


Distributed Group Key Management Based on Blockchain

Jia Ni , Guowei Fang, Yekang Zhao, Jingjing Ren, Long Chen * and Yongjun Ren

Engineering Research Center of Digital Forensics, Ministry of Education, School of Computer Science, Nanjing University of Information Science and Technology, Nanjing 210044, China; 002315@nuist.edu.cn (Y.R.)

* Correspondence: 202212200028@nuist.edu.cn

Abstract: Against the backdrop of rapidly advancing cloud storage technology, as well as 5G and 6G communication technologies, group key management faces increasingly daunting challenges. Traditional key management encounters difficulties in key distribution, security threats, management complexity, and issues of trustworthiness. Particularly in scenarios with a large number of members or frequent member turnover within groups, this may lead to security vulnerabilities such as permission confusion, exacerbating the security risks and management complexity faced by the system. To address these issues, this paper utilizes blockchain technology to achieve distributed storage and management of group keys. This solution combines key management with the distributed characteristics of blockchain, enhancing scalability, and enabling tracking of malicious members. Simultaneously, by integrating intelligent authentication mechanisms and lightweight data update mechanisms, it effectively enhances the security, trustworthiness, and scalability of the key management system. This provides important technical support for constructing a more secure and reliable network environment.

Keywords: blockchain; group key management; cryptography; authentication



Citation: Ni, J.; Fang, G.; Zhao, Y.; Ren, J.; Chen, L.; Ren, Y. Distributed Group Key Management Based on Blockchain. *Electronics* **2024**, *13*, 2216. <https://doi.org/10.3390/electronics13112216>

Academic Editor: Hung-Yu Chien

Received: 12 May 2024

Revised: 28 May 2024

Accepted: 4 June 2024

Published: 6 June 2024



Copyright: © 2024 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

Key management primarily provides services such as key generation, distribution, and updates to all legitimate members of a group [1]. In contemporary distributed networks, key management schemes are mainly divided into two major paradigms: centralized, where key management is controlled by a single authoritative entity, and distributed, where key management responsibilities are dispersed among multiple entities or nodes within the network, forming a decentralized governance approach [2]. Centralized key management solutions suffer from problems such as single points of failure, significant delays in cross-domain key transmission, and low efficiency due to excessive reliance on centralization, and are therefore unsuitable for dynamic distributed network environments. At the same time, traditional distributed management relies on centralized entities to distribute keys, which increases security vulnerabilities and trust issues. This is especially obvious in groups with many members or frequent changes. Traditional key management encounters challenges such as difficulty in key distribution, security threats, management complexity, and trust issues, further exacerbating the security risks and management complexity of the system [3]. In view of these challenges, a distributed key management solution based on blockchain provides an innovative solution. The decentralized nature of blockchain ensures the security and transparency of key distribution, while its immutable records enhance reliability. It improves the trust and reliability of the system and ensures the reliability of key management in large-scale and dynamically changing network environments.

The blockchain-based distributed key management solution not only mitigates the risks caused by a single node failure or attack by decentrally storing keys on multiple nodes of the blockchain, but also enhances the recovery from single point failures [4]. Additionally, the inherent immutability and decentralization of blockchain technology are pillars that ensure the security and integrity of keys. By performing cryptographic operations and

verification on each node, the transparency and traceability of key management is enhanced, thereby improving the overall security and trust of the system [5]. Therefore, the distributed key management solution based on blockchain is not only suitable for large-scale dynamic groups, making the key distribution and update process more secure and efficient, but is also of great significance to improving management efficiency and reducing security threats, which is in line with modern distributed network requirements.

With the rapid development of mobile networks, the demand for complex group communication functionalities continues to rise to meet the rapidly growing needs of contemporary users [6]. Group communication, as an efficient means of information dissemination, plays a crucial role in various scenarios [4]. Although it can effectively deliver messages to all group members while saving resources, thus meeting the demands for multiparty communication and collaboration, the security of group communication is often intricately intertwined with the robustness of group key security, posing significant constraints. Currently, many existing group key protocols suffer from deficiencies in performance, scalability, or security, impacting the stability and reliability of group communication systems [7]. Blockchain-based distributed group key management protocols can adapt to dynamic and complex network environments and achieve tracking of malicious members. By integrating intelligent authentication mechanisms and lightweight data update mechanisms, the security, trustworthiness, and scalability of the key management system are effectively enhanced.

In response to the prevailing challenges, this paper posits a pioneering blockchain-based distributed group key management protocol, boasting the following distinctive contributions:

1. Addressing the security vulnerabilities and insufficient system scalability in traditional key management systems, a scheme that combines key management with the distributed attributes of blockchain is proposed, which achieves secure storage and management of group keys, enhances the scalability of the system, and effectively helps to track down malicious members.
2. Addressing the problems of complicated secret key management and difficult verification of members' identities, it is proposed to integrate the intelligent authentication mechanism and lightweight data update mechanism, which enhances the security, trustworthiness, and scalability of the key management system.

The subsequent sections of this paper are meticulously structured as follows: Section 2 provides a comprehensive review of relevant literature and research prospects. Section 3 provides a problem description and provides a detailed explanation of the system model in this paper. Section 4 elucidates the preparatory knowledge, provides an overview of the system, and elaborates in detail on the proposed distributed group key management scheme. Section 5 provides detailed proof of security and privacy and verifies the robustness of the proposed scheme in detail. Section 6 provides a detailed comparative analysis between the proposed solution and numerous established methods. Finally, in Section 7, this paper presents a comprehensive and convincing conclusion.

2. Related Works

Numerous solutions have been proposed by researchers to ensure the security of group keys, encompassing a wide array of methodologies and approaches, as shown in Table 1, which can generally be classified into four distinct types [8–10].

Centralized group key protocols: In the realm of centralized group key protocols, a predominant paradigm emerges wherein a solitary entity, commonly denoted as the Key Distribution Center (KDC), assumes comprehensive dominion over the entire group, orchestrating the intricate ballet of key generation, distribution, and management, while concurrently facilitating the labyrinthine pathways of group communication. Illustrating this archetype, the protocol posited by Wong et al. stands as a quintessential exemplar, notably founded upon the theoretical scaffolding of Logical Key Hierarchy (LKH) [11]. These protocols exhibit a penchant for frugality, demanding scant real estate for key storage

and commendably mitigating the weighty burden of communication overhead inherent in periodic key updates. Furthermore, within the domain of vehicular ad hoc networks, Islam et al. proffered a bespoke group key protocol wherein the mantle of the KDC is assumed by the Trusted Authority (TA), thereby exemplifying an elegant fusion of centralized governance with vehicular network exigencies [12].

Decentralized group key agreement protocol: Within the domain of distributed group key protocols, an alternative modality emerges wherein the collective constituency is split into discrete subgroups, each governed autonomously by its designated subgroup controller. This decentralization not only alleviates the operational burden on the centralized Key Distribution Center (KDC) but also furnishes a robust solution to mitigate the vulnerabilities inherent in single node failures. Pioneering the discourse, Mitra introduces a groundbreaking framework [13] characterized by its scalability, adeptly segmenting expansive group cohorts into a mosaic of distinct subgroups, each shepherded by a dedicated entity termed a group secure intermediary node or group security agent. Echoing this sentiment, Setia et al. [14] advocate for a paradigm shift in key management dynamics, advocating for periodic key updates as opposed to the conventional reactive approach tied to membership fluctuations. Simultaneously, Naresh et al.'s seminal contribution [15] unveils a sophisticated cluster-based hybrid group key protocol, orchestrating the partitioning of vast assemblages into discrete clusters, with the concluding member of each cluster bestowed with the dual mantle of cluster head and group controller, thereby synthesizing the virtues of centralized and distributed architectures.

