symmetry

MDPI

Article

Blockchain-Based Batch Authentication and Symmetric Group Key Agreement in MEC Environments

Yun Deng [1,2,*], Jing Zhang [1,2], Jin Liu [3] and Jinyong Li [4]

[1] School of Computer Science and Engineering, Guilin University of Technology, Guilin 541006, China; zhangjing123@glut.edu.cn
[2] Guangxi Key Laboratory of Embedded Technology and Intelligent Systems, Guilin 541006, China
[3] School of Basic Education, Loudi Preschool Education College, Loudi 417500, China; 13873856040@163.com
[4] College of Information Engineering, Nanning University, Nanning 530200, China
* Correspondence: 2002078@glut.edu.cn

Abstract

To address the high computational and communication overheads and the limited edge security found in many existing batch verification methods for Mobile Edge Computing (MEC), this paper presents a blockchain-based batch authentication and symmetric group key agreement protocol. A core feature of this protocol is the establishment of a shared symmetric key among all authenticated participants. This symmetry in key distribution is fundamental for enabling secure and efficient broadcast or multicast communication within the MEC group. The protocol introduces a chameleon hash function built on elliptic curves, allowing smart mobile devices (SMDs) to generate lightweight signatures. The edge server (ES) then performs efficient large-scale batch authentication using an aggregate signature technique. Considering the need for secure and independent communication between SMDs and ES, the protocol further establishes a one-to-one session key agreement mechanism and uses a Merkle tree to verify session key correctness. Formal verification with ProVerif2.05 tool confirms the protocol's security and multiple protection properties. Experimental results show that, compared with the CPPBA, ECCAS, and LBVP schemes, the protocol improves computational efficiency of batch authentication by 0.94%, 67.20%, and 49.53%, respectively. For group key agreement, the protocol achieves a 35.26% improvement in computational efficiency over existing schemes.

Keywords: mobile edge computing; blockchain; elliptic curve cryptography; chameleon hash function

check for updates

Academic Editor: Tamas Kiss

Received: 2 November 2025
Revised: 10 December 2025
Accepted: 11 December 2025
Published: 15 December 2025

Citation: Deng, Y.; Zhang, J.; Liu, J.; Li, J. Blockchain-Based Batch Authentication and Symmetric Group Key Agreement in MEC Environments. *Symmetry* **2025**, *17*, 2160. https://doi.org/10.3390/sym17122160

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

1.1. Background and Motivation

With the dramatic increase in the amount of data, Mobile Edge Computing (MEC) has emerged as a key technology for processing large-scale data by offloading computational tasks to the network edge, thereby significantly reducing data transmission latency [1]. However, the openness and decentralization of MEC introduce new challenges. The inherent asymmetry in computational power between resource-constrained mobile devices and powerful edge servers creates security vulnerabilities, particularly in scenarios involving large-scale concurrent device authentication [2]. Moreover, the issue of semi-trusted edge servers in MEC architectures has attracted increasing attention, as their partially trustworthy nature may lead to data leakage and malicious attacks, thereby posing threats to the overall system security [3]. To mitigate these challenges, blockchain technology [4], with

its inherent characteristics of decentralization, immutability, and traceability, has shown strong potential in enhancing data security. Its distributed ledger helps ensure transparency and consistency in data processing, effectively reducing the risk of data tampering [5]. By introducing functions such as smart contracts, blockchain further enhances the security of automated transactions, making it an effective tool for preventing data leakage and ensuring transaction integrity [6].

This hybrid security architecture, which combines decentralized trust based on blockchain with efficient edge processing, has been actively explored in modern industrial scenarios, such as the Industrial Internet of Things. For instance, addressing the uncertainty of attack characteristics in Internet of Things (IoT) networks, Ref. [7] proposed an intelligent framework that combines fuzzy logic with blockchain and improves the reliability of detecting complex threats. In cloud-fog-edge collaborative environments, Ref. [8] designed a collaborative intrusion detection framework based on Deep Blockchain, achieving accurate identification of distributed attacks while preserving data privacy. In addition, facing the increasingly frequent Denial-of-Service attacks, Ref. [9] constructed a blockchain-based distributed defense system, utilizing smart contracts to achieve automated mitigation of Distributed Denial of Service (DDoS) attacks. These examples collectively demonstrate that the integration of blockchain and MEC architectures is important for enhancing data security, improving system transparency, and promoting decentralized management [10]. However, these efforts mainly focus on post-threat detection and defense mechanisms. As the first line of defense for system security, "access authentication" for massive terminals faces significant challenges in both efficiency and trust; yet decentralized architecture designs targeting this specific phase remain relatively underdeveloped. Therefore, designing an efficient authentication protocol that balances blockchain-based trust with the low-latency characteristics of the edge is a necessary component to complete the overall security framework.

Meanwhile, another critical challenge in MEC environments lies in how to efficiently and securely authenticate a large number of terminal devices [11]. With the rapid increase in connected devices and the growing complexity of application scenarios [12], traditional single-device authentication mechanisms can no longer meet the requirements for low latency and high reliability. Therefore, batch identity authentication has become a key technical solution [13]. It enables simultaneous verification of multiple devices, reducing server-side computation and network latency, and improving overall system performance and security. Researchers have conducted in-depth studies on this approach in different application scenarios. For example, Ref. [14] addressed the conflict between large-scale data authentication and limited terminal resources in Industrial Internet of Things (IIoT) by designing a lightweight signature algorithm based on a hash chain. In addition, the chameleon hash function [15] has been widely used in batch authentication scenarios due to its collision controllability, which provides enhanced security and efficiency [16]. By integrating aggregate signature (AS) structures, the chameleon hash function can flexibly adapt to dynamic authentication demands and substantially improve the authentication efficiency of large-scale device requests [17], thereby serving as an important approach to enhance both the security and performance of MEC batch authentication systems. Recent studies, such as Ref. [18] and Ref. [19], further validated the advantages of this technology by integrating it with editable blockchains and Physical Unclonable Functions (PUFs), respectively, to balance security and efficiency.

1.2. Related Work

Batch verification and symmetric group key agreement are two critical procedures for ensuring the security of multi-party communications [20]. In recent years, scholars

worldwide have proposed various innovative schemes aimed at improving authentication efficiency, optimizing the use of chameleon hash function applications, enhancing privacy protection, and integrating blockchain architectures.

To address the computational bottlenecks of resource-constrained devices, a mainstream line of work focuses on the lightweight design of cryptographic protocols. Ref. [21] and Ref. [22], respectively, incorporated the Chinese Remainder Theorem and chaotic mapping into the design of efficient intra-group authentication and key agreement protocols. To mitigate high concurrency latency, Ref. [23] employed a distributed collaborative authentication approach to more effectively distribute verification tasks, thereby significantly reducing message-verification delays. To improve efficiency in decentralized key generation, Ref. [24] proposed adaptively secure BLS threshold signature schemes, optimizing both the security and efficiency of key generation. Furthermore, to reduce the high computational overhead caused by bilinear pairing operations, Ref. [25] proposed a pairing-free certificateless aggregate signcryption scheme, which significantly improves communication efficiency in Internet of Medical Things environments. Similar lightweight dynamic key-management schemes have also been applied to unmanned aerial vehicle (UAV) swarms and space information networks [26,27].

The chameleon hash function plays a vital role in constructing redactable and efficient authentication protocols due to its trapdoor property. Beyond traditional applications, the optimization of this technology within blockchain environments is gaining increasing attention. Ref. [18] combined chameleon hashing with aggregate signatures to construct an efficient data transmission protocol. In recent research, Ref. [28] proposed a redactable consortium blockchain scheme based on chameleon hashing and multi-authority attribute-based encryption. This approach further unleashes the potential of chameleon hashing in data correction and privacy management through fine-grained access control, providing new insights for addressing trusted data modification in MEC.

In semi-trusted MEC environments, protecting user identity privacy is a core requirement for authentication protocols. Ref. [29] introduced identity-based anonymous authentication techniques, while Refs. [30,31] utilized zero-knowledge proofs to enhance privacy and post-quantum resistance. Addressing trust and efficiency issues in Internet of Vehicles communications, Ref. [32] designed a blockchain-based lightweight certificateless authenticated key agreement protocol, effectively reducing certificate management overhead while ensuring communication privacy. Additionally, Refs. [33,34] achieved conditional privacy-preserving batch authentication through elliptic-curve cryptography (ECC).

To resolve the single point of failure and lack of trust in centralized architectures, integrating blockchain into authentication protocols has become a research hotspot. For instance, Ref. [35] leveraged blockchain to implement decentralized dynamic node key management, where shared keys were constructed through polynomial operations to resist node mobility. Ref. [36] designed a blockchain-based fast handover authentication scheme for mobile environments, introducing a token mechanism to reduce authentication overhead. Ref. [37] combined blockchain with ECC to propose a lightweight yet resilient batch-verification mechanism, thereby optimizing certificate management. To visually demonstrate the characteristics of existing studies and the motivation for the proposed scheme, Table A1 in Appendix A provides a detailed comparative analysis of the key related works mentioned above.

In summary, although existing studies have made notable progress in specific areas, challenges remain. In MEC environments, how to simultaneously achieve low computational costs, privacy protection under semi-trusted servers, and efficient group key agreement remains an open issue.

1.3. Contributions

To address the limitations of existing studies, this paper proposes a blockchain-based batch authentication and group key agreement protocol. The following innovations in architectural design and security logic have been achieved:

(1) A unified "Authentication–Negotiation" security architecture is proposed. To overcome the limitations of traditional schemes that isolate authentication from negotiation, a tightly coupled protocol framework is designed. By reusing the parameters from the chameleon-hash authentication phase for the subsequent group key agreement, the protocol achieves joint optimization of communication and computation, reducing overall system overhead.

(2) A lightweight chameleon-hash primitive tailored for MEC is constructed. A specific chameleon hash function based on elliptic curves is developed. By leveraging its trapdoor collision property for efficient signature aggregation, the computational complexity of batch verification is reduced from linear to near-constant time, thereby addressing efficiency bottlenecks in high-concurrency scenarios.

(3) A Merkle Tree-based mechanism against internal attacks is designed. Merkle Trees are applied to session key verification in an innovative way. This mechanism enables terminals to independently validate the correctness of keys, thus providing a theoretical guarantee against internal tampering attacks by semi-trusted ES and filling a security gap in existing schemes.

1.4. Organization

The structure of this paper is as follows. Section 2 presents the system model of the proposed scheme and describes the batch authentication protocol in MEC environments. Section 3 presents a rigorous security analysis of the proposed scheme, including both informal reasoning and formal verifications of the authentication and group key agreement phase using ProVerif2.05 and Burrows-Abadi-Needham (BAN) Logic, respectively. Section 4 discusses the experimental results and performance evaluation. Finally, Section 5 concludes the paper.

2. System Model and Protocol

2.1. Blockchain Technology Overview

Blockchain is a decentralized and tamper-resistant distributed ledger maintained by multiple nodes without relying on a central authority, and it provides transparency and traceability. Figure 1 illustrates the structure of the blockchain. Each block contains transaction records, timestamps, and the hash of the previous block, forming an immutable chain structure secured by cryptographic mechanisms. Consensus algorithms, such as Proof of Work and Proof of Stake, ensure that distributed nodes agree on the correctness of newly added blocks. Smart contracts further enable automatic execution of predefined rules, enhancing the trustworthiness and automation of the system. These properties make blockchain suitable for secure authentication and key management in MEC environments, helping reduce data-tampering risks and improving the reliability of the authentication process.

