of the International Conference on Provable Security, Wollongong, Australia, 1–2 November 2007; Springer: Berlin/Heidelberg, Germany, 2007; pp. 1–16.
21. Bala, S.; Sharma, G.; Verma, A.K. PF-ID-2PAKA: Pairing free identity-based two-party authenticated key agreement protocol for wireless sensor networks. *Wirel. Pers. Commun.* **2016**, *87*, 995–1012. [CrossRef]
22. Mohammadali, A.; Haghighi, M.S.; Tadayon, M.H.; Mohammadi-Nodooshan, A. A novel identity-based key establishment method for advanced metering infrastructure in smart grid. *IEEE Trans. Smart Grid* **2016**, *9*, 2834–2842. [CrossRef]
23. Zhang, J.; Huang, X.; Wang, W.; Yue, Y. Unbalancing pairing-free identity-based authenticated key exchange protocols for disaster scenarios. *IEEE Internet Things J.* **2018**, *6*, 878–890. [CrossRef]
24. Cao, X.; Kou, W.; Du, X. A pairing-free identity-based authenticated key agreement protocol with minimal message exchanges. *Inf. Sci.* **2010**, *180*, 2895–2903. [CrossRef]
25. Fiore, D.; Gennaro, R. Making the Diffie-Hellman protocol identity-based. In *Topics in Cryptology-CT-RSA 2010: The Cryptographers' Track at the RSA Conference 2010, San Francisco, CA, USA, 1–5 March 2010*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 165–178.
26. Xie, M.; Wang, L. One-round identity-based key exchange with perfect forward security. *Inf. Process. Lett.* **2012**, *112*, 587–591. [CrossRef]
27. Ni, L.; Chen, G.; Li, J.; Hao, Y. Strongly secure identity-based authenticated key agreement protocols without bilinear pairings. *Inf. Sci.* **2016**, *367*, 176–193. [CrossRef]
28. Dang, L.; Xu, J.; Cao, X.; Li, H.; Chen, J.; Zhang, Y.; Fu, X. Efficient identity-based authenticated key agreement protocol with provable security for vehicular ad hoc networks. *Int. J. Distrib. Sens. Netw.* **2018**, *14*, 1550147718772545. [CrossRef]
29. Deng, L.; Shao, J.; Hu, Z. Identity based two-party authenticated key agreement scheme for vehicular ad hoc networks. *Peer-to-Peer Netw. Appl.* **2021**, *14*, 2236–2247. [CrossRef]
30. Daniel, R.M.; Rajsingh, E.B.; Silas, S. An efficient ECK secure identity based two party authenticated key agreement scheme with security against active adversaries. *Inf. Comput.* **2020**, *275*, 104630. [CrossRef]
31. Kumar, M.; Chand, S. A lightweight cloud-assisted identity-based anonymous authentication and key agreement protocol for secure wireless body area network. *IEEE Syst. J.* **2020**, *15*, 2779–2786. [CrossRef]
32. Rakeei, M.A.; Moazami, F. Cryptanalysis of an anonymous authentication and key agreement protocol for secure wireless body area network. *Cryptol. ePrint Arch.* **2020**, 1–4.
33. Pu, C.; Wall, A.; Choo, K.K.R.; Ahmed, I.; Lim, S. A lightweight and privacy-preserving mutual authentication and key agreement protocol for Internet of Drones environment. *IEEE Internet Things J.* **2022**, *9*, 9918–9933. [CrossRef]
34. Zhang, Q.; Zhu, L.; Li, Y.; Ma, Z.; Yuan, J.; Zheng, J.; Ai, S. A group key agreement protocol for intelligent internet of things system. *Int. J. Intell. Syst.* **2022**, *37*, 699–722. [CrossRef]

35. Zhou, T.; Wang, C.; Zheng, W.; Tan, H. Secure and efficient authenticated group key agreement protocol for AI-based automation systems. *ISA Trans.* **2023**, *141*, 1–9. [CrossRef]
36. Pan, X.; Jin, Y.; Li, F. An efficient heterogeneous authenticated key agreement scheme for unmanned aerial vehicles. *J. Syst. Archit.* **2023**, *136*, 102821. [CrossRef]
37. Zhang, Y.; He, D.; Vijayakumar, P.; Luo, M.; Huang, X. SAPFS: An Efficient Symmetric-Key Authentication Key Agreement Scheme with Perfect Forward Secrecy for Industrial Internet of Things. *IEEE Internet Things J.* **2023**, *10*, 9716–9726. [CrossRef]
38. Abdussami, M.; Amin, R.; Vollala, S. Provably secured lightweight authenticated key agreement protocol for modern health industry. *Ad Hoc Netw.* **2023**, *141*, 103094. [CrossRef]
39. Cheng, Q.; Li, Y.; Jiang, Q.; Li, X. Security Analysis of Two Unbalancing Pairing-free Identity-based Authenticated Key Exchange Protocols. *Int. J. Netw. Secur.* **2020**, *22*, 597–601.
40. He, D.; Wang, H.; Khan, M.K.; Wang, L. Lightweight anonymous key distribution scheme for smart grid using elliptic curve cryptography. *IET Commun.* **2016**, *10*, 1795–1802. [CrossRef]
41. Gura, N.; Patel, A.; Wander, A.; Eberle, H.; Shantz, S.C. Comparing elliptic curve cryptography and RSA on 8-bit CPUs. In *Cryptographic Hardware and Embedded Systems-CHES 2004: 6th International Workshop Cambridge, MA, USA, 11–13 August 2004*; Proceedings 6; Springer: Berlin/Heidelberg, Germany, 2004; pp. 119–132.