Distributed group key protocols: Distributed group key protocols epitomize a paradigm shift away from centralized governance structures, fostering a dynamic ecosystem where every member holds equal footing, devoid of hierarchical impositions such as a central Key Distribution Center (KDC) or overarching group controller. Wang et al. spearheaded this progressive movement with their pioneering device-to-device group key negotiation protocol, which was meticulously crafted to operate autonomously without reliance on a base station, thus ensuring the intrinsic anonymity of individual devices while harnessing the cryptographic prowess embedded within the Gap–Diffie–Hellman group [1,16]. In parallel, Kavitha et al. introduced an intricately designed distributed group authentication protocol tailored specifically for IoT healthcare frameworks, ingeniously amalgamating hyperelliptic curve digital signatures with the Elgamal algorithm to fortify the authentication process within this sensitive domain [17]. Meanwhile, the groundbreaking work by Zhang et al. delineated a bifurcated approach wherein the authentication and key negotiation endeavor unfolds across two discrete rounds, commencing with a preliminary stage centered on mutual authentication among group members, culminating in a subsequent phase dedicated to the collaborative generation of group keys [18]. Remarkably, these protocols streamline the authentication process, requiring each member to engage in mutual authentication with only a select pair of counterparts. Furthermore, the confluence of authentication and key negotiation processes into a unified orchestration characterizes the innovative propositions by Zhang et al. [19] and Shi et al. [20]. Noteworthy strides in 2018 include the seminal contributions of Zhang et al. and Gupta et al., who introduced distributed group key protocols endowed with the remarkable capability of self-authentication [21,22].

Asymmetric Group Key Agreement: The Asymmetric Group Key Agreement (AGKA), as initially introduced in [23], diverges significantly from conventional group key agreement (GKA) protocols used for negotiation. Unlike the iterative nature of GKA, AGKA streamlines the negotiation process by enabling a collective bargaining for a shared group encryption key and individual decryption keys among a cohort of users within a single round of interaction. This distinctive approach grants the ability for any entity, irrespective of their group membership status, to disseminate encrypted messages to users within the group using the common encryption key. However, decryption privileges are reserved solely for authenticated group members, who possess the requisite individual decryption keys. It is noteworthy that the AGKA framework elucidated in [23] exclusively addresses static group configurations. Subsequently, to cater to dynamic group dynamics, a series of

nerabilities or tampering attempts. In such a context, blockchain-based distributed key management schemes play a crucial role in improving the security and trustworthiness of systems. Within a decentralized framework, keys are distributed across numerous nodes, which not only mitigates the potential impact of attacks or node failures but also strengthens the system's resilience against single-point failures. Additionally, the inherent immutability and decentralization of blockchain technology serve as pillars for ensuring the security and integrity of keys. This blockchain-based distributed key management model provides a more reliable and secure solution for systems with high security requirements.

3. Problem Statement

Existing key management research has multiple problems in addressing the challenges of dynamic distributed network environments, group communication security, and complex team environments. Centralized schemes are vulnerable to a single point of failure, while traditional distributed schemes face trust and security vulnerabilities. The security of group communication is closely tied to key management, and existing protocols fall short in terms of performance, scalability, and security. In complex team environments, traditional approaches face challenges such as difficulty in key distribution, increased security threats, and increased management complexity. The presence of these challenges can lead to privilege confusion and security breaches, exacerbating the security risks and management complexity faced by the system.

In order to solve the above existing problems, this paper proposes a distributed group key management (DGKM) scheme, which complexly divides blockchain nodes into discrete groups by utilizing the intrinsic ability of blockchain technology and coordinates their interconnections in a hierarchical manner similar to a tree structure by different levels, as shown in Figure 2.

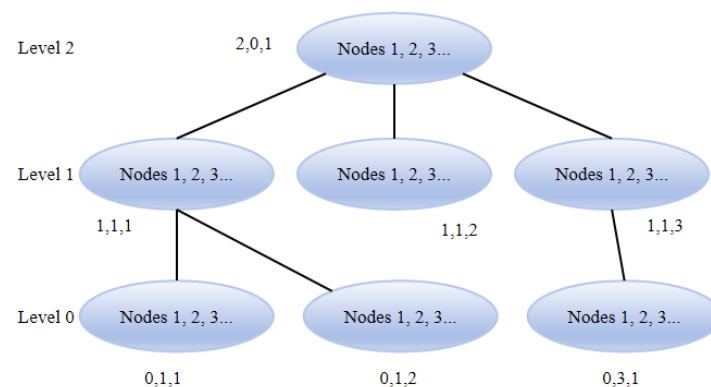


Figure 2. DGKM framework.

At each hierarchical level, a multitude of clusters can be found, each composed of a myriad of nodes, which encompass both the Key Distribution Center and General Nodes (GNs). Except for the KDC, all nodes are considered GNs. As shown in Figure 3, within this network framework, Group Nodes (GNs) coexist as peers, devoid of any hierarchical or subordinate relationships, embodying a harmonious equilibrium in their collective autonomy. These GNs, typically characterized by their stationary nature and lack of energy constraints, possess a reservoir of computational and storage resources, affording them versatility to seamlessly integrate into or depart from group configurations as the exigencies dictate. In the intricate tapestry of this network, the Key Distribution Center (KDC) and GNs converge to form a cohesive blockchain network. However, while the KDC assumes the pivotal role of block genesis, leveraging a proof-of-work mechanism to forge new blocks, GNs are relegated to the more passive roles of block validation and extracting information from the blockchain. Consequently, the KDC stands as the bastion of trust within this intricate network fabric, safeguarding the integrity and reliability of the blockchain infrastructure.

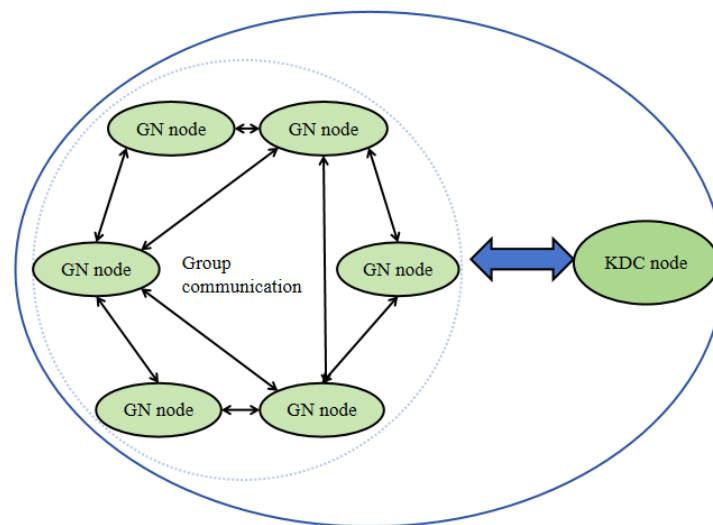


Figure 3. Node network model within the group.

The articulation of our threat model is structured as follows:

- The Key Distribution Center is entrusted with the mantle of being a node of unequivocal trustworthiness within the network architecture.
- Adversaries wield a formidable capability to intercept and manipulate all data traversing through insecure communication channels, thereby wielding the power to introduce novel data, supplant existing information, or engage in the repetition of previously transmitted data.
- Within the ecosystem, all Group Nodes (GNs) are categorized as semi-trusted entities, denoting a propensity for individual instances of misconduct. However, crucially, they are not predisposed towards collusion or coordinated malfeasance with other GNs.

4. Distributed Group Key Management Based on Blockchain

A blockchain-based distributed group key management scheme is proposed to address the security and management complexity problems faced by traditional key management systems in a dynamic distributed network environment. This scheme exploits the distributed and tamper-proof features of blockchain to enhance the security of key management and the reliability of the system. It reduces the risk of single point of failure by storing keys in multiple nodes in a decentralized manner, and improves the overall system efficiency and security by adopting smart authentication and lightweight data update mechanisms.

4.1. Bilinear Pairing

Bilinear pairing, expounded upon in reference [39], occupies a pivotal position at the nexus of cryptographic methodologies, where its intricate functionality and multifaceted applications underscore its indispensability in safeguarding digital communications and fortifying data security. It is a function that maps two group elements to a scalar value, similar to a single number in mathematics. This technology is primarily used to map high-dimensional digital elements to low-dimensional scalars, thereby supporting various cryptographic algorithms, including anti-counterfeiting, digital signatures, key exchange, and key distribution. The security of bilinear pairing is closely related to its computational complexity. To date, no attacks capable of breaking bilinear pairing within a reasonable time frame have been discovered [40]. Therefore, bilinear pairing technology is widely adopted in many cryptographic protocols and applications.

Let us denote G_1 and G_2 as two distinct cyclic groups, both characterized by a prime order q . Within G_1 , let P be designated as its generator, while Q assumes the role of generator

within G_2 . It is presupposed that the discrete logarithm problem remains computationally arduous within both of these groups.