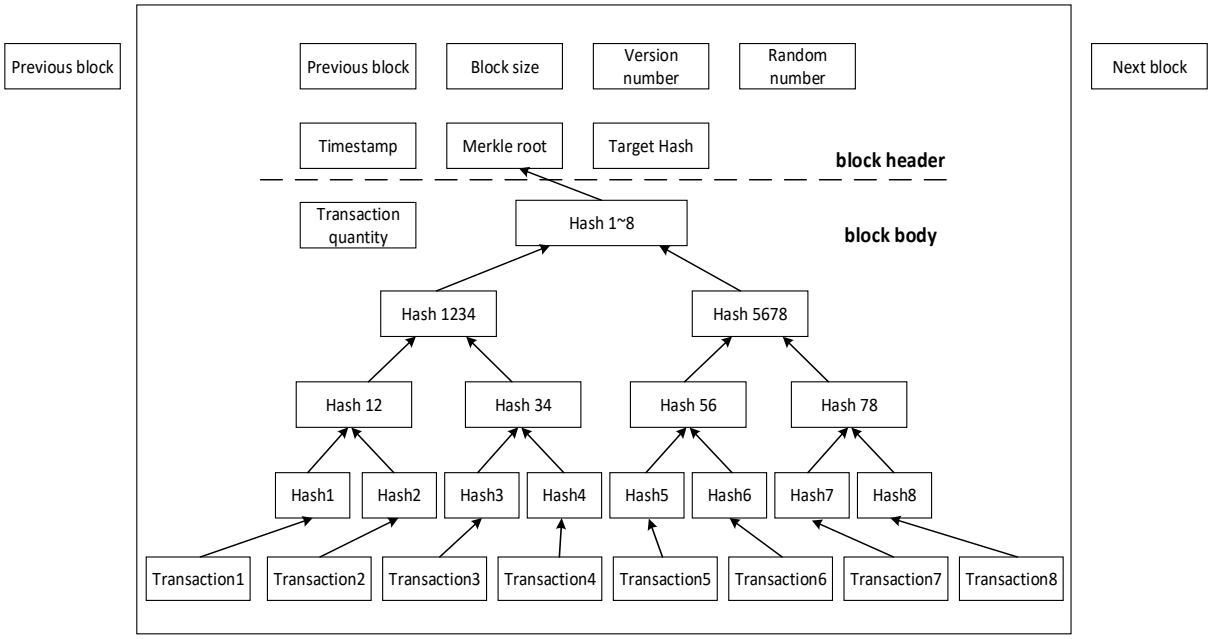


Figure 1. Blockchain structure.

2.2. System Model

The network model for batch authentication is illustrated in Figure 2, consisting of four main entities: Registration Authority (RA), ES, SMD, and Blockchain(BC). Their respective roles are described as follows:

(1) RA: The RA is a fully trusted entity deployed on the cloud server and trusted by all participants. Its primary responsibilities include registering and updating both SMDs and ESs, as well as maintaining the integrity of the data stored on the blockchain.

(2) ES: The ES is a semi-trusted entity equipped with moderate storage and computational capabilities. The permissioned blockchain is located at the edge layer. The ES follows an honest-but-curious model, meaning that it correctly executes the key agreement protocol but may attempt to infer or leak user privacy information.

(3) SMD: The SMD represents the user entities in the MEC system, such as UAVs, smartphones, and smartwatches. An SMD may connect to multiple ESs, but it typically selects the nearest one for authentication.

(4) BC: The BC operates as a permissioned blockchain, which stores device identity information and key materials. Smart contracts are employed to manage key materials—recording, updating, and revoking them as necessary—thereby serving as a trusted and secure container for key management operations.

Based on the framework depicted in Figure 2, a blockchain-driven protocol combining batch authentication and group key agreement via a chameleon hash function is proposed to enhance reliability and meet the security requirements of MEC environments. The proposed protocol consists of seven phases: system initialization, registration of ES and SMD, data uploading to the blockchain, distributed key generation, batch authentication, group key agreement, and session key agreement. The protocol process is illustrated in Figure 3.

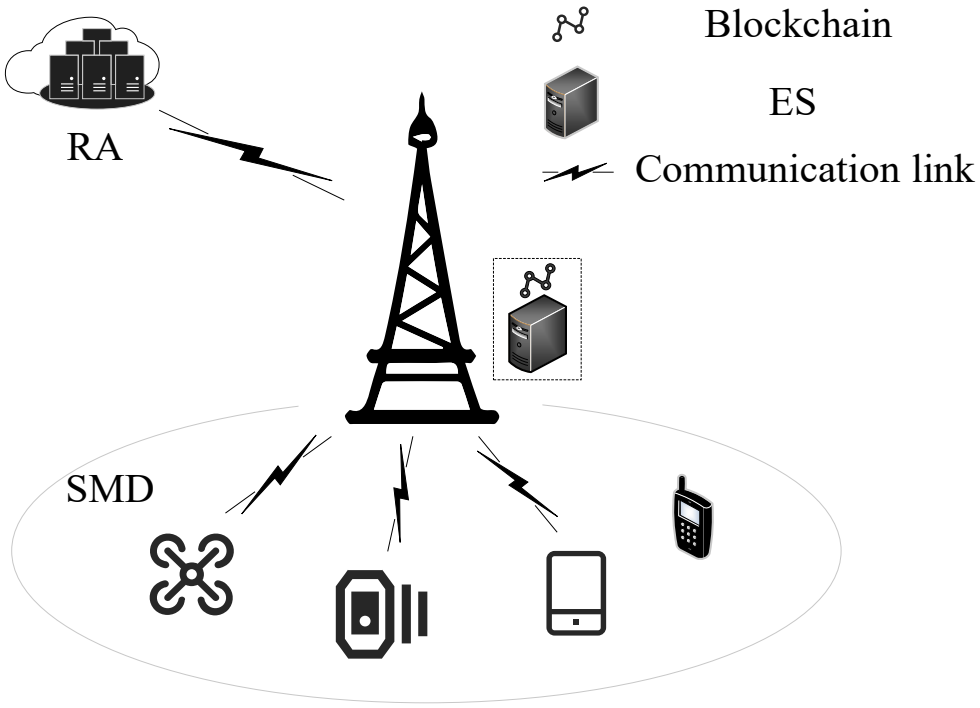


Figure 2. Batch authentication network model.

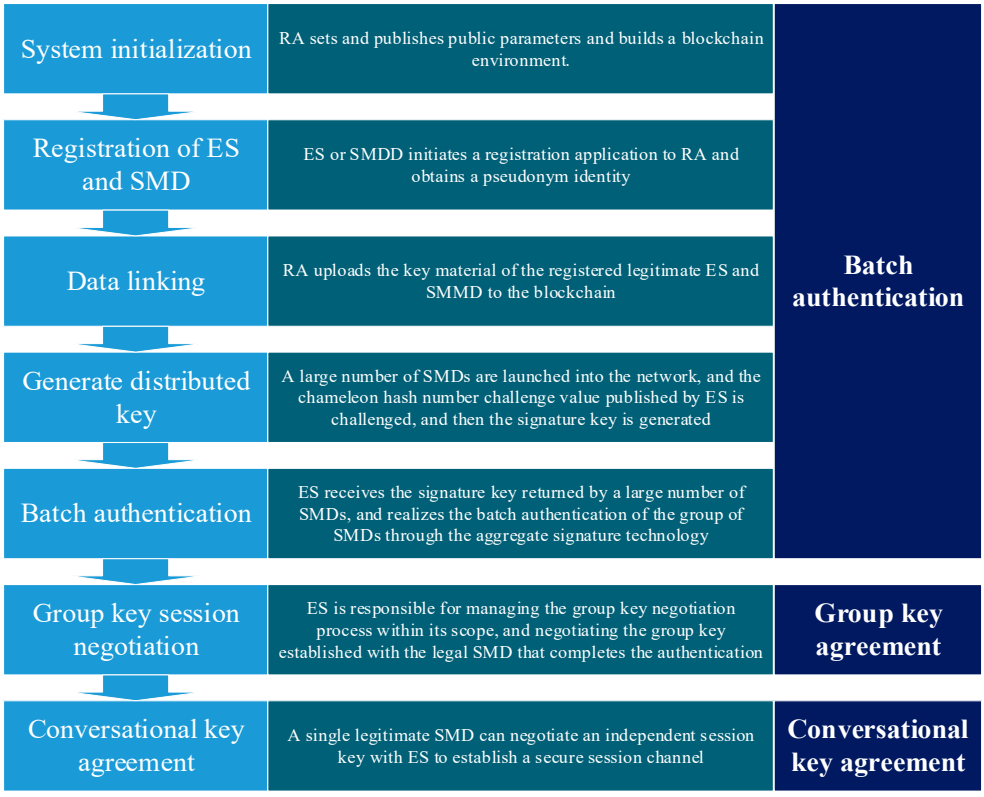


Figure 3. System protocol model.

2.3. Batch Authentication Protocol in MEC

The purpose of the protocol is to negotiate an independent session key between the ES and each SMD, as well as a shared group key among SMDs, so as to achieve mutual secure communication between the ES and SMDs, and among the SMDs themselves. It is assumed that SMD_i and the ES are mutually aware of each other's public keys, PK_i and

PK_j, respectively. The key parameters involved in the protocol and their corresponding meanings are listed in Table 1.

Table 1. Key parameters and their meanings.

Parameter	Meaning
q	Large prime number
s	System private key
PK	System public key
ID_i	Identity of the i-th SMD
ID_j	Identity of the j-th ES
(x_i, PK_i)	Private/public key pair of the i-th SMD
(x_j, PK_j)	Private/public key pair of the j-th ES
HID_i	Signature data of the i-th SMD
m_i, r_i, m_i'	chameleon hash function challenge random value
$TSMD_i, TES_i$	Timestamp
GSK	Group session key
SK	Session key shared between SMD and ES
$h_i(i = 1, 2)$	Hash function
$\oplus$	XOR encryption operation
$\|\|$	Concatenation operator

2.3.1. System Initialization Phase

The RA selects an additive cyclic group G of prime order q with generator P over an elliptic curve $E(Fp)$ defined on a finite field Fp. It then chooses two secure one-way hash functions:$h_1 : \{0,1\}^* \rightarrow Z_q^*$, $h_2 : \{0,1\}^* \rightarrow \{0,1\}^l$, where l denotes the bit length. A chameleon hash function is defined as $\mathcal{F}_C : M \times Z_q^* \rightarrow Z_q^*$, where M represents the message space. The RA generates a random secret $s \in Z_q^*$ as the master private key and derives the corresponding master public key $PK = s \cdot P$. The finalized system parameters are then released as: params = $\{G, P, q, PK, h_1, h_2, \mathcal{F}_C\}$.

2.3.2. Registration Phase of ES and SMD

This phase is executed over a secure channel. Each SMD or ES submits its identity identifier, ID_i or ID_j, to the RA for registration. The registration process, taking the SMD with identity ID_i as an example, is described as follows:

(1) $RA \rightarrow SMD : M1 = (x_i, TP_i, k)$

- Upon receiving ID_i, the RA selects a random number $tp_i, x_i \in Z_q^*$ and computes $PK_i = x_i \cdot P$, $TP_i = tp_i \cdot P$, and $HID_i = h_2(ID_i||PK_i||x_i)$;
- The RA then computes $PID_i = h_2(PK_i)$, and $CP_i = ID_i \bigoplus h_2(HID_i||PID_i||s)$. Then the RA securely stores (CP_i, PID_i, HID_i);
- The RA calculates $k = tp_i + s \cdot HID_i$, and sends the message $M1 = (x_i, TP_i, k)$ to the SMD.

(2) $SMD : (x_i||PK_i||ID_i)$

- Upon receiving $M1$, the SMD computes $PK_i = x_i \cdot P$, $HID_i' = h_2(ID_i||PK_i||x_i)$, then verifies whether $k \cdot PTP_i + HID_i' \cdot PK$. If the verification succeeds, the SMD securely stores $(x_i||PK_i||ID_i)$.

2.3.3. Data On-Chain Phase

The purpose of this phase is to store key information on the blockchain, thereby enhancing the trustworthiness of the edge server. After registration, the RA sends a transaction containing $(HID_i||PK_i||C_i)$ to invoke a smart contract, which records the

information on the blockchain. The smart contract supports the operations of adding, modifying, and deleting SMD information. If the public information of the SMD, namely the data $(HID_i||PK_i||CP_i)$, is successfully uploaded to the blockchain, the corresponding SMD will be recognized as a legitimate terminal within the MEC system.

Following this phase, the edge server can readily access the verification key material of the corresponding SMD, while the SMD's real identity cannot be derived from the blockchain.

2.3.4. Distributed Key Generation Phase

Figure 4 illustrates the chameleon hash-based authentication model of the protocol. The signature phase is initially triggered by the SMD requesting authentication and is completed after two rounds of interaction between the SMD and the ES. This phase is conducted over an open public channel. The detailed procedure is described as follows.

(1) $SMD_i \rightarrow ES : Q_{i1} = \left(HID'_i, EID_i, TSMD_i\right)$

- The SMD computes $HID'_i = h_2(ID_i||PK_i||x_i)$, then obtains the current timestamp $TSMD_i$ and calculates $EID_i = h_2\left(HID'_i||TSMD_i\right)$;
- Next, the SMD sends $Q_{i1} = \left(HID'_i, EID_i, TSMD_i\right)$ to the ES.