We can define a bilinear map $e : G_1 \times G_1 \rightarrow G_2$, satisfying the following properties:

1. Bilinearity: For all $P \in G_1$, $Q \in G_1$, and scalars a, b , we have $e(P^a, Q^b) = e(P, Q)^{ab}$ in G_2 .
2. Non-degeneracy: If $e(P, Q) = 1$, then there exists $Q \in G_1$, resulting in $P = O$.
3. Computability: There exists an efficient algorithm to compute $e(P, Q)$ for any given P and Q .

These properties make bilinear maps useful in various cryptographic applications, such as identity-based encryption, digital signatures, and cryptographic protocols.

4.2. System Overview

Before integration into the blockchain network, each prospective Group Node undergoes a meticulous procedure involving identity submission to the Key Distribution Center. In response, the KDC assumes the role of identity arbiter, assigning a unique identity ID to the node and orchestrating the remote generation of a key pair—an operation autonomously executed by the node itself. Upon successful authentication, the KDC takes the initiative to compile a new block incorporating the identity particulars and pertinent details of the freshly enlisted node. This block, thereafter, undergoes a stringent validation process orchestrated by fellow nodes within the group, thereby ensuring the veracity and integrity of the new entrant. It is of paramount importance to underscore that, despite its pivotal role in identity verification and key generation, the KDC's involvement remains confined solely to these administrative tasks, abstaining from direct participation in the execution of the group key protocol.

Within our protocol architecture, each Group Node (GN) is entrusted with the initial verification of its left neighbor's identity—a one-time process, following which communication channels are established exclusively with the right neighbor for the purpose of group key negotiation. Moreover, in scenarios involving GN entry or exit from the group, the responsibility of parameter updates rests solely with the left neighbor of the respective GN, thus effectively curtailing computational and communication overhead. The protocol unfolds across six intricately crafted phases: initialization, registration, mutual authentication, group key generation, GN join event, and GN leave event, each meticulously engineered to ensure seamless network operation and robust security protocols.

4.3. The Proposed Scheme

Within the illustrated framework delineated in Figure 2, parent groups are vested with elevated authority, affording them the privilege to access the confidential data belonging to subordinate child groups. Conversely, no group within the hierarchy possesses the capacity to access the sensitive information encapsulated within parent groups or peer groups situated at the same hierarchical level. In the event of any modification to the membership roster of a group, the pertinent group keys undergo updating, a procedural step contingent upon the consensus achieved among the members constituting the root group. Furthermore, at level 0, the computation of group keys is executed for the internal nodes encompassed by each individual group. The group keys, $GK_{0,1,1}$, and $GK_{0,3,1}$ represent the encoding of $(0, 1, 1)$ and $(0, 3, 1)$ for the group, respectively. Through the application of a one-way function, the derivation of group keys for higher-level groups (excluding level 0) entails the utilization of group keys stemming from their respective child groups. This process employs the one-way function, denoted as f , which generates an output of length d . Consequently, it follows that the length of $GK_{i,j,k}$ is also d , thereby ensuring consistency and integrity within the cryptographic framework.

Nodes within the same group are inherently vested with commensurate levels of authority, fostering parity in decision-making and access privileges. Particularly at the lower echelons of the hierarchy, the process of updating group keys hinges crucially upon the consensus achieved among the members comprising the root group, thereby ensuring a

robust and cohesive cryptographic infrastructure. Within the intricate framework of the DGKM model, each group is distinctly delineated by a coded representation, wherein i denotes the hierarchical level, j signifies the positional relation of the parent group within the upper layer, and k indicates the relative position of the parent group within the current layer. It is worth noting that the notation $GK_{i,j,k}$ serves as the symbolic representation for the group key associated with the group identified by the code (i, j, k) , wielding pivotal significance in the encryption and decryption processes pertinent to disseminating public messages among members affiliated with the group specified by the code (i, j, k) .

Let us consider a scenario wherein a multitude of instances, denoted as GN_i ($1 \leq i \leq n$), require the generation of congruous group keys, each instance being uniquely identified by ID_i ($1 \leq i \leq n$), with n representing the total count of GN instances.

The intricate delineation of the aforementioned six components unfolds as follows.

(1) Initialization Phase

The Key Distribution Center (KDC) undertakes the comprehensive generation of parameters $\{G_1, G_2, Q, e, p\}$, where G_1 symbolizes a cyclic additive group of order p , G_2 epitomizes a cyclic multiplicative group of order p , Q acts as the generator, and $e : G_1 \times G_1 \rightarrow G_2$ delineates the bilinear mapping function. Following this, the KDC generates a random private key s and proceeds to calculate the corresponding public key $P_{pub} = sQ$. Subsequently, the KDC meticulously disseminates the parameter set $\{p, G_1, G_2, Q, e, P_{pub}, h, E_k, D_k\}$ while securely preserving s in its memory repository. Herein, H denotes the hash function meticulously integrated into the protocol, E_k symbolizes the symmetric encryption algorithm, and D_k signifies the symmetric decryption algorithm, all playing integral roles in the robustness and security of the system architecture. At this phase, efficient parameter distribution is employed to pre-generate and distribute crucial parameters, ensuring rapid and secure instance initialization while reducing overhead and enhancing throughput.

(2) Registration Phase

Step 1: The initiation of this process commences with the KDC generating a unique identifier ID_i for each GN_i . Employing an asynchronous remote key generation method, it proceeds to create distinctive public-private key pairs (PK_i, SK_i) , concurrently calculating the verification code $S_i = sPK_i$. Subsequently, the KDC securely dispatches the private key to each GN_i via an impregnable channel and publicly discloses (ID_i, PK_i) .

Step 2: To construct a cyclic list L , the KDC meticulously arranges all ID_i in descending order, ensuring seamless connectivity between the highest and lowest values.

Step 3: Introducing a layer of randomness, a random number a_i is generated and utilized to compute $A_i = a_iQ$. This resulting value A_i is then transmitted back to the KDC.

Step 4: Engaging in a process of aggregation, the KDC amalgamates L with multiple tuples (ID_i, A_i) , thereby culminating in the formation of a novel block. This block, subject to validation by all GNs, undergoes meticulous scrutiny. Upon successful validation, it is seamlessly appended to the existing blockchain structure, reinforcing its integrity and expanding its breadth.

During this phase, asynchronous key generation and distribution are utilized, leveraging blockchain for storing public information, achieving independent scaling and efficient secure access, while also establishing a circular list and introducing randomness to disperse computational loads.

(3) Mutual Authentication Phase

Messages are sent to the right neighbor for authentication, while messages received from the left neighbor also need to be authenticated. The following operations are performed:

Step 1: Generate a pseudo-random number m_i and a timestamp t_{l_i} , while simultaneously fetching A_i from the blockchain repository.

Step 2: Calculate $M_i = m_iQ$, $KT_{i+1} = a_iA_{i+1}$, $SE_{i+1} = E_{KT_{i+1}}(M_i)$, and $C_i = h(SE_{i+1}, KT_{i+1}, t_{l_i})S_i$.

Step 3: Send a message (SE_{i+1}, C_i, t_{l_i}) to the right neighbor.

Step 4: Receive a message (SE_i, C_{i-1}, t_{i-1}) from the left neighbor and retrieve A_{i-1} from the blockchain.

Step 5: Verify if $|t_{new} - t_{i-1}| \leq \Delta$, where t_{new} signifies the time of message reception and Δ denotes the maximum permissible communication delay. In the event that this criterion is not satisfied, initiate the dissemination of a message indicating authentication failure.

Step 6: Calculate $KT_i = a_i A_{i-1}$.

Step 7: Examine whether the equation $e(Q, C_{i-1}) = e(P_{pub}, h(SE_i, KT_i, t_{i-1}))PK_{i-1}$ holds true. In the event that this condition is not satisfied, proceed to terminate the ongoing session.

Step 8: Use decryption to obtain M_{i-1} .

Step 9: Produce a stochastic variable b_i and record a timestamp t_{2_i} from the system clock.

Step 10: Calculate

$$X_i = b_i PK_i Z_i = e(M_i - M_{i-1}, Q) \tag{1}$$

$$Y_i = (b_i i + h(X_i, Z_i, t_{2_i})) S_i \tag{2}$$

Step 11: Broadcast $R_i = (X_i, Y_i, Z_i, t_{2_i})$.