(2) $ES \rightarrow SMD_i : Q_{i2} = (\sigma_i, A_i, d_{i1}, kID_i, TES_i)$

- The ES first verifies the timeliness of the message by checking whether the relation $|T - TSMD_i| \leq \Delta T$ holds, where ΔT is the preset time threshold and T is the current timestamp of the ES. All subsequent time verifications are performed in the same manner. The ES then uses HID'_i to query the blockchain and retrieve the corresponding data tuple (HID_i, PK_i), after which it computes and verifies $EID_iEID'_i = h2(HID_i||TSMD_i)$. If the verification fails, the ES discards the message and rejects the request. Otherwise, the ES selects four random numbers $a_i, m_i, r_i, m'_i \in Z_q^*$, records them as $\sigma_i = \left(m_i, r_i, m'_i\right)$, and then computes $A_i = a_i \cdot P$;
- To safeguard data communication across the public channel, the ES must perform encryption on the transmitted data. The ES obtains the current timestamp TES_i, then computes $d_{i1} = \sigma_i \oplus h_2\left(TES_i\big|\big|(a_j + x_j) \cdot PK_i\right)$ and the verification value $kID_i = h_2\left(A_i||\sigma_i||PK_j\right)$;
- The ES sends message $Q_{i2} = (\sigma_i, A_i, d_{i1}, kID_i, TES_i)$ to the SMD.

(3) $SMD_i \rightarrow ES : Q_{i3} = \left(d_{i2}, TSMD'_i\right)$

- The SMD first checks the timeliness of the received message. If the time condition is not satisfied, the data is discarded. Otherwise, the SMD decrypts and computes the data $\sigma_i = d_{i1} \bigoplus h_2\left(TES_i\big|\big|x_i \cdot (A_i + PK_j)\right)$;
- The SMD computes and verifies whether $kID'_i = h_2\left(A_i||\sigma_i||PK_j\right)$ holds. If the verification succeeds, the SMD proceeds to compute $r'_i = (x_i)^{-1} \cdot \left(m_i - m'_i\right) + r_i$ and $R_i = r'_i \cdot P$. Then, the SMD calculates $Y_i = r_i \cdot PK_i$;
- Similarly, the SMD encrypts the data before transmission. The SMD obtains the current timestamp $TSMD'_i$ and computes $d_{i2} = (Y_i||R_i) \bigoplus h2\left(A_i||\sigma_i||TSMD'_i\right)$;
- The SMD sends the data $\mathrm{Q}_{i3} = \left(\mathrm{d}_{i2}, TSMD'_i\right)$ to the ES.

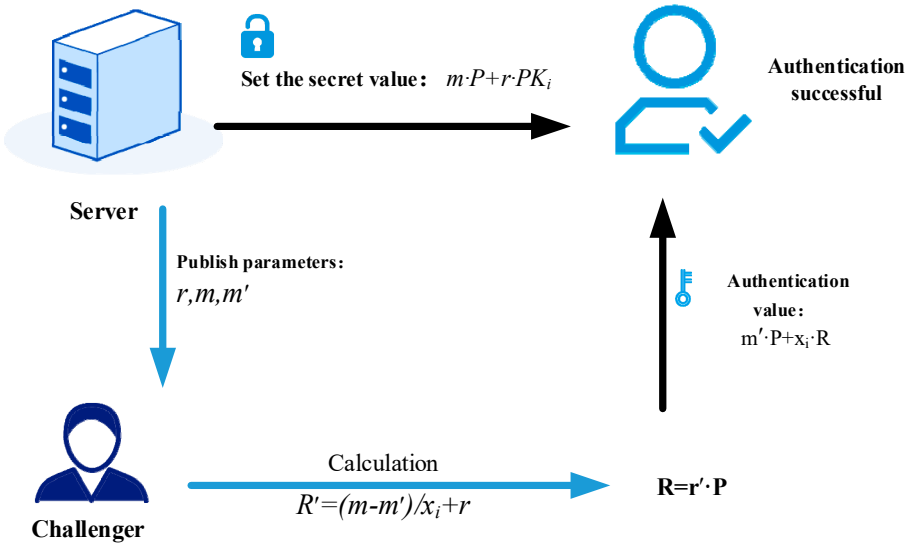


Figure 4. Chameleon Hash-based Authentication Model.

2.3.5. Batch Authentication Phase

In the batch verification phase, the ES needs to verify n messages $(d_{12}, TSMD_1')$, $(d_{22}, TSMD_2')$, …, $(d_{n2}, TSMD_n')$ simultaneously;

(1) The ES uses $TSMD_i'$ to verify the timeliness of the message. If the condition is not satisfied, the message is discarded. Otherwise, the ES decrypts the message $(Y_i||R_i) = d_{i2} \oplus h_2(A_i||\sigma_i||TSMD_i')$;

(2) The ES performs batch authentication on the n signatures. It verifies whether the equation $\sum_{i=1}^{n} Y_i = \sum_{i=1}^{n} (m_i \cdot P + r_i \cdot PK_i) - \sum_{i=1}^{n} m_i' \cdot P$ holds. If the equation is satisfied, the ES accepts the n signatures; otherwise, it rejects the n;

(3) At this point, the ES completes the batch authentication of the SMDs.

As illustrated in Figure 5, the message Q_{i3} encapsulates specific hash commitments and encryption parameters. The aggregate signature verification equation $\sum_{i=1}^{n} Y_i = \sum_{i=1}^{n} (m_i \cdot P + r_i \cdot PK_i) - \sum_{i=1}^{n} m_i' \cdot P$, executed by the ES, is explicitly detailed within the figure, demonstrating the core logic of the batch verification process. Next, we provide the proof of correctness for the batch authentication process:

(1) First, according to the computation of the challenge value r_i', it can be derived:

$$r_i' = \frac{m_i - m_i'}{x_i} + r_i \tag{1}$$

$$(m_i + r_i \cdot x_i) \cdot P = (m_i' + r_i' \cdot x_i) \cdot P \tag{2}$$

(2) Then, the following equation can be derived:

$$\begin{aligned} & m_i' \cdot P + x_i \cdot R_i \\ &= m_i' \cdot P + r_i' \cdot x_i \cdot P \\ &= (m_i' + r_i' \cdot x_i) \cdot P \\ &= (m_i + r_i \cdot x_i) \cdot P \\ &= m_i \cdot P + r_i \cdot x_i \cdot P \\ &= m_i \cdot P + r_i \cdot PK_i \end{aligned} \tag{3}$$

(3) So:

$$\begin{aligned} &\sum_{i=1}^{n}(m_i \cdot P + r_i \cdot PK_i) - \sum_{i=1}^{n} m_i' \cdot P \\ &= \sum_{i=1}^{n} m_i' \cdot P + \sum_{i=1}^{n} x_i \cdot R_i - \sum_{i=1}^{n} m_i' \cdot P \\ &= \sum_{i=1}^{n} x_i \cdot R_i \\ &= \sum_{i=1}^{n} Y_i \end{aligned} \tag{4}$$

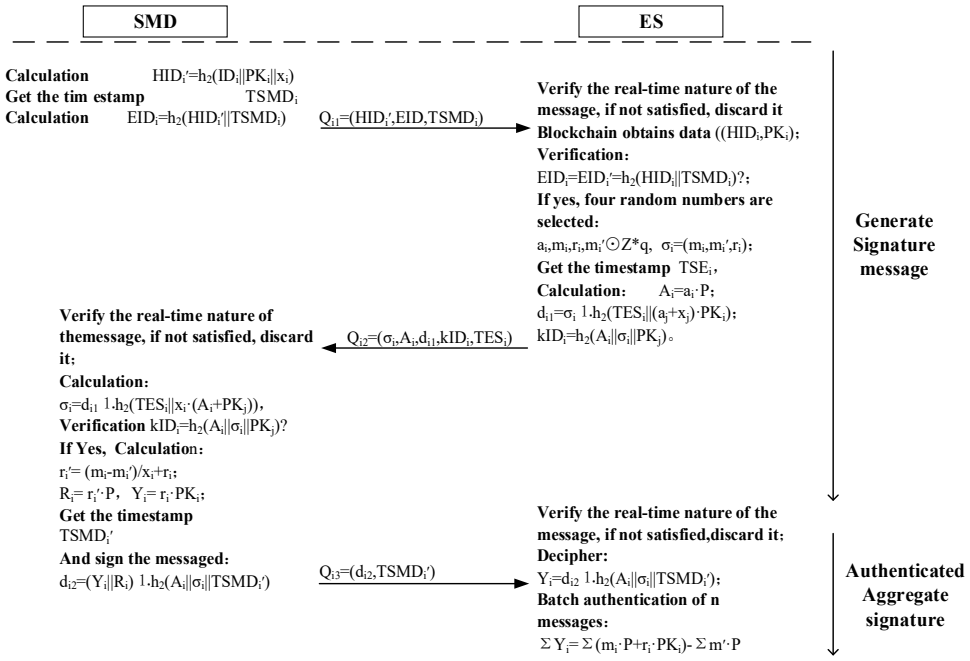


Figure 5. Batch Authentication Process of the Proposed Protocol.

Therefore, the correctness of the batch authentication process is verified.

2.3.6. Group Key Agreement Phase

For the SMDs that have successfully passed authentication and initiated a session request, the ES is responsible for managing the group key agreement process within its domain. The negotiated group key functions as the shared session key among all authenticated SMDs and the ES.

(1) Negotiation Process for a New Terminal Joining the Session Group

In the scenario where v new SMDs join the session group for group key agreement:

(1) $ES \rightarrow SMD_i : Q_{i4} = (E_{SK}, T_{SK}, \Omega)$

- Assume there are k SMDs currently in the group (if no existing SMDs are in the group, then $k = 0$), and n SMDs want to join the group. The ES selects a random number $b_i \in \{0,1\}^q$ for each $i \in [k+1, k+n]$, computes $B_i = h_2(b_i \bigoplus \mathrm{h}_2(Y_i||\sigma_i))$, and the group key $GSK_b^Y = h_2(B_1||B_2||\ldots||B_{k+n})$;
- Subsequently, the ES obtains the timestamp T_{SK}, encrypts GSK_b^Y, and computes $E_{SK} = GSK_b^Y \bigoplus h_2\left(\mathrm{T}_{SK}\middle|\middle|\mathrm{a}_i \cdot R_i + x_j \cdot PK_i\right)$, where $i = 1, 2, \ldots, k+n$;
- The ES computes the signature $\Omega = h_2\left(\mathrm{T}_{SK}\middle|\middle|E_{SK}\middle|\middle|GSK_b^Y\middle|\middle|\mathrm{a}_i \cdot R_i + x_j \cdot PK_i\right)$ on the message and broadcasts message $\mathrm{Q}_{i4} = (E_{SK}, \mathrm{T}_{SK}, \Omega)$ to nearby SMDs.

(2) $SMD_i : GSK_b^Y$

- $\mathrm{SMD}_i (i = 1, 2, \ldots, k+n)$ verifies the timeliness of the message, if it is not satisfied, the message is discarded. Otherwise, the SMD decrypts and computes $GSKSK_b^Y = E_{SK} \bigoplus h_2(T_{SK}||r_i \cdot A_i + x_i \cdot PK_i)$;
- The SMD verifies the signature, computes $\Omega' = h2\Big(T_{SK}\Big|\Big|E_{SK}\Big|\Big|GSK_b^Y\Big|\Big|r_i \cdot A_i + x_i \cdot PK_i\Big)$, and if equation $\Omega\Omega'$ holds, accepts GSK_b^Y as the group key.

(2) The negotiation process for existing terminals leaving the session group
Assume that from a session group containing k SMDs, u members are preparing to leave. The process of updating the group key is similar to the key agreement process when new members join the session group; therefore, the description here will be simplified accordingly:

(1) $\mathrm{ES} \rightarrow \mathrm{SMD}_i : Q'_{i4} = \left(E'_{SK}, T'_{SK}, \omega\right)$

- Each remaining $\mathrm{i} \in [1, \mathrm{k} - \mathrm{u}]$ selects a random number $e_i \in \{0,1\}^q$, computes $E_i = h2(e_i \bigoplus \mathrm{h}_2(Y_i||\sigma_i))$, and the group key $GSK_e^Y = h2(E_1||E_2|| \ldots ||E_{k-u})$;
- The ES obtains timestamp T'_{ES}, computes $E'_{SK} = GSK_e^Y \bigoplus h2\left(T'_{SK}\Big|\Big|\mathrm{a}_i \cdot R_i + x_j \cdot PK_i\right)$, and signs $\omega = \mathrm{h2}\Big(T'_{ES}\Big|\Big|E'_{SK}\Big|\Big|GSK_e^Y\Big|\Big|\mathrm{a}_i \cdot R_i + x_j \cdot PK_i\Big)$;
- The ES broadcasts message $Q'_{i4} = \left(E'_{SK}, T'_{SK}, \omega\right)$ to nearby SMDs.

(2) $\mathrm{SMD}_i : GSK_e^Y$

- Similarly, the SMD verifies the message timeliness. If valid, it decrypts the newly received group key GSK_e^Y and verifies the correctness of the group key using the signature. If the verification succeeds, it sets GSK_e^Y as the new group key.

2.3.7. Session Key Agreement Phase

The Group Session Key (GSK) is a shared key among group members, including the ES and SMD_i ($i \in [1, n]$). However, when a specific SMD needs to upload its independent data to the ES, the group shared session key cannot meet this requirement. Therefore, the protocol uses Ri as the negotiation public key for SMD_i and A_i as the negotiation public key for the ES. Then, SMD_i sets $SK_{SMD}^i = r_i \cdot A_i$ as the shared session key, and the ES $SK_{ES}^i = a_i \cdot R_i$ as the shared session key.

Furthermore, to ensure the independence of the shared session key relative to the group shared key and to enhance its security, the ES employs a chameleon hash function to construct a Merkle tree:

(1) The ES computes $C_i = h_2(m_i \cdot P + r_i \cdot PK_i)$ using the random numbers m_i and r_i selected during the key setup phase, sets C_i as the leaf nodes, and constructs the Merkle tree upwards to generate a Merkle root hash.
(2) The ES broadcasts the root hash value and securely sends the hash values of the sibling nodes along the Merkle path to SMD_i.
(3) SMD_i computes $C'_i = h_2\left(m'_i \cdot P + r'_i \cdot PK_i\right)$ using (m_i, r_i, m'_i) received during the setup phase, and uses this to verify the correctness of the Merkle root and the hash values of the sibling nodes at each level.
(4) If the verification is correct, SMD_i stores the received Merkle root and the hash values of the sibling nodes at each level.

The Merkle tree constructed by the ES allows SMD_i to verify the correctness of R_i at any time. Meanwhile, the redactable Merkle tree using the chameleon hash function allows ciphertext to be modified without changing the hash value, all without exposing the private key. Therefore, even if SMD_i's negotiation public key R_i is compromised, updating R_i will not affect the group shared key. The Merkle proof process is illustrated in Figure 6.

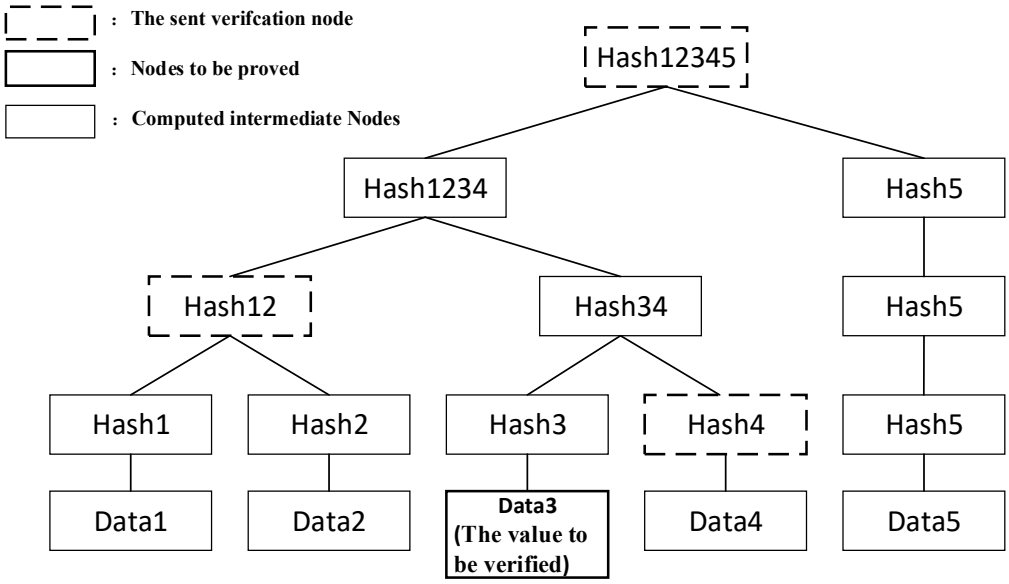


Figure 6. Merkle Proof Process.

3. Security Analysis

3.1. Informal Security Analysis

MEC requires that the designed authentication and key agreement protocol satisfies certain basic security requirements of the system and can resist common malicious attacks. The security of the proposed protocol is analyzed below.

(1) Mutual Authentication. Mutual authentication indicates that the SMD and the ES achieve mutual, bidirectional authentication. During the authentication process between the SMD and the ES, the SMD can authenticate the identity of the ES via the RA. Conversely, the ES authenticates the members within the group and those wishing to join by performing aggregate signature verification using the chameleon Hash algorithm, thereby authenticating the SMDs.

(2) Conditional Anonymous Traceability. Conditional anonymity indicates that throughout the authentication and key agreement process, the SMD sends data anonymously, and only the RA has the authority to trace its real identity. The SMD uses the pseudonym HID_i to send messages, which are stored on the blockchain and protected by the user's private key and a secure hash function. An adversary can only analyze the user's real identity by solving the one-way hash function problem and breaking the user's private key; however, the success probability of this assumption is negligible. When tracing the message source is required, the RA can decrypt CP_i and compute the SMD's real identity $ID_i = CP_i \bigoplus h_2(HID_i||PID_i||s)$. Therefore, the protocol provides conditional anonymous traceability.

(3) Forward and Backward Secrecy. In the protocol proposed in this paper, the group key agreement phase only allows authenticated SMDs to participate and successfully generate the group's public session key. Unauthenticated SMDs or adversaries cannot obtain this session key. Furthermore, the group key ($GSK_b^Y = h2(B_1||B_2||\ldots\Big|\Big|B_{n+v})$) generation method involves each SMD and the challenge value provided by the ES jointly participating in the computation. Whenever a new vehicle joins or an old SMD leaves, the ES issues a new challenge value and updates the group key. Any newly joined SMD, departed SMD, or adversary cannot deduce the previous key (GSK_{b-1}^Y) or the subsequent key (GSK_{b+1}^Y) from the current session key (GSK_b^Y). Therefore, the protocol ensures both forward and backward secrecy.

(4) Resistance to Sybil Attacks. A Sybil attack refers to an adversary attempting to disrupt the normal operation of the network by creating bogus SMDs to send forged messages. In the proposed protocol, the identity of each SMD is authenticated by the ES upon completion of registration; any non-legitimized SMD will not be recognized. Furthermore, an adversary might cause a batch verification containing forged signatures to pass incorrectly by selecting specific messages and forging opposing signatures. However, in the proposed protocol, the ES provides a unique challenge value σ_i to each legitimate SMD, which prevents adversaries from forging signatures and breaks the linear relationship in the aggregate signature.

(5) Resistance to Replay Attacks. A replay attack is an act where an attacker intercepts and re-sends data packets to deceive the system. The proposed protocol effectively resists replay attacks by utilizing a timestamp and hash-signature mechanism. It ensures communication freshness by checking the message timestamp for a reasonable time difference ΔT, and ensures the timestamp has not been tampered with via its inclusion in the hash. This prevents both conventional replay attacks and replay attacks where the attacker modifies the timestamp.

(6) Resistance to Man-in-the-Middle (MitM) Attacks. A MitM attack is a network security threat where an attacker intercepts and potentially modifies the communication between two parties. In this protocol, the Elliptic Curve Diffie–Hellman key exchange between the SMD and the ES, combined with the use of cryptographic hash functions, ensures the authenticity of both parties. To forge a message, an adversary would need the private key of either the SMD or the ES, which remains confidential and is not exposed to the attacker. Therefore, the protocol can effectively resist MitM attacks.

(7) Resistance to Denial-of-Service (DoS) Attacks and Scalability. The proposed protocol employs a multi-layered defense strategy to mitigate DoS threats and ensure high scalability. Firstly, the ES performs lightweight pre-filtering by verifying the freshness of timestamps T_{SMD} and the validity of identities ID_i before executing computationally intensive cryptographic operations, effectively discarding replayed or illegitimate requests at negligible cost. Secondly, the core aggregate signature mechanism establishes a significant computational asymmetry: the cost for an adversary to generate massive valid signatures is significantly higher than the ES's cost to verify them in a batch, thereby inherently suppressing computational DoS attacks. Finally, for practical deployment, we recommend incorporating rate-limiting mechanisms (e.g., Token Bucket algorithm) based on IP or ID at the ES side to further block malicious high-frequency triggers from specific sources, ensuring system availability.

(8) Resistance to Internal Server Attacks. Although our system model assumes an "honest-but-curious" semi-trusted ES, we also address the scenario where an ES becomes malicious or colludes with other entities. Firstly, the protocol adheres to the principle of data minimization; the ES acts solely as an authentication executor and does not store any long-term private keys or sensitive identity information of the SMDs. Therefore, even if an ES is compromised or colludes, it cannot leak critical credentials to compromise an SMD's security. Secondly, relying on the permissioned blockchain architecture, only authorized ESs are registered. During authentication, the SMD strictly verifies the ES's legitimacy using its public key PK_j. If the ES fails verification or is determined to be unreliable, the SMD will immediately abort the connection and refuse access through that ES, effectively isolating the threat from malicious servers.

(9) Resistance to Blockchain-Specific Attacks and Side-Channel Threats. Addressing blockchain-specific vectors, our scheme is deployed on a permissioned consortium blockchain utilizing the Practical Byzantine Fault Tolerance (PBFT) consensus algorithm. PBFT provides deterministic finality, inherently preventing forking issues

common in public chains, and effectively resists consensus manipulation as long as malicious nodes do not exceed one-third of the total network. Regarding smart contract vulnerabilities, the contract logic in our protocol is restricted to simple key-value lookups and hash verifications without complex asset transfers, and has undergone rigorous logic testing to minimize the attack surface. Furthermore, regarding side-channel threats, our protocol design avoids conditional branching based on secret values. In the implementation, we adopt constant-time cryptographic primitives (leveraging standard configurations of the Multiprecision Integer and Rational Arithmetic C Library (MIRACL)) to eliminate timing leakages.

3.2. Formal Analysis Using the ProVerif Tool

ProVerif is a widely recognized automatic protocol verification tool in the field of cryptography. It is based on the applied pi-calculus and operates under the classic Dolev-Yao (DY) attacker model, specifically designed for analyzing and verifying the security of cryptographic protocols. The complete ProVerif verification code used in this article can be found in Appendix B. To construct a feasible formal model in ProVerif, this study adopts the following abstraction strategies based on applied pi-calculus: First, we employ the DY attacker model, assuming perfect cryptographic primitives (e.g., hashing, encryption) where the adversary fully controls the public channel but lacks computational breaking capabilities. Second, for ECC we utilize equation theories in ProVerif to symbolically model the commutativity and associativity of group operations, ensuring the correctness of the Diffie-Hellman key agreement logic. Finally, participating entities are modeled as unbounded concurrent processes to verify security properties under multi-session parallel environments.

Since the batch authentication part operates over an open communication channel, facing dual threats from external attackers and the "honest-but-curious" ES, this chapter primarily focuses on proving the security of identity authentication and the keys during the distributed key generation phase. As shown in Table 2, receivedMessage indicates that the SMD's identity is authenticated by the ES and it receives the batch authentication challenge value from the ES; receivedKey indicates that the ES has authenticated the SMD's identity and received the SMD's batch authentication key material; sentMessage indicates that the ES has completed the authentication of the SMD's identity and sent the challenge value; and sentKey indicates that the batch authentication protocol has been executed by the SMD.

Table 2. Principal Event Definitions.

Key Principal Events
event receivedMessage (bitstring, bool). event receivedKey (point). event sentMessage (bitstring, bool). event sentKey (point).

Based on the definitions of the key principals above, the protocol in this chapter is proven to satisfy basic security properties using the specialized typed process language. As shown in Table 3, the first and second queries are used to verify whether a legitimate SMD and ES can execute the scheme normally; the third query is used to verify the confidentiality of the challenge value exchanged between the SMD and the ES; the fourth query is used to verify the confidentiality of the batch authentication material sent by the SMD.

Table 3. Protocol Security Property Objectives.

Description of Security Objectives
query m: bitstring, Time: timestamp; inj-event (sentMessage(m,false)) ==> inj-event (receivedMessage (m, false)). query Yi: point; event (sentKey (Yi)) ==> event (receivedKey (Yi)). query attacker (m). query attacker (Yi).

To verify the security of the key and the authentication of identity during the distributed key generation phase, as shown in Table 4, a legitimate SMD that has completed registration initiates an authentication request, and the ES responds with a message and interacts with the SMD to complete the protocol. Based on this, two principals are defined: ES (x_j: bignum) and SMD (ID_i: bitstring, x_i: bignum). The execution of the protocol is modeled as these two principals, based on their identities and keys, engaging in a series of message exchanges under the DY attacker model.

Table 4. Main Protocol Process.