In this phase, node processing data volume is reduced through local computation and parallel operations, thereby improving throughput.

(4) Group Key Generation Phase

In this phase, the reception of messages R_r ($1 \leq r \leq n, r \neq i$) from all other GNs initiates the process. Following the authentication of group identities, the negotiation of the group key ensues. The procedural steps are elaborated as follows:

Step 1: Conduct a meticulous examination of the timestamps t_{new} embedded within each received message. In the event of failure to meet validation criteria, instigate the broadcast of a comprehensive notification, indicating the occurrence of identity authentication failure.

Step 2: After receiving all other messages, check if

$$e(\sum_{r \neq i} Y_r, Q)? = e(\sum_{r \neq i} (X_r + h(Z_r, t_{2_r})) PK_r), P_{pub}) \tag{3}$$

If the check fails, broadcast a message of identity authentication failure.

Step 3: Calculate

$$k = e(nM_i, Q) Z_{i+1}^{n-1} Z_{i+2}^{n-2} \cdots Z_{i-1} \tag{4}$$

The group key is given by:

$$K_s = h(k, R_1, R_2, \cdots, R_n) \tag{5}$$

Step 4: If there are child group key computations, assuming c_1, c_2, \cdots, c_n is a child node of group (i, j, k) , compute as follows:

$$GK_{i,j,k} = f(GK_{i-1,k,c_1}, GK_{i-1,k,c_2}, \cdots, GK_{i-1,k,c_l}) \tag{6}$$

During this phase, distributed processing and efficient group key computation are adopted, harnessing blockchain features to enhance scalability.

(5) Join

Upon initiation of the membership process for a new GN seeking inclusion within the group, the initial procedural requirement necessitates undergoing a thorough verification process administered by the KDC.

Following this verification, the KDC meticulously orchestrates the seamless integration of G_j 's identity into the list L at the designated position, ensuring the appropriate alignment within the group's hierarchical structure.

Subsequently, the ensuing step entails the generation of a pseudo-random variable a_j , meticulously calculated to derive $A_j = a_jQ$. This resulting value A_j is then disseminated across the network through a broadcast mechanism, ensuring its propagation to all relevant nodes.

Embark on the generation of a novel pseudo-random variable a_{j-1} , meticulously computed to derive $A_{j-1} = a_{j-1}Q$, intended for dissemination across the network through a broadcast mechanism, ensuring the ubiquitous propagation of A_{j-1} to all pertinent nodes within the network fabric.

The KDC undertakes the meticulous aggregation of all updated tuples (ID_i, A_i) , encapsulating the identities and corresponding authentication parameters of every group member, including G_i , seamlessly consolidating them into a meticulously crafted block. This newly forged block undergoes rigorous scrutiny and validation by all GNs, culminating in its harmonious integration with the existing blockchain architecture, thereby fortifying its structural integrity and augmenting its functional scope.

Analogous to the preceding mutual authentication phase, initiate the transmission of (SE_{j+1}, C_j, t_{l_j}) to the immediate right neighbor, concurrently engaging in the reception of messages $(SE_j, C_{j-1}, t_{l_{j-1}})$ from the adjacent left neighbor.

Subsequent to the meticulous authentication and validation of both (SE_{j+1}, C_j, t_{l_j}) and $(SE_j, C_{j-1}, t_{l_{j-1}})$, undertake the dissemination of $R_i = (X_i, Y_i, Z_i, t_{2_i})$ across the network, thereby instigating the progression to the subsequent phase of group key generation, thereby consummating the key update procedure. The process of node joining a group is shown in Figure 4 (left).

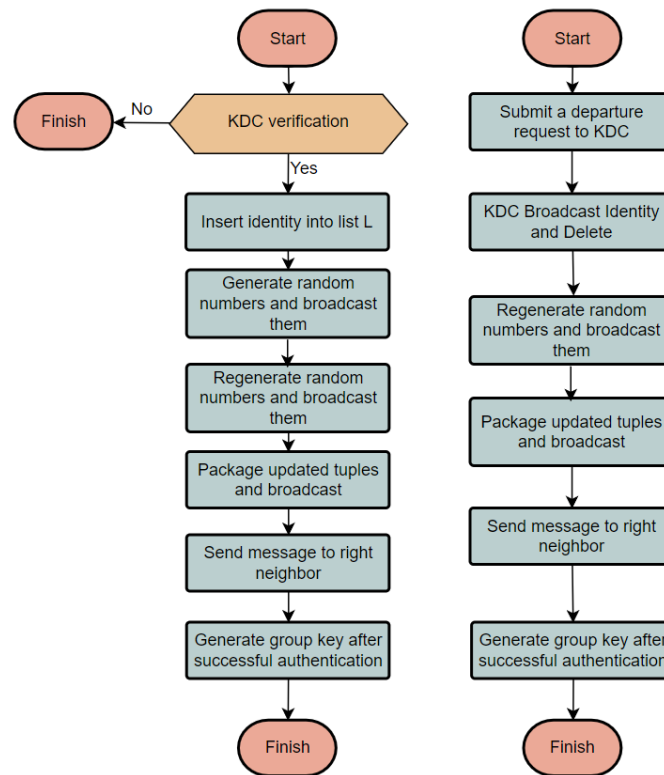


Figure 4. The process diagram for node joining and leaving.

At this phase, network expansion is supported through dynamic list management and efficient broadcasting.

(6) Leave

Upon a GN's decision to depart from the group, the following steps must be taken.

Firstly, initiate the formal submission of an exit application to the KDC, thereby commencing the procedural protocol for departure approval.

Secondly, upon reception of the departure application, the KDC promptly disseminates G_i 's identity throughout the network, concomitantly expunging it from the roster L , thus effectuating the necessary adjustments to the group's membership composition.

Thirdly, embark on the generation of a freshly minted pseudo-random variable a_{j-1} , meticulously computed to derive $A_{j-1} = a_{j-1}Q$, subsequently disseminating it across the network to facilitate widespread acknowledgment and awareness.

Fourthly, the KDC diligently consolidates all modified tuples (ID_i, A_i) , encapsulating the identities and corresponding authentication parameters of every group member, including G_i , into a meticulously assembled block. This newly formed block undergoes exhaustive validation and scrutiny by all GNs, culminating in its seamless integration with the existing blockchain infrastructure, thereby reinforcing its structural integrity and enhancing its operational efficacy.

Fifthly, akin to the antecedent phase of mutual authentication, dispatch $(SE_j, C_{j-1}, t_{l_{j-1}})$ to the immediate right neighbor, fostering the exchange of authentication messages in adherence to the established protocol.

Lastly, subsequent to the meticulous authentication and validation of $(SE_j, C_{j-1}, t_{l_{j-1}})$, initiate the dissemination of $R_i = (X_i, Y_i, Z_i, t_{2_i})$ across the network, thereby signaling the transition to the subsequent phase of group key generation, culminating in the finalization of the key update process. The process of node departure from the group is depicted in Figure 4 (right).

In this phase, controlled exit protocols and seamless key updates are employed to ensure network security consistency while minimizing performance impacts.

5. Security Analysis

5.1. Correctness

Theorem 1. GN_i and GN_{i+1} exhibit the capacity to compute a common symmetric key denoted as KT_{i+1} , thereby enabling GN_{i+1} to acquire M_i .

Proof of Theorem 1. Considering GN_i computes KT_{i+1} as $a_i A_{i+1}$ to derive the aforementioned key, and similarly, GN_{i+1} computes KT_{i+1} as $a_{i+1} A_i$, it follows that KT_{i+1} can be expressed as $a_i A_{i+1} = a_i a_{i+1} Q = a_{i+1} A_i$. Through this equivalence, wherein both computations yield identical KT_{i+1} , it is evident that GN_i and GN_{i+1} possess the requisite cryptographic capabilities for encryption and decryption of messages transmitted utilizing the symmetric key KT_{i+1} . \square

Theorem 2. The adoption of batch authentication for other group members emerges as a highly effective strategy during the group key generation phase.