Main Verification Process
process new ID_i: bitstring; new x_i: bignum; new x_j: bignum; (!$SMD(ID_i, x_i$ | !ES(x_j)))

The security verification results are illustrated in Figure 7. The first and second "Query" statements indicate that the scheme for the distributed key generation phase can be executed normally and possesses authentication capabilities; the third "Query" indicates that the challenge value of the scheme is confidential and can provide secure and reliable key material for group key agreement; the fourth "Query" indicates that the batch authentication key material is confidential and can be securely sent to the ES to reliably proceed with the next step of the group key agreement operation.

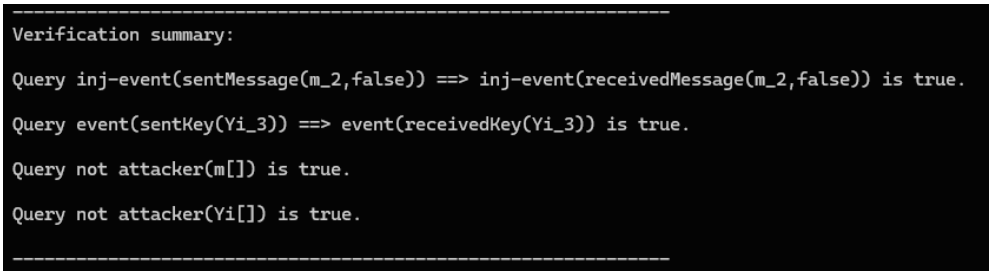


Figure 7. Protocol Execution Security Results.

3.3. Formal Analysis of GKA Phase Using BAN Logic

Given the limitations of the ProVerif tool in simulating dynamic group membership changes and potential state space explosion, we employ BAN Logic to formally prove the security of the Group Key Agreement (GKA) phase, specifically focusing on the confidentiality and authentication of the group key.

In the GKA phase, the ES generates the group key GSK and utilizes the shared secret $K_{seed} = a_i R_i + x_j PK_i$ to encrypt and broadcast it to SMD_i. The idealized protocol is described as follows:

Message 1: $ES \rightarrow SMD_i : \{GSK, T_{SK}, \Omega\} K_{seed}$

To proceed with the logical derivation, we establish the following Initial Assumptions:

A1: ES $\mid \equiv \#(T_{SK})$ (ES believes the timestamp is fresh)

A2: $SMD_i \mid \equiv \#(T_{SK})$ (SMD believes the timestamp is fresh)

A3: ES $\mid \equiv (ES \overset{K_{seed}}{\leftrightarrow} SMD_i)$ (ES believes it shares the seed key with SMD)

A4: $SMD_i \mid \equiv (ES \overset{K_{seed}}{\leftrightarrow} SMD_i)$ (SMD believes it shares the seed key with ES)

A5: $SMD_i \mid \equiv ES \Longrightarrow (ESSMD_i)$ (SMD believes ES has jurisdiction over the group key)

Our goal is to prove that SMD_i believes that GSK is a valid group key generated by ES and is fresh. The specific proof steps are as follows:

Step 1: Based on Message 1 and Assumption A4, applying the Message Meaning Rule:

$$\frac{SMD_i \lhd \{GSK, T_{SK}\}_{K_{seed}}, SMD_i \Big| \equiv (ES \overset{K_{seed}}{\leftrightarrow} SMD_i)}{SMD_i | \equiv ES| \sim (GSK, T_{SK})} \tag{5}$$

This indicates that SMD_i believes ES sent the message containing GSK.

Step 2: Based on Assumption A2 (freshness of the timestamp) and the Nonce Verification Rule:

$$\frac{SMD_i | \equiv \#(T_{SK}), SMD_i | \equiv ES| \sim (GSK, T_{SK})}{SMD_i | \equiv ES| \equiv (GSK, T_{SK})} \tag{6}$$

This indicates that SMD_i believes that ES is confident in the validity of GSK.

Step 3: Based on Assumption A5$ (jurisdiction) and the Jurisdiction Rule:

$$\frac{SMD_i | \equiv ES \Rightarrow \#(GSK), SMD_i | \equiv ES | \equiv GSK}{SMD_i | \equiv GSK} \tag{7}$$

We finally derive that SMD_i believes GSK is a valid group key. Furthermore, since K_{seed} is possessed only by ES and legitimate SMD_i, and the GSK is transmitted in encrypted form, the confidentiality requirement is satisfied.

4. Results and Discussion

To comprehensively evaluate the proposed scheme, this study adopts a "Construction-Verification-Simulation" hybrid research methodology. The research process is structured into three phases: first, constructing the protocol logic based on ECC and Chameleon Hash primitives; second, employing the ProVerif tool to formally verify security properties under the Dolev-Yao attacker model; and third, conducting a two-layer empirical simulation to evaluate performance. Unlike data mining studies that rely on static public datasets, this work focuses on cryptographic protocol design; therefore, our experimental artefacts consist of dynamic execution logs. These include: (1) the precise execution times of fundamental operations (e.g., Scalar Multiplication T_{Gm}) measured via the MIRACL, which serves as the baseline for computational cost analysis; and (2) synthetic transaction logs (latency and throughput) generated by the Hyperledger Caliper tool under a stress test rate of 1050 TPS. To ensure reproducibility and fairness, the experimental environment was configured as follows: Server-side cryptographic operations were benchmarked on a workstation running Ubuntu 16.04 (64-bit) equipped with an Intel Core i3 processor. To establish a consistent baseline, comparative analysis aligns with the experimental parameters of the ECCAS scheme [38]. Furthermore, to address performance concerns regarding heterogeneous devices in real-world MEC environments, we constructed a mobile testbed using a legacy Google Nexus One smartphone (1 GHz ARM CPU, 512 MB RAM) to verify the scheme's feasibility on strictly resource-constrained low-power devices. All underlying cryptographic computations were implemented using the standard MIRACL.

As shown in Table 5, the mobile-side data (SMD/Mobile) is based on actual measurements from a Google Nexus One. The millisecond-level execution time demonstrates the feasibility of the proposed scheme on resource-constrained devices, satisfying the real-time requirements of MEC. It is important to note that in the subsequent cross-scheme comparisons, since the comparative schemes in Refs. [39–41] lack standardized mobile-side data, the analysis will be uniformly based on the benchmark data from the ES (Server) side to ensure the fairness of the comparison. Additionally, the execution times of some lightweight operations and the performance evaluation of the registration phase are omitted, as their impact on the overall system performance is negligible.

Table 5. Experimental Results.

Definitions	Description	Operation Costs (ES/Server)	Operation Costs (SMD/Mobile)
Curve	Elliptic Curve Type (E/F_p)	Type-1 Pairing (p = 512 bits)	-
T_{Gm}	Scalar multiplication on G	0.4420 ms	19.9190 ms
T_{Ga}	Scalar addition on G	0.0180 ms	0.1180 ms
T_h	Hash function	0.0001 ms	0.0890 ms
$\|G\|$	Bit length of an element in G	1024 bits	1024 bits
$\|GT\|$	Bit length of an element in GT	1024 bits	1024 bits
$logq$	Private Key Length (Size of Zq)	160 bits	160 bits
$\|T\|$	Timestamp length	32 bits	32 bits
$\|ID\|$	Identity length	256 bits	256 bits

4.1. Comparative Analysis of the Computational Cost of Batch Authentication

This paper compares the computational efficiency of the proposed scheme with that of the CPPBA [39], ECCAS, LBVP [40], and Ref. [41] schemes. Table 6 provides a detailed comparison of the computational costs incurred when processing a single request and when performing batch processing of n requests.

Table 6. Batch Authentication Operations of Each Scheme.

Definitions	Single Request	Batch Processing	Operation Length
LBVP	$11T_{Gm} + 4T_{Ga} + 8T_h$	$2nT_{Gm} + 2nT_{Ga} + nT_h$	$4\|G\| + 4\log q + 2\|T\|$
ECCAS	$6T_{Gm} + 10T_{Ga} + 2T_h$	$3nT_{Gm} + 5nT_{Ga} + nT_h$	$3\|G\| + 5\log q + \|T\|$
CPPBA	$4T_{Gm} + 2T_{Ga}$	$(n + 2)T_{Gm} + nT_{Ga}$	$3\|G\| + 2\log q + 2\|T\|$
PS [1]	$4T_{Gm} + 6T_h$	$(n + 1)T_{Gm} + nT_{Ga}$	$\|G\| + 10\log q + 3\|T\|$

[1] Proposed Scheme (PS).

First, the efficiency of authenticating a single SMD is analyzed. In the CPPBA scheme, four scalar multiplication operations and two scalar addition operations are required, resulting in a computational cost of $4T_{Gm} + 2T_{Ga} \approx 1.8040$ ms; In the ECCAS scheme, six scalar multiplications, ten scalar additions, and two hash operations are required, leading to a total computational cost of $6T_{Gm} + 10T_{Ga} + 2Th \approx 2.8322$ ms; The LBVP scheme requires

eleven scalar multiplications, four scalar additions, and eight hash operations, resulting in a cost of $11T_{Gm} + 4T_{Ga} + 8T_h \approx 4.9348$ ms. In contrast, the proposed scheme only requires four scalar multiplications and six hash operations, with a total computational cost of $4T_{Gm} + 6T_h \approx 1.7686$ ms. To provide a more intuitive comparison among these schemes, Figure 8 illustrates the computational cost of authenticating a single SMD request.

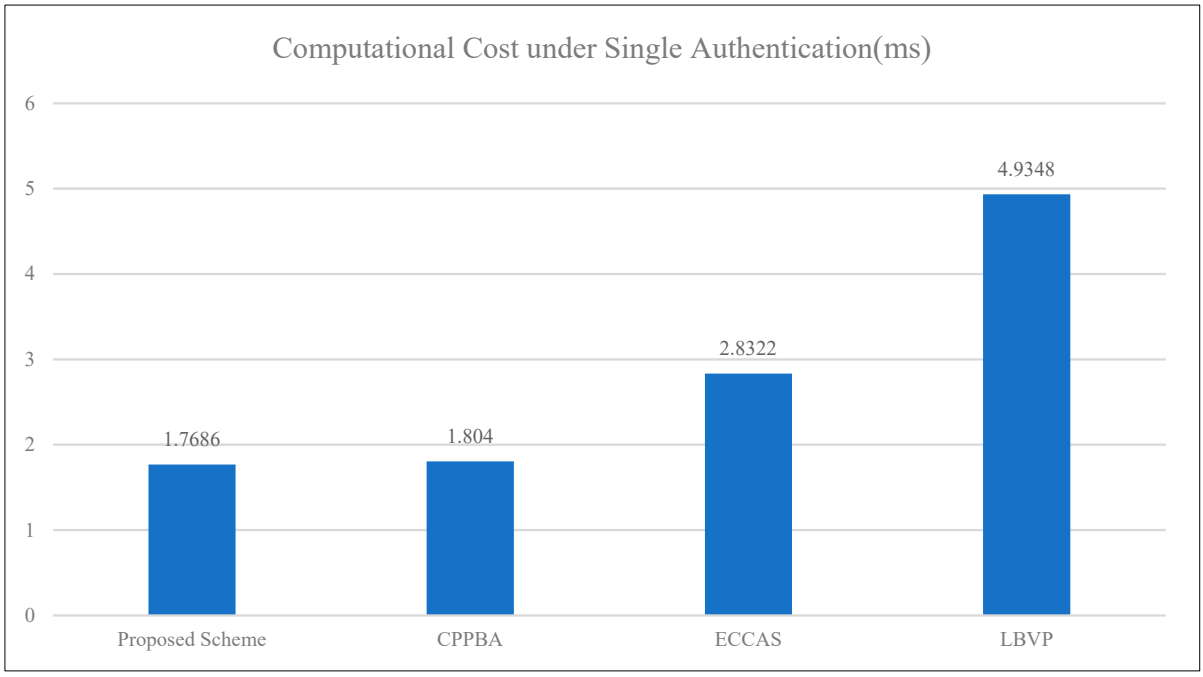


Figure 8. Computational Cost of a Single Authentication.

In all the compared schemes, only scalar multiplication, scalar addition, and hash functions are employed. The proposed scheme exhibits the lowest computational cost, achieving computational efficiency improvements of 1.96%, 37.55%, and 64.16% over the CPPBA, ECCAS, and LBVP schemes, respectively.

To evaluate the efficiency of batch verification, the proposed scheme is compared with the CPPBA, ECCAS, and LBVP schemes. The summarized analysis results are illustrated in Figure 8. In the CPPBA scheme, for batch authentication involving n user terminals, the receiver needs to perform (n + 2) scalar multiplications and n scalar additions. Therefore, its execution time can be expressed as $(n + 2)T_{Gm} + nT_{Ga} \approx (0.46n + 0.884)$ ms. In the ECCAS scheme, the receiver must perform 3n scalar multiplications, 5n scalar additions, and n hash operations, leading to an execution time of $3nT_{Gm} + 5nT_{Ga} + nT_h \approx (1.4161n)$ ms; In the LBVP scheme, 2n scalar multiplications, 2n scalar additions, and n hash operations are required, resulting in an execution time of $2nT_{Gm} + 2nT_{Ga} + nT_h \approx (0.9201n)$ ms. In contrast, the proposed protocol only requires (n + 1) scalar multiplications and n scalar additions, with an execution time of $(n + 1)T_{Gm} + nT_{Ga} \approx (0.46n + 0.442)$ ms.

As shown in Figure 9, during batch authentication, the compared schemes all exhibit the characteristic that their time overhead increases linearly with the number of authenticated SMDs. In contrast, the scheme proposed in this paper demonstrates optimal performance in batch authentication efficiency. Compared to the CPPBA scheme, our scheme maintains a consistent efficiency advantage regardless of the number of SMDs. However, the CPPBA scheme is vulnerable to MitM attacks during its vehicle key generation phase, and its security properties are weaker compared to our scheme. As for the ECCAS and LBVP schemes, their time overhead growth rates are faster, which implies that as the number of authenticated SMDs increases, the performance gap compared to our

scheme will further widen. When processing 100 authentication messages, compared to the CPPBA, ECCAS, and LBVP schemes, our scheme's computational efficiency is improved by 0.94%, 67.20%, and 49.53%, respectively.

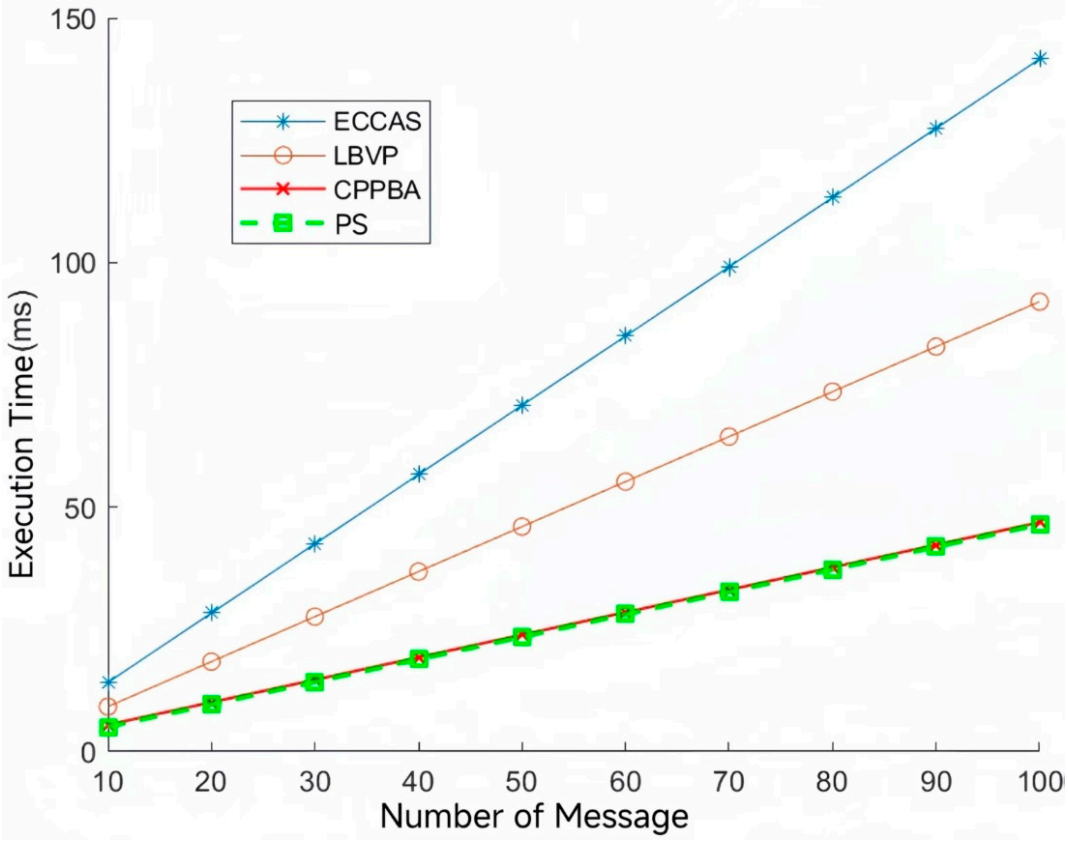


Figure 9. Computational Cost of Batch Authentication.

4.2. Batch Authentication Communication Cost Comparative Analysis

In the MEC environment, SMDs are typically resource-constrained, and their battery life and network bandwidth are precious resources. Therefore, the communication cost of the authentication protocol is an essential indicator of its efficiency and applicability.

Given that the communication cost of batch authentication scales linearly with the cost of a single authentication message, analyzing the communication overhead of a single message provides an effective method for evaluating the overall communication overhead. The overhead required for communication operations is listed in Table 6, and a comparison of the schemes' communication costs is presented in Figure 10. As shown in Figure 10, the protocol proposed in this study has the lowest communication cost. Compared to the CPPBA, ECCAS, and Zhang et al.'s schemes, the communication overhead is reduced by 23.15%, 31.97%, and 44.67%, respectively. Lower communication costs imply lower energy consumption. Our scheme, through its innovative protocol design, effectively addresses the communication bottleneck issue during large-scale device access. This holds critical practical significance for the long-term, high-density operation of resource-constrained devices in the MEC environment.

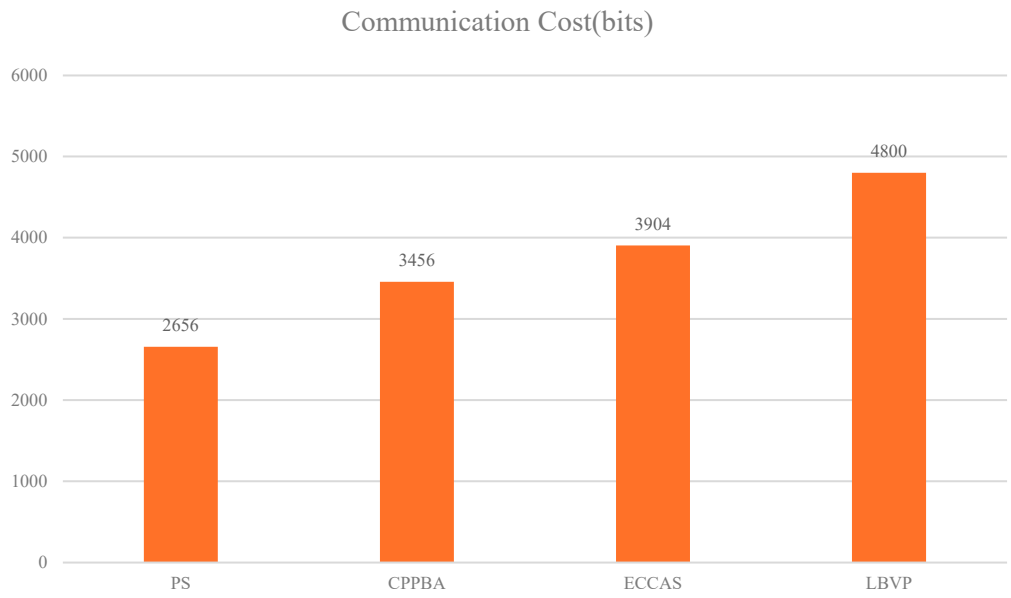


Figure 10. Batch Authentication Communication Cost.

4.3. Comparative Analysis of Group Key Agreement

Next, the computational cost of group key agreement when new devices join the session group will be comparatively analyzed. Since the CPPBA, ECCAS, and LBVP schemes only involve batch authentication, this protocol is compared with the group key agreement in the scheme by Ref. [42]. The experimental data is referenced from Ref. [42], and the relevant cryptographic operations and their costs are shown in Table 7.

Table 7. The Relevant Cryptographic Operations and Their Costs.

Cryptographic Operation	Operation Description	Execution Time/ms
Th	Hash Operation	0.008
Tse	Symmetric Encryption	0.0183
Tsd	Symmetric Decryption	0.0182
Tecc	Elliptic Curve Scalar Multiplication	0.0514
Tcm	Chebyshev Map	0.0336

To simulate and analyze the scenario of new devices joining the session group in greater detail, this experiment designed a simulation process starting with 10 SMDs and incrementally adding 10 devices at a time, until the total number of members in the session group reached 100. This incremental approach is intended to provide a more practical evaluation of the computational cost as the number of session group members increases. Furthermore, to enhance the performance comparison, Table 8 details the specific computational operations of the group key agreement process as described in our proposed scheme and in Ref. [42]. This allows for an intuitive comparison and analysis of the efficiency of the two methods at different session group scales.

The computational cost of group key agreement is summarized in Figure 11. Although Ref. [42] adopts the more efficient Chebyshev map technique, its authentication ring construction process requires performing two computation operations (on the left and right sides of the ring, respectively), thereby introducing redundant resource consumption. In contrast, our scheme utilizes the Elliptic Curve Diffie-Hellman key exchange, requiring fewer operations when updating the group key, which effectively reduces the computational cost.

Table 8. Group Key Agreement Operation [1].

	New Device Join	New Group Key Establishment
Proposed Scheme	n*Th*	(2k + 2n)*Tecc* + (4k + 4n + 1)*Th*
Ref. [42]	4n*Tcm* + 6n*Th*	4(k + n)*Tcm* + (4k + 4n + 1)*Th* + (k + n)*Tse* + (k + n)*Tsd*

[1] Assume there are k SMDs in the original session group, and n SMDs join.

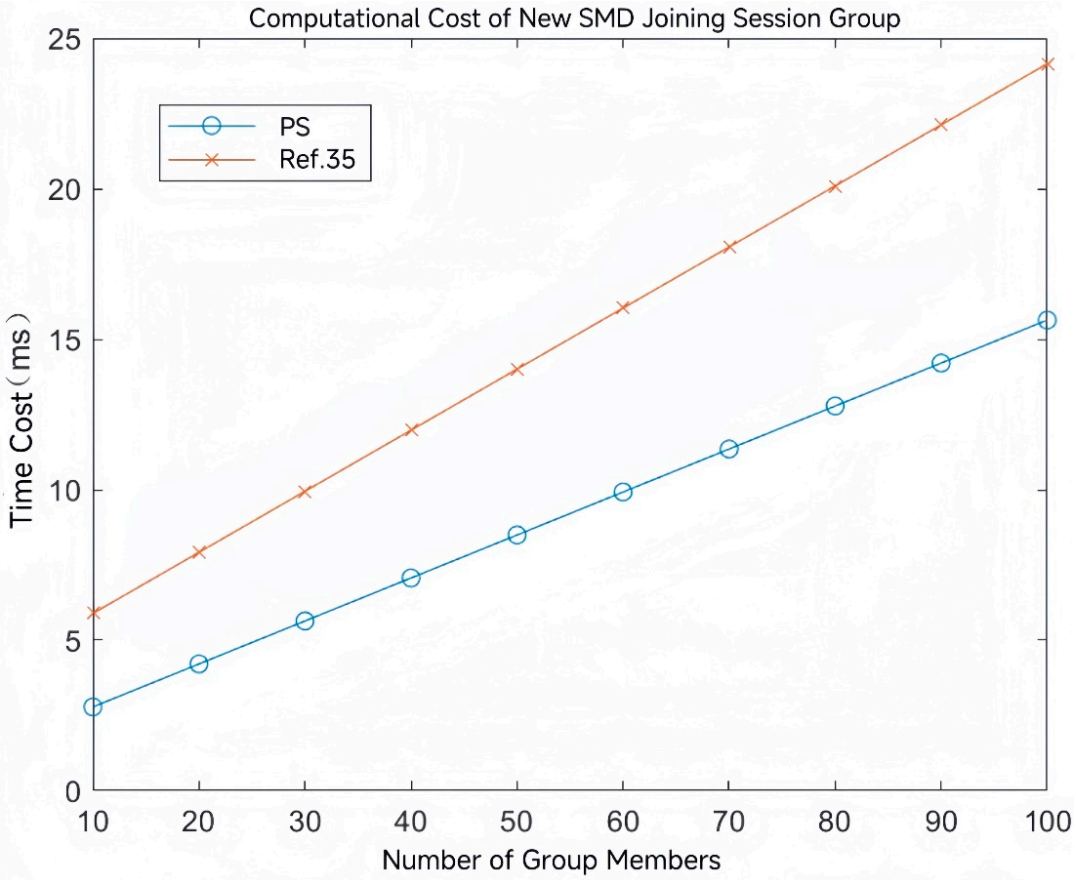


Figure 11. Group Key Update Cost Comparison.