Proof of Theorem 2. During the intricate process of group key generation, GN_i meticulously scrutinizes the equation: $e(\sum_{r \neq i} Y_r, Q) \stackrel{?}{=} e(\sum_{r \neq i} (X_r + h(X_r, Z_r, t_{2_r}) PK_r), P_{pub})$ and concurrently conducts partial verification in collaboration with fellow group members. The intricate correctness of this equation unfolds as follows.

$$\begin{aligned}
 e\left(\sum_{r \neq i} Y_r, Q\right) &= e\left(\sum_{r \neq i} (b_r + h(X_r, Z_r, t_{2_r})S_r), Q\right) \\
 &= e\left(\sum_{r \neq i} (b_r + h(X_r, Z_r, t_{2_r})sPK_r), Q\right) \\
 &= e\left(\sum_{r \neq i} (b_r PK_r + h(X_r, Z_r, t_{2_r})PK_r), sQ\right) \\
 &= e\left(\sum_{r \neq i} (X_r + h(X_r, Z_r, t_{2_r})PK_r), P_{pub}\right)
 \end{aligned} \tag{7}$$

□

Theorem 3. *In the event that all participating entities GN_i involved in the group key generation phase maintain honesty and integrity, a collective agreement can be established wherein all said entities GN_i are able to collaboratively negotiate and converge upon a unified group key.*

Proof of Theorem 3. As per Theorem 1, provided that all participants partaking in the group key generation phase uphold honesty, it follows that each participant possesses the capacity to procure the parameters M_{i-1} transmitted by its immediate leftward neighbor. Consequently,

$$\begin{aligned}
 k &= e(nM_i, Q)Z_{i+1}^{n-1}Z_{i+2}^{n-2} \cdots z_{i-1} \\
 &= e(m_i Q, Q)^n Z_{i+1}^{n-1} Z_{i+2}^{n-2} \cdots z_{i-1} \\
 &= e(Q, Q)^{nm_i + (n-1)(m_{i+1} - m_i) + (n-2)(m_{i+2} - m_{i+1}) + \cdots + (m_{i-1} - m_{i-2})}
 \end{aligned} \tag{8}$$

□

As inferred from the preceding exposition, it becomes manifestly evident that each participant possesses the computational capacity to derive the uniform parameter k . Henceforth, this collective computation ensures the congruence of their resultant group keys K_s across all members.

5.2. Threat Model and Security Analysis

5.2.1. Threat Model

In this enhanced security analysis, we will cover comprehensive threat models including simulation attacks, modification attacks, replay attacks, man-in-the-middle (MitM) attacks, transient secret leakage, [41], and perfect forward secrecy/security attacks, and integrate these threats with the security mechanisms discussed earlier, as shown in Table 2.

(1) Simulation Attacks

Threat Description: In simulation attacks, the attacker attempts to simulate a legitimate node to gain unauthorized network access.

Security Mechanisms:

- **Multi-layered Authentication:** During registration, the Key Distribution Center (KDC) generates unique public–private key pairs for each node, verifying these keys to ensure attackers have difficulty simulating legitimate nodes.
- **Message Authentication:** Using hash functions and timestamps to authenticate messages, preventing unauthorized node access.

(2) Modification Attacks

Threat Description: In modification attacks, the attacker intercepts and alters transmitted data to disrupt communication or gain unauthorized access.

Security Mechanisms:

- Integrity Check: Hash functions ensure data integrity by generating and validating hash values. Any data tampering triggers a hash mismatch, thus triggering an alert.
 - Digital Signatures: Provides an additional layer of verification, ensuring data has not been tampered with.
- (3) Replay Attacks
- Threat Description: Replay attacks involve the attacker intercepting legitimate communications and replaying them to cause chaos or gain unauthorized access.
- Security Mechanisms:
- Timestamps: Including timestamps in messages ensures old messages cannot be reused. If the time difference exceeds a certain threshold, the message is invalidated.
 - Session Keys: Generating unique session keys for each communication session prevents replay of old messages.
- (4) Man-in-the-Middle (MitM) Attacks
- Threat Description: In MitM attacks, the attacker intercepts and potentially alters communication between two nodes, unbeknownst to both parties.
- Security Mechanisms:
- End-to-End Encryption: Using symmetric encryption and decryption ensures that even if communication is intercepted, attackers without the correct key cannot decrypt it.
 - Mutual Authentication: Before exchanging sensitive information, each node must authenticate using public–private key pairs and hash values, preventing unauthorized intermediaries.
- (5) Transient Secret Leakage
- Threat Description: If transient secrets (temporary keys) are leaked, attackers may decrypt communication for that session.
- Security Mechanisms:
- Frequent Key Updates: Regularly updating temporary keys ensures that even if one key is leaked, it cannot be used to decrypt future communications.
 - Session-Specific Keys: Generating unique keys for each session reduces the impact of transient secret leakage as it is limited to a single session.
- (6) Perfect Forward Secrecy/Security Attacks
- Threat Description: In these attacks, the attacker attempts to decrypt past communications by compromising long-term keys.
- Security Mechanisms:
- Perfect Forward Secrecy: Ensuring that new session keys are independent of long-term key generation, even if long-term keys are compromised, past communications remain secure.
 - Key Derivation Functions: Using one-way functions to derive higher-level group keys from subgroup keys ensures that even if current keys are known, past session keys cannot be reconstructed.

Table 2. Summary of threats and security mechanisms.

Threat	Security Mechanisms
Simulation Attacks	Multi-layered authentication, message authentication
Modification Attacks	
Replay Attacks	Integrity check, digital signatures
Man-in-the-Middle Attacks	Timestamps, session keys
Transient Secret Leakage	End-to-end encryption, mutual authentication
Perfect Forward Secrecy/Security Attacks	Frequent key updates, session-specific keys
	Perfect forward secrecy, key derivation functions

By employing these security mechanisms to address these threats, the proposed framework ensures the confidentiality, integrity, and authenticity of the system. These mechanisms also provide effective protection against denial of service attacks and key leakage attacks, ensuring the robustness and reliability of the system.

5.2.2. Security Analysis

Theorem 4. *It is impossible for any attacker to obtain the value w through the inverse function $z = f(w)$.*

Proof of Theorem 4. The function f functions as a one-way cryptographic function, accepting inputs comprising two or more values, each characterized by a length of d . Upon processing, it yields an output in the form of a fixed-length bit string, also of length d . In the hypothetical scenario where an adversary gains possession of a value z , their potential pursuit to reverse engineer the function in a bid to unveil the original input proves futile. This futile endeavor is primarily attributed to the secrecy maintained around z from non-member entities and the deliberate selection of a sufficiently large value for d , which bolsters the cryptographic resilience of the encryption mechanism against unauthorized decryption endeavors. Even if f were available, obtaining the input is not feasible, as the one-way function is irreversible; it is impossible to determine the input from the output. Therefore, it is impossible to obtain the output in reverse. Thus, the mechanism is secure. \square

Theorem 5. *An adversary cannot construct a GN_i^* satisfying $e(Y_i^*, Q) = e((b_i + h(X_i^*, Z_i, t_{2_i}))sPK_i, Q)$ from the disclosed information in polynomial time.*

Proof of Theorem 5. Despite the adversary's facile acquisition of $\sum_{r \neq i} Y_r, Q, \sum_{r \neq i} (X_r + h(X_r, Z_r, t_{2_r})W_r)$, and the leverage of the Decisional Diffie–Hellman assumption, discerning whether $e(\sum_{r \neq i} Y_r, Q) = e(\sum_{r \neq i} (X_r + h(X_r, Z_r, t_{2_r})PK_r), P_{pub})$ holds within a polynomial time frame remains elusive. To fabricate a counterfeit that successfully navigates the aforementioned batch verification, the adversary must contrive a legitimate sum that satisfies $e(Y_i^*, Q) = e((b_i + h(X_i^*, Z_i, t_{2_i}))sPK_i, Q)$. Initially, the adversary confronts the challenge of acquiring s , rendering the computation of $(b_i + h(X_i^*, Z_i, t_{2_i}))sPK_i$ arduous, contingent upon the Elliptic Curve Discrete Logarithm Problem (ECDLP). Subsequently, assuming the revelation of $(b_i + h(X_i^*, Z_i, t_{2_i}))$ by the adversary, the acquisition of s is precluded. Thus, in accordance with ECDLP, the adversary remains incapable of computing valid X_i^* and Y_i^* within a polynomial time frame. \square

Theorem 6. *This scheme ensures the security of data through various security mechanisms, including confidentiality and integrity, forward secrecy, and prevention of denial-of-service attacks.*

(1) Confidentiality

Confidentiality is achieved through strict layered access control and encryption mechanisms. Parent groups can access data from child groups, but there is no access between groups at the same level or between parent groups. Encryption and decryption operations using symmetric encryption algorithm E_k and decryption algorithm D_k are based on shared group key $GK_{i,j,k}$, ensuring that only legitimate members can access and decrypt relevant data.