From the analysis of the results, it is evident that as the number of group members increases, the computational cost growth rate of the scheme by Ref. [42]. is significantly higher than that of our proposed scheme. When the number of session group members reaches 100, the computational cost of the scheme by Ref. [42]. reaches 24.1510 ms, whereas our scheme is only 15.6360 ms, achieving an efficiency improvement of 35.26%.

To further evaluate the scalability of the protocol in large-scale group scenarios, we performed a linear regression analysis based on the empirical data for n = 10 to 100 (as shown in Figure 11). The results indicate that the computational overhead T (ms) exhibits a high linear correlation with the number of group members n, with a coefficient of determination $R^2 > 0.99$, verifying that the algorithmic time complexity is strictly O(n). Based on this fitted model, we extrapolated the performance for larger scales: for group sizes of n = 500, 1000, and 5000, the projected group key update overheads are approximately 78 ms, 156 ms, and 780 ms, respectively. These projections demonstrate that even under extreme hyperscale concurrency (n = 5000) within an edge server's coverage, the key update latency remains under 1 s, fully satisfying the real-time requirements for asynchronous group communication in MEC environments.

4.4. Blockchain Performance Evaluation

To comprehensively evaluate the practical feasibility of the proposed scheme in the MEC environment, this section presents a rigorous empirical performance test of the smart contracts within the proposed scheme, focusing on analyzing the additional overhead (latency) introduced by the permissioned blockchain and the system's throughput performance.

This study established a consortium blockchain runtime environment based on the FiscoBcos blockchain platform. The nodes were set up using a multi-virtual machine deployment method, and the simulation experiments were conducted on an Ubuntu virtual machine configured with an 8-core CPU and 16GB RAM. To accurately quantify blockchain performance, the Hyperledger Caliper stress testing tool was deployed, focusing on stress testing the core smart contract algorithms (such as key material query and verification).

Figure 12 presents the stress test report of the smart contract under different loads. The target send rate was set to 1050 TPS. As can be seen from the empirical data: the average execution latency of the smart contract is only 0.05 s, with the maximum latency controlled within 0.82 s. This millisecond-level response speed fully satisfies the requirements for quasi-real-time authentication in MEC scenarios. Under high concurrent requests, the transaction success rate remained at 100% (Fail = 0), demonstrating the robustness of the system.

Name	Succ	Fail	Send Rate (TPS)	Max Latency (s)	Min Latency (s)	Avg Latency (s)	Throughput (TPS)
get	10,000	0	1050.0	0.82	0.00	0.05	1049.9

Resource consumption

Type	Name	Memory(max)	Memory(avg)	CPU% (max)	CPU% (avg)
Process	node fiscoBcosClientWorker.js(avg)	78.9MB	76.9MB	5.36	1.99

Test Environment

benchmark config

```
{
  "name": "KMMT",
  "description": "This is a KMMT benchmark of FISCO BCOS for
caliper",
  "clients": {
    "type": "local",
    "number": 70
  },
  "rounds": [
    {
      "label": "get",
      "description": "Test performance",
      "txNumber": [
        10,000
      ],
      "rateControl": [
        {
          "type": "fixed-rate",
          "opts": {
            "tps": 1500
          }
        }
      ],
      "callback": "benchmarks/samples/fisco-bcos/KMMT.js"
    }
  ]
}
```

Figure 12. Smart contract performance stress test results.

To further evaluate the scalability of the system, we tested the system throughput (Z-axis) under different numbers of nodes (X-axis) and different numbers of concurrent clients (Y-axis). The results are shown in Figure 13. When the number of users is constant, as the number of consensus nodes increases, the system throughput changes smoothly without significant decline, indicating that the PBFT consensus algorithm has good scalability under the consortium chain configuration. When the number of nodes is constant, as the number of concurrent users increases, the throughput shows a trend of first rising and then stabilizing, with the peak throughput stabilizing above 1000 TPS.

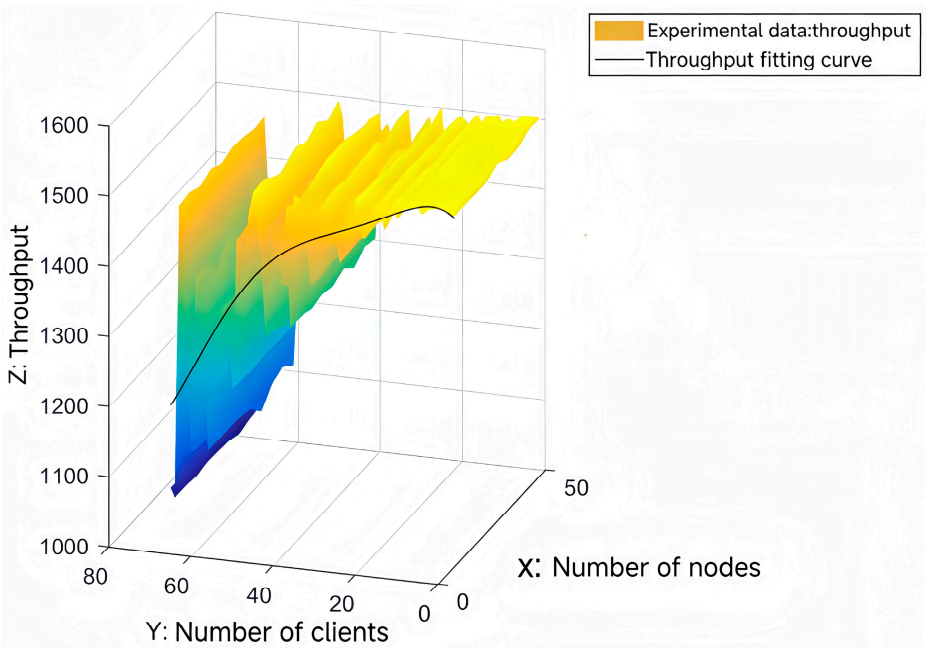


Figure 13. Blockchain throughput analysis under different node and user scales.

The above empirical data indicate that although the introduction of a permissioned blockchain adds a certain amount of computational overhead, the resulting latency (average 0.05 s) is acceptable for most MEC applications. In particular, maintaining a throughput of thousands of TPS at a scale of 50 nodes is sufficient to support large-scale device access management in regional MEC networks. Therefore, the proposed scheme achieves a good balance between security enhancement and performance cost.

4.5. Discussion

The experimental findings presented above provide important insights into the balance between efficiency and security in MEC environments. Notably, the performance advantage of the protocol is structural rather than solely numerical. By leveraging the chameleon hash-based aggregate signature, the verification overhead is decoupled from the number of accessing devices, reducing the batch verification complexity from linear growth to a near-constant-time operation. This architectural shift is crucial for mitigating congestion bottlenecks that commonly arise during large-scale device access.

Although the use of a Merkle Tree for session key verification incurs a marginal computational cost, the analysis indicates that this overhead represents a worthwhile trade-off for strengthening protection against internal threats from semi-trusted servers, a security gap often overlooked in prior efficiency-oriented schemes. The linear regression projections ($R^2 > 0.99$) indicate that the system maintains stable responsiveness even in hyperscale scenarios ($n = 5000$), confirming its suitability for city-scale IoT deployments. These results indicate that the proposed unified architecture achieves an effective balance between

high-performance batch authentication and enhanced resistance to semi-trusted-server attacks, although deployment in public blockchain environments may require additional optimization of consensus latency.

5. Conclusions

To address the challenges of high computational and communication costs and the limited trustworthiness of edge servers in existing batch authentication schemes, this paper proposes a blockchain-based batch authentication and group key agreement protocol that incorporates chameleon hashing. The main contributions are summarized as follows: First, to mitigate efficiency bottlenecks in massive device access, a unified "Authentication–Negotiation" architecture is constructed through an elliptic-curve-based chameleon hash function. This allows the ES to perform aggregate signature authentication on batch messages and efficiently negotiate group keys by reusing authentication parameters, successfully reducing the verification complexity to near-constant time. Second, compared with existing technologies, the proposed approach improves key-management efficiency through smart contracts and enhances the intrinsic security of the edge. For semi-trusted environments, a Merkle-Tree-based verification mechanism is introduced to enable terminals to independently validate key correctness, thereby providing protection against internal tampering attacks. Finally, ProVerif formal analysis and multi-dimensional simulation experiments demonstrate that the proposed method significantly reduces communication and computational costs. In particular, stress tests under hyperscale concurrent scenarios ($n = 5000$) show that the scheme provides clear performance advantages and strong robustness, making it suitable for latency-sensitive and low-power edge-computing environments.

Author Contributions: Conceptualization, Y.D.; methodology, Y.D.; formal analysis, J.Z.; writing—original draft preparation, J.Z.; investigation, J.L. (Jin Liu); writing—review and editing, J.L. (Jinyong Li). All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Collaborative Action Plan (Guangxi Regional Innovation Capacity Enhancement Program), grant number GuikeXT2503960041. This research was also funded by the Guilin city science and technology, grant number 20220138-1.

Data Availability Statement: Data is unavailable due to privacy or ethical restrictions.

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

Appendix A

Table A1. Feature Comparison of Authentication and Key Agreement Schemes.

Scheme (Ref.)	Core Technique	Batch Auth	GKA	Decentralized Trust	Anti-Internal Attack	Main Limitations
Refs. [21,22]	CRT/Chaotic Map	√	√	×	×	Relies on centralized trust models; Vulnerable to internal attacks from semi-trusted servers.
Ref. [23]	Distributed Collab.	√	×	×	×	Coordination between nodes adds communication overhead; Lacks blockchain auditing.

Table A1. *Cont.*

Scheme (Ref.)	Core Technique	Batch Auth	GKA	Decentralized Trust	Anti-Internal Attack	Main Limitations
Ref. [24]	BLS Threshold Sig	×	×	√	×	Focuses on optimizing threshold signature security; Not specifically optimized for massive batch access in MEC.
Ref. [25]	Pairing-free Signcryption	×	√	×	×	Efficient but lacks a blockchain trust anchor; Does not support batch verification.
Ref. [28]	Chameleon Hash + ABE	×	×	√	√	Focuses on redactable storage and fine-grained access control; Not designed for concurrent authentication.
Ref. [32]	Blockchain + CL-AKA	√	√	√	×	Achieves lightweight privacy protection for V2V, but lacks aggregation optimization for ultra-scale batch verification.
Ref. [37]	Blockchain + ECC	√	×	√	×	Achieves decentralized authentication but lacks a subsequent group key agreement mechanism.
Proposed	Chameleon Hash + Merkle Tree	√	√	√	√	(Advantage) Unifies authentication and negotiation; Effectively resists internal attacks via Merkle Tree.

Appendix B

Note: Lines enclosed by (*. . .*) represent comment statements in the ProVerif specification.