(2) Integrity

Hash functions h and digital signatures are used to ensure data integrity during transmission and storage. During message passing and group key updating processes, the integrity of data is ensured by generating and verifying hash values $C_i = h(SE_{i+1}, KT_{i+1}, t_i)S_i$. If data are tampered with, the hash values will not match, and the verification process will fail, triggering an alert mechanism.

(3) Authentication

Each node mutually authenticates its identity, with the KDC generating unique identifiers ID_i and public–private key pairs PK_i, SK_i . During message exchange between nodes, authentication is performed using pseudo-random numbers m_i , timestamps t_i , and hash values C_i , ensuring that only legitimate members participate in communication and key generation.

(4) Key Distribution

Key distribution is managed by the KDC, which generates and distributes system parameters and public–private key pairs during initialization. Group key generation and updating are based on group member consensus and pseudo-random number generation mechanisms. Whenever a member joins or leaves the group, the key is updated, ensuring that new and old members cannot access each other’s data.

(5) Revocation

When a member leaves the group, the KDC promptly updates the group key, ensuring that the departing member cannot continue to access group data. The information of the departing member is deleted from the list L and the corresponding hash value and key are updated, achieving key revocation and data protection.

(6) Forward Secrecy

This ensures that even if a key is leaked, previous communication remains secure. Through regular updates of group keys and the use of pseudo-random numbers to generate new keys a_jQ , each communication uses a new key, ensuring forward secrecy.

(7) Protection against Denial of Service (DoS) Attacks

The system is designed with multi-layered authentication and message verification mechanisms. Each message must pass through a hash function and timestamp verification before being accepted and processed, increasing the difficulty of carrying out DoS attacks.

(8) Protection against Key Leakage Simulation Attacks

If a key is leaked, the system protects against:

- Key Update Mechanism: Regularly updating keys ensures that even if one key is leaked, it cannot be used to decrypt new communication.
- Multi-factor Authentication: Each communication requires strict authentication, preventing attackers from impersonating legitimate members.
- Layered Encryption: Different levels of group keys ensure that even if one level of keys is leaked, it does not affect the security of data in other levels.

6. Performance and Simulation Experiment Analysis

6.1. Performance Analysis

In our comparative evaluation, we conducted a meticulous scrutiny, juxtaposing our protocol against counterparts such as ID-GKM [18], CGKA [42], and other variants [21], across multiple facets including computational complexity, communication overhead, and security robustness.

Within the mutual authentication and group key generation phases of our protocol, the computational burden per GN encompasses a spectrum of tasks, ranging from ECC scalar point multiplication to multiple bilinear pairing operations, in addition to intricate hash computations and symmetric encryption or decryption operations. Correspondingly, the communication overhead per GN entails the transmission of a suite of parameters, including common cryptographic constructs and temporal indicators. As expounded in Table 3, we present a meticulous comparison, meticulously delineating the computational and communication costs of our protocol vis-à-vis related counterparts. Herein, n symbolizes the count of GNs, while C denotes the length of common parameters, assumed to be a concise 160 bits, and the timestamp T is conservatively set at 64 bits.

From the detailed analysis presented in Table 3, it emerges unequivocally that our protocol emerges as the frontrunner, boasting the most parsimonious computational and communication costs compared to its peers. Delving into security considerations, it is discerned that the ID-GKM protocol flounders in both forward and backward secrecy domains, owing to its inertia in updating temporary secret parameters following the ingress or egress of group members. In stark contrast, the CGKA and AGKA protocols exhibit commendable resilience, devoid of any apparent security lapses.

Table 3. The comparison of computational and communication costs between this protocol and related protocols.

	ID-GKM	CGKA	AGKA	DGKM
Point multiplication operation on ECC	$n + 4$	$3n + 2$	$4n$	$n + 7$
Pairing	6	$2n$	0	4
Hashing operation	$n + 4$	0	5	$n + 3$
Symmetric encryption/decryption	0	0	0	2
Point addition operation on ECC	0	0	$2n + 1$	0
Communication cost	$9C + 2T$	$(4n + 8)C$	$7nC$	$5C + 2T$

6.2. Simulation Experiment Analysis

The evaluation of protocol performance is crucial as it reflects the costs we face in the distribution process. We use the Intel i7 quad core processor system to perform these operations, with a clock frequency of 3.4 GHz and 8 GB of memory, running on the Windows 7 operating system. In comparison with other group key management protocols, we conducted a comparative analysis of our approach with schemes GKMT [43] and ID-GKM, focusing on communication costs, computational costs, and other aspects. To ensure the credibility of the results, we evaluated the performance of distributed group key management schemes through extensive simulations of systems with a large number of participants. For groups of varying sizes, we conducted multiple simulations and analyzed the final results. We conducted experimental analyses with groups of different scales, ensuring that the configuration of each node was identical to ensure the comparability and fairness of the experiments. The results were consistent with the theoretical expectations of the schemes, indicating the protocol's robustness in asynchronous environments.

Computational Cost: The computational cost of the DGKM protocol was compared with schemes GKMT and ID-GKM. In terms of computation, the DGKM protocol demonstrated higher efficiency. In contrast, schemes GKMT and ID-GKM exhibited higher computational costs. In the DGKM protocol, the computational overhead per GN encompasses a range of intricate operations, including ECC scalar point multiplication, four bilinear pairing operations, hash computations, and two symmetric encryption or decryption procedures, all contributing to its computational complexity.

In the DGKM protocol, the reduction in computational cost is mainly attributed to optimization strategies embedded in its design. The DGKM protocol effectively utilizes caching of previous computation results and reduces overall costs by minimizing unnecessary computations. Additionally, the DGKM protocol employs a series of carefully designed encryption algorithms to minimize computational burdens during key generation and exchange processes. These optimization measures result in a significant advantage for the DGKM protocol in terms of computational cost. Conversely, schemes GKMT and ID-GKM exhibit relatively higher computational costs. Their higher computational costs primarily stem from complex computational operations required during key generation and exchange processes. Particularly, a significant number of elliptic curve scalar point multiplication operations and bilinear pairing operations are needed in the computation process of each GN, significantly increasing the overall computational burden. Moreover, the incorporation of hash operations and symmetric encryption or decryption procedures not only amplifies the computational overhead but also adds to the intricacy of the computational burden borne by each participant. The research results are illustrated in Figure 5. From the figure,

it is evident that there are significant differences in computational costs between these two group key protocols.

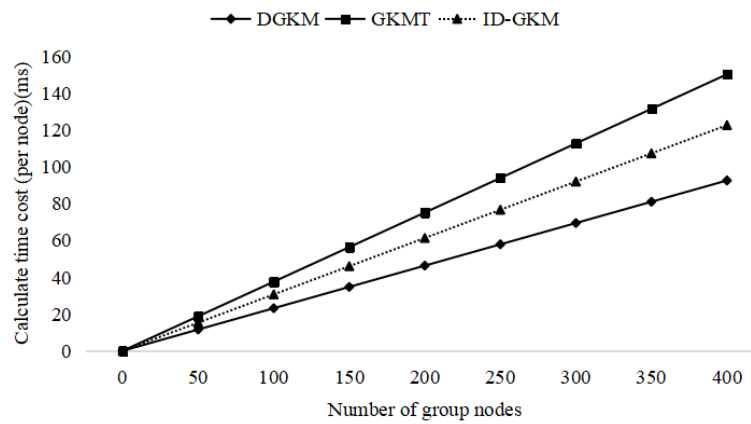


Figure 5. Comparison of the number of group nodes and computational overhead in different schemes.

Communication Cost: To calculate the communication cost, we need to understand the size of information exchanged. In this scenario, we assume a security key length of 160 bits for elliptic curve cryptography. Both protocols employ elliptic curve encryption, and the key length is also 160 bits. Assuming the length of information exchange is 160 bits, the communication overhead refers to the amount of data transmitted during the key re-establishment process. The protocol proposed in this paper considers real-world requirements in its design, aiming to provide more efficient and secure key management services.

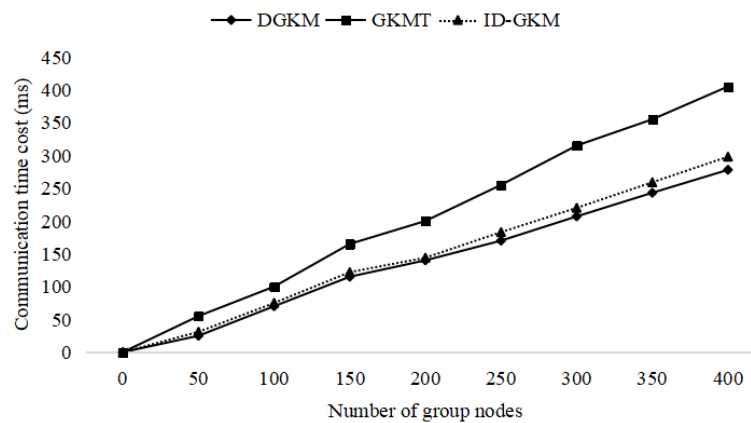


Figure 6. Comparison results of the number of group nodes and communication costs in different schemes.

In modern communication systems, key management is a crucial component. Effective key management ensures the security and confidentiality of communication, preventing information leakage and attacks. The protocol we propose considers not only the security of communication but also its efficiency. By reducing communication overhead, we can improve the overall performance of the system and lower communication costs. This is crucial for enhancing the reliability and efficiency of modern communication systems. Our research results clearly demonstrate significant differences in communication costs among these three group key protocols, as shown in Figure 6. Figure 6 illustrates the communication time as the number of group nodes varies from 0 to 200. Through analysis of the experimental data, we find that our proposed protocol outperforms schemes GKMT and

ID-GKM in terms of performance. This means that our scheme can handle communication tasks more effectively and has better feasibility and reliability in practical applications.

In our proposed scheme, the communication overhead for each group node only includes sending five common parameters and two timestamps, without the need for additional data transmission. Therefore, the communication cost is directly proportional to the parameter length, effectively reducing the overall communication burden of the system, which is one of the advantages of our scheme. Through this approach, we can manage keys more efficiently, ensuring the security and reliability of communication.

Storage overhead: The proposed solution requires storing the following data:

Parameter Set: The KDC requires secure storage of the parameter set $\{G_1, G_2, Q, e, p, P_{pub}, h, E_k, D_K\}$.

Private Key: Each GN possesses a private key SK necessitating secure storage.

Public Key: The public key PK of each GN is stored publicly.

Block: In a blockchain structure, tuples (ID_i, A_i) , (SE_{j+1}, C_j, t_{ij}) , and $R_i = (X_i, Y_i, Z_i, t_{2i})$, accompanied by a timestamp, are stored.

The comparison of the number of group nodes and storage overhead in different solutions is illustrated in Figure 7.

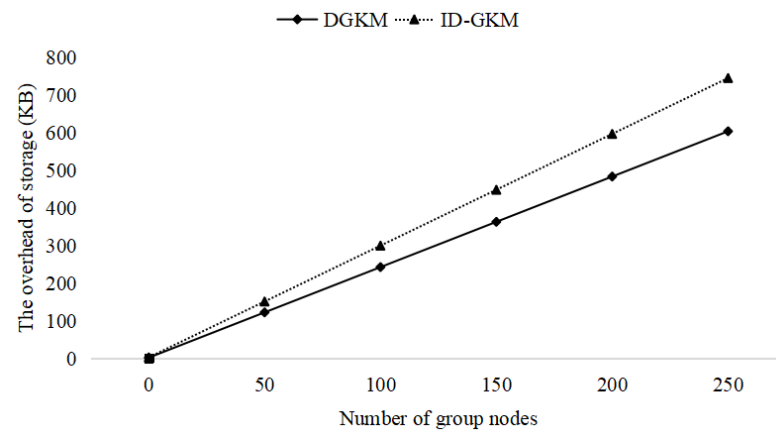


Figure 7. Comparison of the number of group nodes and storage overhead in different schemes.

7. Conclusions

As network environments continue to evolve and system scale grows, the number of required keys also sharply increases. Traditional key management methods are no longer effective in addressing the key management needs in large-scale and complex network environments. To alleviate the burden of key management, this paper introduces a blockchain-based distributed group key management scheme aimed at safeguarding the confidentiality of sensitive records in blockchain networks. The network architecture of distributed group key management is introduced, and a new scheme focusing on optimizing authentication efficiency and communication costs is proposed. By employing intelligent authentication mechanisms and lightweight data update mechanisms, the system's performance is enhanced, and operational costs are reduced. This scheme not only ensures the security and confidentiality of communication but also improves the efficiency and flexibility of the system, providing a novel solution for secure communication in blockchain networks with broad application prospects.

Author Contributions: Conceptualization, J.N.; methodology, J.N.; formal analysis, J.N. and J.R.; investigation, J.N. and J.R.; data curation, J.N.; visualization, Y.Z.; project administration, G.F.; supervision, Y.R.; writing-original draft preparation, J.N.; writing-review and editing, G.F., J.R., L.C. and Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (No. 62072249).

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

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

Abbreviations

The following abbreviations are used in this manuscript:

KDC	Key Distribution Center
AGKA	Asymmetric Group Key Agreement
GKA	Group Key Agreement
DAGKA	Dynamic Distributed Group Key Management
DGKM	Distributed Group Key Management
GN	General Node
ECC	Elliptic Curve Cryptography

References

1. Zheng, J.; Yang, C.; Xue, J.; Zhang, C. A dynamic id-based authenticated group key agreement protocol. In Proceedings of the 2015 4th National Conference on Electrical, Electronics and Computer Engineering, Xi'an, China, 12–13 December 2015; Atlantis Press: Amsterdam, The Netherlands, 2015; pp. 1079–1084.
2. Rafaeli, S.; Hutchison, D. A survey of key management for secure group communication. *ACM Comput. Surv. (CSUR)* **2003**, *35*, 309–329. [[CrossRef](#)]
3. Imine, A.; Fernandez, J.M.; Marion, J.Y.; Logrippo, L.; Garcia-Alfaro, J. *Foundations and Practice of Security*; Springer: Berlin/Heidelberg, Germany, 2018.
4. Xu, Z.; Li, F.; Tan, M.; Zhang, J. A Blockchain-Based Distributed Authentication and Dynamic Group Key Agreement Protocol. In Proceedings of the Blockchain and Trustworthy Systems: Second International Conference, BlockSys 2020, Dali, China, 6–7 August 2020; Revised Selected Papers 2; Springer: Berlin/Heidelberg, Germany, 2020; pp. 142–151.
5. Li, J.; Wu, J.; Chen, L.; Li, J.; Lam, S.K. Blockchain-based secure key management for mobile edge computing. *IEEE Trans. Mob. Comput.* **2021**, *22*, 100–114. [[CrossRef](#)]
6. Sun, L.; Wang, Y.; Ren, Y.; Xia, F. Path Signature-based XAI-enabled Network Time Series Classification. *Sci. China Inf. Sci.* **2024**, *1*–15. [[CrossRef](#)]
7. Barolli, L.; Takizawa, M.; Xhafa, F.; Enokido, T. *Web, Artificial Intelligence and Network Applications: Proceedings of the Workshops of the 33rd International Conference on Advanced Information Networking and Applications (WAINA-2019)*; Springer: Berlin/Heidelberg, Germany, 2019; Volume 927.
8. Seetha, R.; Saravanan, R. A survey on group key management schemes. *Cybern. Inf. Technol.* **2015**, *15*, 3–25. [[CrossRef](#)]
9. Wong, C.K.; Gouda, M.; Lam, S.S. Secure group communications using key graphs. *IEEE/ACM Trans. Netw.* **2000**, *8*, 16–30. [[CrossRef](#)]
10. Islam, S.H.; Obaidat, M.S.; Vijayakumar, P.; Abdulhay, E.; Li, F.; Reddy, M.K.C. A robust and efficient password-based conditional privacy preserving authentication and group-key agreement protocol for VANETs. *Future Gener. Comput. Syst.* **2018**, *84*, 216–227. [[CrossRef](#)]
11. Mitra, S. Iolus: A framework for scalable secure multicasting. *ACM SIGCOMM Comput. Commun. Rev.* **1997**, *27*, 277–288. [[CrossRef](#)]
12. Koussih, S.S.S.; Jajodia, S. Kronos: A Scalable Group Re-Keying Approach for Secure Multicast. In Proceedings of the 2000 IEEE Symposium on Security and Privacy. S&P 2000, Berkeley, CA, USA, 14–17 May 2000; p. 215.
13. Naresh, V.S.; Reddi, S.; Murthy, N.V. A provably secure cluster-based hybrid hierarchical group key agreement for large wireless ad hoc networks. *Hum.-Centric Comput. Inf. Sci.* **2019**, *9*, 26. [[CrossRef](#)]
14. Wang, L.; Tian, Y.; Zhang, D.; Lu, Y. Constant-round authenticated and dynamic group key agreement protocol for D2D group communications. *Inf. Sci.* **2019**, *503*, 61–71. [[CrossRef](#)]
15. Boneh, D.; Lynn, B.; Shacham, H. Short signatures from the Weil pairing. *J. Cryptol.* **2004**, *17*, 297–319. [[CrossRef](#)]
16. Kavitha, S.; Alphonse, P.; Reddy, Y.V. An improved authentication and security on efficient generalized group key agreement using hyper elliptic curve based public key cryptography for IoT health care system. *J. Med. Syst.* **2019**, *43*, 260. [[CrossRef](#)]
17. Zhang, Q.; Wang, R.; Tan, Y. Identity-based authenticated asymmetric group key agreement. *J. Comput. Res. Dev.* **2014**, *51*, 1727–1738.
18. Shi, Y.; Chen, G.; Li, J. ID-based one round authenticated group key agreement protocol with bilinear pairings. In Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05)-Volume II, Las Vegas, NV, USA, 4–6 April 2005; Volume 1, pp. 757–761.
19. Qikun, Z.; Yong, G.; Quanxin, Z.; Ruifang, W.; Yu-An, T. A dynamic and cross-domain authentication asymmetric group key agreement in telemedicine application. *IEEE Access* **2018**, *6*, 24064–24074. [[CrossRef](#)]