ProVerif Code for Formal Verification:

```
(* Type and Function Declarations *)
type bignum.
type point.
type timestamp.
type key.

(* Hash functions, freshness check, Chameleon hash, and Encryption *)
fun h2(bitstring): bitstring.
fun h1(point): bitstring.
fun h3(bitstring): point.
(* Symmetric Encryption *)
fun senc(bitstring, bitstring): bitstring.
(* Decryption destructor *)
reduc forall m:bitstring, k:bitstring; sdec(senc(m, k), k) = m.
(* Chameleon Hash Function *)
fun chameleon(bitstring): bignum.
```

```
fun currentTime(): timestamp.

(* Elliptic Curve Cryptography Operations *)
fun pk(bignum): point.
fun ec_mul(bignum, point): point.
fun ec_add(point, point): point.
fun G(): point.

(* Equation theory to model ECDH commutativity *)
(* This abstraction allows checking g^xy = g^yx *)
equation forall x:bignum, y:bignum; ec_mul(y, ec_mul(x, G)) = ec_mul(x, ec_mul(y, G)).

(* Public Channel *)
free c: channel.
(* Public Parameters *)
free PKi: point.

(* Event Declarations for Authentication *)
event receivedMessage(bitstring, bool).
event receivedKey(point).
event sentMessage(bitstring, bool).
event sentKey(point).

(* Private Data Secrecy Assumptions *)
free m:bitstring[private].
free Yi:point[private].

(* Security Queries *)
(* 1. Authentication: Injectivity correspondence (Resistance to Replay) *)
query m: bitstring, Time: timestamp; inj-event(sentMessage(m,false)) ==> inj-event(receivedMessage(m,false)).
query Yi: point; event(sentKey(Yi)) ==> event(receivedKey(Yi)).

(* 2. Secrecy: Reachability analysis *)
query attacker(m).
query attacker(Yi).

(* ------------------------------------------------------------- *)
(* Process: Message Sending (SMD side) *)
(* ------------------------------------------------------------- *)
let send(ID_i: bitstring, x_i: bignum) =
(* Step 1: Generate initial request *)
new Time:timestamp;
let EID_i = h2((ID_i,Time)) in
out(c, (ID_i,EID_i,Time));

(* Step 3: Receive challenge from ES *)
in(c,(Ai_:point,di1_:bitstring,Kid_:bitstring,Time_ES_:bitstring));
```

```
(* Verify and Decrypt *)
let HES_ = h2((Time_ES_,ec_mul(x_i,Ai_))) in
let m = sdec(di1_,HES_) in

(* Event: Message received successfully *)
event receivedMessage(m,true);

new Time: timestamp;

(* Chameleon Hash Calculation *)
let ri = chameleon(m) in
let Yi = ec_mul(ri,PKi) in
let HashY = h1(Yi) in

(* Encrypt Response *)
let di2 = senc(HashY,h2((m,Time))) in
out(c,(di2,Time));

(* Event: Key material sent *)
event sentKey(Yi).

(* ---------------------------------------------------------------- *)
(* Process: Message Receiving (ES side) *)
(* ---------------------------------------------------------------- *)
let receive(x_j: bignum) =
(* Step 2: Receive request from SMD *)
in(c, (ID_i_: bitstring, EID_i_: bitstring, Time1_:timestamp));

(* Timestamp check (Abstracted) *)
let current_time = currentTime() in
if ((current_time = Time1_) && (EID_i_ = h2((ID_i_,Time1_)))) then

(* Generate Challenge *)
new ai:bignum;
let Ai = ec_mul(ai,G()) in
new Time:timestamp;

(* Encrypt challenge m using shared secret (ECDH) *)
let HES = h2((Time,ec_mul(ai,PKi))) in
let di1 = senc(m,HES) in

(* Event: Challenge sent *)
event sentMessage(m,true);
out(c, (Ai,di1,h2((Ai,m,PKi)),Time));

(* Step 4: Receive Key Material *)
in(c, (di2_:bitstring, Time2_:timestamp));
let HashY_ = sdec(di2_,h2((m,Time2_))) in

(* Event: Key material verified *)
```

```
event receivedKey(Yi).

(* ------------------------------------------------------------- *)
(* Main Process *)
(* ------------------------------------------------------------- *)
process
new ID_i:bitstring;
new x_i:bignum;
new x_j:bignum;
(* Allow unbounded concurrent sessions *)
(!send(ID_i, x_i) | !receive(x_j))
```

References

1. Zhang, X.; Debroy, S. Resource management in mobile edge computing: A comprehensive survey. *ACM Comput. Surv.* **2023**, *55*, 1–37. [CrossRef]
2. Lee, H.; Ryu, J.; Won, D. Secure and anonymous authentication scheme for mobile edge computing environments. *IEEE Internet Things J.* **2023**, *11*, 5798–5815. [CrossRef]
3. Xiao, Y.; Jia, Y.; Liu, C.; Cheng, X.; Yu, J.; Lv, W. Edge computing security: State of the art and challenges. *Proc. IEEE* **2019**, *107*, 1608–1631. [CrossRef]
4. Gad, A.G.; Mosa, D.T.; Abualigah, L.; Abohany, A.A. Emerging trends in blockchain technology and applications: A review and outlook. *J. King Saud Univ.-Comput. Inf. Sci.* **2022**, *34*, 6719–6742. [CrossRef]
5. Krichen, M.; Ammi, M.; Mihoub, A.; Almutiq, M. Blockchain for modern applications: A survey. *Sensors* **2022**, *22*, 5274. [CrossRef]
6. Wan, Z.; Zhou, Y.; Ren, K. zk-AuthFeed: Protecting data feed to smart contracts with authenticated zero knowledge proof. *IEEE Trans. Dependable Secur. Comput.* **2022**, *20*, 1335–1347. [CrossRef]
7. Yazdinejad, A.; Dehghantanha, A.; Parizi, R.M.; Srivastava, G.; Karimipour, H. Secure intelligent fuzzy blockchain framework: Effective threat detection in iot networks. *Comput. Ind.* **2023**, *144*, 103801. [CrossRef]
8. Alkadi, O.; Moustafa, N.; Turnbull, B.; Choo, K.-K.R. A deep blockchain framework-enabled collaborative intrusion detection for protecting IoT and cloud networks. *IEEE Internet Things J.* **2020**, *8*, 9463–9472. [CrossRef]
9. Kumar, R.; Kumar, P.; Tripathi, R.; Gupta, G.P.; Garg, S.; Hassan, M.M. A distributed intrusion detection system to detect DDoS attacks in blockchain-enabled IoT network. *J. Parallel Distrib. Comput.* **2022**, *164*, 55–68. [CrossRef]
10. Salama, R.; Al-Turjman, F.; Bhatla, S.; Mishra, D. Mobile edge fog, Blockchain Networking and Computing–A survey. In Proceedings of the 2023 International Conference on Computational Intelligence, Communication Technology and Networking (CICTN), Ghaziabad, India, 20–21 April 2023; pp. 808–811.
11. Tanveer, M.; Alharbi, A.G.; Akhtar, M.J. A secure and resource-efficient authenticated key agreement framework for mobile edge computing. *Peer-Peer Netw. Appl.* **2025**, *18*, 200. [CrossRef]
12. Li, Y.; Cheng, Q.; Liu, X.; Li, X. A secure anonymous identity-based scheme in new authentication architecture for mobile edge computing. *IEEE Syst. J.* **2020**, *15*, 935–946. [CrossRef]
13. Maurya, C.; Chaurasiya, V.K. Efficient anonymous batch authentication scheme with conditional privacy in the Internet of Vehicles (IoV) applications. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 9670–9683. [CrossRef]
14. Cui, J.; Wang, F.; Zhang, Q.; Gu, C.; Zhong, H. Efficient batch authentication scheme based on edge computing in IIoT. *IEEE Trans. Netw. Serv. Manag.* **2022**, *20*, 357–368. [CrossRef]
15. Khalili, M.; Dakhilalian, M.; Susilo, W. Efficient chameleon hash functions in the enhanced collision resistant model. *Inf. Sci.* **2020**, *510*, 155–164. [CrossRef]
16. Thanalakshmi, P.; Anitha, R.; Anbazhagan, N.; Cho, W.; Joshi, G.P.; Yang, E. A hash-based quantum-resistant chameleon signature scheme. *Sensors* **2021**, *21*, 8417. [CrossRef]
17. Li, C.; Shen, Q.; Xie, Z.; Dong, J.; Fang, Y.; Wu, Z. Efficient identity-based chameleon hash for mobile devices. In Proceedings of the ICASSP 2022–2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Singapore, 23–27 May 2022; pp. 3039–3043.
18. Chang, L.; Yang, X.; Zhang, K.; Sui, Z.; Zhao, J. Blockchain-based synchronized data transmission with dynamic device management for digital twin in IIoT. *Peer-Peer Netw. Appl.* **2025**, *18*, 85. [CrossRef]
19. Zhang, Q.; Zhou, X.; Zhong, H.; Cui, J.; Li, J.; He, D. Device-side lightweight mutual authentication and key agreement scheme based on chameleon hashing for industrial internet of things. *IEEE Trans. Inf. Forensics Secur.* **2024**, *19*, 7895–7907. [CrossRef]
20. Modiri, M.M.; Mohajeri, J.; Salmasizadeh, M. A novel group-based secure lightweight authentication and key agreement protocol for machine-type communication. *Sci. Iran.* **2022**, *29*, 3273–3287.

21. Zhang, H.B.; Chen, Z.; Huang, H.W.; He, X.F. Intra-group mutual authentication key agreement protocol based on Chinese remainder theorem in VANET system. *J. Commun.* **2022**, *43*, 182–193.
22. Miao, J.; Wang, Z.; Miao, X.; Xing, L. A secure and efficient lightweight vehicle group authentication protocol in 5G networks. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 4079092. [CrossRef]
23. Wu, F.; Zhang, X.; Zhang, C.; Chen, X.; Fan, W.; Liu, Y. Batch-assisted verification scheme for reducing message verification delay of the vehicular ad hoc networks. *IEEE Internet Things J.* **2020**, *7*, 8144–8156. [CrossRef]
24. Das, S.; Ren, L. Adaptively secure BLS threshold signatures from DDH and co-CDH. In *Annual International Cryptology Conference*; Springer Nature: Cham, Switzerland, 2024; pp. 251–284.
25. Singh, M.R.; Moulik, S.; Thokchom, S. A novel pairing free certificateless aggregate signcryption scheme for IoMT. *Comput. Electr. Eng.* **2025**, *123*, 110055. [CrossRef]
26. Liu, J.; Yuan, L.; Feng, Z.S. Lightweight dynamic key management scheme for UAV group. *Appl. Res. Comput.* **2023**, *40*, 1515–1521.
27. Xu, G.; Li, X.; Jiao, L.; Wang, W.; Liu, A.; Su, C.; Zheng, X.; Liu, S.; Cheng, X. BAGKD: A batch authentication and group key distribution protocol for VANETs. *IEEE Commun. Mag.* **2020**, *58*, 35–41. [CrossRef]
28. Dong, Y.; Li, Y.; Cheng, Y.; Yu, D. Redactable consortium blockchain with access control: Leveraging chameleon hash and multi-authority attribute-based encryption. *High-Confid. Comput.* **2024**, *4*, 100168. [CrossRef]
29. Wang, M.T. Research on Safe and Efficient Message Authentication Method in Vehicle Ad Hoc Network Environment. Master's Thesis, Xi'an Polytechnic University, Shaanxi, China, April 2022.
30. Zhou, Y.W.; Xue, Q.S.; Sun, C.X.; Ma, H.F.; Ju, X.Z.; Cui, M.X. Batch zero-knowledge identity authentication scheme based on attribute access strategy. *Appl. Res. Comput.* **2023**, *40*, 2487–2492.
31. Feng, X.; Shi, Q.; Xie, Q.; Wang, L. P2BA: A privacy-preserving protocol with batch authentication against semi-trusted RSUs in vehicular ad hoc networks. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 3888–3899. [CrossRef]
32. Lin, H.T.; Jhuang, W.L. Blockchain-based lightweight certificateless authenticated key agreement protocol for V2V communications in IoV. *IEEE Internet Things J.* **2024**, *11*, 27744–27759. [CrossRef]
33. Zhang, J.; Zhong, H.; Cui, J.; Xu, Y.; Liu, L. An extensible and effective anonymous batch authentication scheme for smart vehicular networks. *IEEE Internet Things J.* **2020**, *7*, 3462–3473. [CrossRef]
34. Sutrala, A.K.; Bagga, P.; Das, A.K.; Kumar, N.; Rodrigues, J.J.P.C.; Lorenz, P. On the design of conditional privacy preserving batch verification-based authentication scheme for internet of vehicles deployment. *IEEE Trans. Veh. Technol.* **2020**, *69*, 5535–5548. [CrossRef]
35. Zhang, Z.L.; Wang, H.Q. Dynamic Key Management of Industrial Internet Based on Blockchain. *J. Comput. Res. Dev.* **2023**, *60*, 386–397.
36. Ning, J.G.; Dong, G.F. Blockchain-based vehicle-to-infrastructure fast handover authentication scheme in VANET. *J. Comput. Appl.* **2024**, *44*, 252–260.
37. Yang, Y.; He, D.; Wang, H.; Zhou, L. An efficient blockchain-based batch verification scheme for vehicular ad hoc networks. *Trans. Emerg. Telecommun. Technol.* **2022**, *33*, e3857. [CrossRef]
38. Jia, X.; He, D.; Kumar, N.; Raymond Choo, K.-K. A provably secure and efficient identity-based anonymous authentication scheme for mobile edge computing. *IEEE Syst. J.* **2019**, *14*, 560–571. [CrossRef]
39. Rajkumar, Y.; Kumar, S.V.N.S. An elliptic curve cryptography based certificate-less signature aggregation scheme for efficient authentication in vehicular ad hoc networks. *Wirel. Netw.* **2024**, *30*, 335–362. [CrossRef]
40. Xiong, W.J.; Wang, R.M.; Wang, Y.J.; Zhou, F.; Luo, X.N. A Conditional Privacy-Preserving Batch Authentication Scheme Based on Certificateless Aggregate Signature for VANETs. *J. Cryptologic Res.* **2023**, *10*, 462–475.
41. Zhang, X.; Zhong, H.; Cui, J.; Bolodurina, I.; Liu, L. Lbvp: A lightweight batch verification protocol for fog-based vehicular networks using self-certified public key cryptography. *IEEE Trans. Veh. Technol.* **2022**, *71*, 5519–5533. [CrossRef]
42. Zhang, H.B.; Lan, K.; Chen, Z.; Wang, R.Y.; Zou, C.; Wang, M.Y. Ring-based efficient batch authentication and group key agreement protocol with anonymity in Internet of vehicles. *J. Commun.* **2023**, *44*, 103–116.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.