20. Gupta, S.; Kumar, A.; Kumar, N. Design of ECC based authenticated group key agreement protocol using self-certified public keys. In Proceedings of the 2018 4th International Conference on Recent Advances in Information Technology (RAIT), Dhanbad, India, 15–17 March 2018; pp. 1–5.
21. Zhao, X.; Wei, D.; Wang, H. Asymmetric group key agreement with traitor traceability. In Proceedings of the 2010 International Conference on Computational Intelligence and Security, Nanning, China, 11–14 December 2010; pp. 347–351.
22. Zhang, L.; Wu, Q.; Domingo-Ferrer, J.; Qin, B.; Dong, Z. Round-efficient and sender-unrestricted dynamic group key agreement protocol for secure group communications. *IEEE Trans. Inf. Forensics Secur.* **2015**, *10*, 2352–2364. [[CrossRef](#)]
23. Zhang, L.; Wu, Q.; Qin, B.; Domingo-Ferrer, J. Identity-based authenticated asymmetric group key agreement protocol. In Proceedings of the Computing and Combinatorics: 16th Annual International Conference, COCOON 2010, Nha Trang, Vietnam, 19–21 July 2010; Proceedings 16; Springer: Berlin/Heidelberg, Germany, 2010; pp. 510–519.
24. Zhang, L.; Meng, X.; Choo, K.K.R.; Zhang, Y.; Dai, F. Privacy-preserving cloud establishment and data dissemination scheme for vehicular cloud. *IEEE Trans. Dependable Secur. Comput.* **2018**, *17*, 634–647. [[CrossRef](#)]
25. Salman, T.; Zolanvari, M.; Erbad, A.; Jain, R.; Samaka, M. Security services using blockchains: A state of the art survey. *IEEE Commun. Surv. Tutor.* **2018**, *21*, 858–880. [[CrossRef](#)]
26. Gan, S. *An IoT Simulator in NS3 and a Key Based Authentication Architecture for IoT Devices using Blockchain*; Indian Institute of Technology: Kanpur, India, 2017.
27. Matsumoto, S.; Reischuk, R. Turning a PKI around with decentralized automated incentives. In Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2017.
28. Balenson, D.; McGrew, D.; Sherman, A. *Key Management for Large Dynamic Groups: One-Way Function Trees and Amortized Initialization*; Technical report; IETF Internet Draft: Washington, DC, USA, 1999.
29. Lei, A.; Cruickshank, H.; Cao, Y.; Asuquo, P.; Ogah, C.P.A.; Sun, Z. Blockchain-based dynamic key management for heterogeneous intelligent transportation systems. *IEEE Internet Things J.* **2017**, *4*, 1832–1843. [[CrossRef](#)]
30. Crosby, M.; Pattanayak, P.; Verma, S.; Kalyanaraman, V.; et al. Blockchain technology: Beyond bitcoin. *Appl. Innov.* **2016**, *2*, 71.
31. Billings, J. Image-Based Proof of Work Algorithm for the Incentivization of Blockchain Archival of Interesting Images. *arXiv* **2017**, arXiv:1707.04558.
32. Ren, Y.; Leng, Y.; Qi, J.; Sharma, P.K.; Wang, J.; Almkhadmeh, Z.; Tolba, A. Multiple cloud storage mechanism based on blockchain in smart homes. *Future Gener. Comput. Syst.* **2021**, *115*, 304–313. [[CrossRef](#)]
33. Park, J.H.; Park, J.H. Blockchain security in cloud computing: Use cases, challenges, and solutions. *Symmetry* **2017**, *9*, 164. [[CrossRef](#)]
34. Ren, Y.; Leng, Y.; Cheng, Y.; Wang, J. Secure data storage based on blockchain and coding in edge computing. *Math. Biosci. Eng.* **2019**, *16*, 1874–1892. [[CrossRef](#)]
35. Huckle, S.; Bhattacharya, R.; White, M.; Beloff, N. Internet of things, blockchain and shared economy applications. *Procedia Comput. Sci.* **2016**, *98*, 461–466. [[CrossRef](#)]
36. Fang, G.; Sun, Y.; Almutiq, M.; Zhou, W.; Zhao, Y.; Ren, Y. Distributed Medical Data Storage Mechanism Based on Proof of Retrievability and Vector Commitment for Metaverse Services. *IEEE J. Biomed. Health Inform.* **2023**, 1–9. [[CrossRef](#)]
37. Lin, I.C.; Liao, T.C. A survey of blockchain security issues and challenges. *Int. J. Netw. Secur.* **2017**, *19*, 653–659.
38. Ren, Y.; Lv, Z.; Xiong, N.N.; Wang, J. HCNCT: A Cross-chain Interaction Scheme for the Blockchain-based Metaverse. *ACM Trans. Multimed. Comput. Commun. Appl.* **2023**, *20*, 188. [[CrossRef](#)]
39. Yang, B. *Provable Security in Cryptography*; Tsinghua University Press: Beijing, China, 2017.
40. Wang, Z.; Zhang, C.; Wei, L. An adapter signature scheme based on bilinear pairing. *J. Cryptologic Res.* **2022**, *9*, 686.
41. Shahidinejad, A.; Abawajy, J. An All-Inclusive Taxonomy and Critical Review of Blockchain-Assisted Authentication and Session Key Generation Protocols for IoT. *ACM Comput. Surv.* **2024**, *56*, 186. [[CrossRef](#)]
42. Wu, Q.; Mu, Y.; Susilo, W.; Qin, B.; Domingo-Ferrer, J. Asymmetric group key agreement. In Proceedings of the Advances in Cryptology-EUROCRYPT 2009: 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, 26–30 April 2009; Proceedings 28; Springer: Berlin/Heidelberg, Germany, 2009; pp. 153–170.
43. Sudheeradh, K.; Jahnavi, N.N.; Chine, P.N.; Kasbekar, G.S. Efficient and Secure Group Key Management Scheme Based on Factorial Trees for Dynamic IoT Settings. *IEEE Access* **2024**, *12*, 5659–5671. [[CrossRef](#)]

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.