Journal of King Saud University Computer and Information Sciences (2025) 37:57
https://doi.org/10.1007/s44443-025-00057-8

ORIGINAL PAPER

Asynchronous distribution scheme of group quantum keys based on max heap in classical network

Dexin Zhu[1,2] · Zilong Zhao[2] · Huanjie Zhang[2] · Zhiqing Zhou[2] · Yuanbo Li[2] · Jian Zhao[2] · Lijun Song[3] · Jun Zheng[1]

Received: 22 February 2025 / Accepted: 7 May 2025

Abstract
With the advancement of the information age, various information-based products have become increasingly prevalent, leading to a corresponding rise in the volume of data requiring encrypted transmission. Consequently, secure key distribution has become particularly critical, especially in large-scale communication scenarios. Considering practical limitations such as offline availability and low performance among communication participants, we propose an asynchronous group quantum key distribution scheme based on a max heap structure and blockchain technology within classical networks. In this scheme, quantum keys serve as group communication keys, and the encrypted transmission of quantum keys is facilitated through a max heap constructed according to user information. Specifically, during the initialization phase, users upload relevant information to the Key Distribution Center (KDC), while in the key distribution phase, the blockchain stores encrypted temporary group keys, enabling asynchronous access. The proposed scheme not only reduces performance consumption for users within the group but also leverages the intrinsic properties of the max heap to dynamically balance loads. Moreover, the scheme achieves both forward and backward secrecy and demonstrates favorable performance in terms of storage efficiency and computational cost.

Keywords Blockchain · Group key distribution · Max heap · Quantum key

1 Introduction

With the rapid advancement of internet technologies, an increasing amount of information exchange is being conducted electronically. Ensuring the secure transmission of such information has thus become an important challenge. In multi-user environments, such as video conferences, social media group chats, and Internet of Things (IoT) scenarios, the transmitted information often passes through insecure public channels, enabling attackers to eavesdrop and intercept messages sent and received by users. This scenario can result in the leakage of private user data, potentially causing significant harm. Therefore, effective measures must be implemented to secure message transmission. A common approach to addressing this issue is encrypting messages before transmission. However, a critical prerequisite for such encryption is ensuring that all participating users possess a consistent shared session key (Houzhen et al. 2023).

Quantum Key Distribution (QKD) (Lo et al. 2014; Pirandola et al. 2020) is a secure key-exchange technique that leverages the principles of quantum mechanics, enabling two remote communication parties to generate and share a random, secure key over an insecure communication channel. This key can subsequently be utilized for message encryption and decryption. Nevertheless, practical implementations of QKD currently face several challenges, including imperfections in photon sources, finite-key effects, and hardware limitations. Consequently, QKD technology remains primarily in the scientific research phase rather than the stage of commercial deployment, indicating that numerous technical and theoretical aspects still require verification and optimization.

✉ Dexin Zhu
3220215219@bit.edu.cn

[1] School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China

[2] College of Computer Science and Technology, Changchun University, Changchun 130022, Jilin, China

[3] Jilin Engineering Laboratory for Quantum Information Technology, Changchun Institute Of Technology, Changchun 130022, Jilin, China

Published online: 05 June 2025

In comparison, classical approaches for distributing quantum keys presently exhibit significant advantages: First, classical methods rely on well-established mathematical theories and standardized protocols, thus avoiding the need for complex physical devices for quantum state manipulation. This leads to lower deployment costs and better compatibility with existing communication infrastructures. Second, classical key-distribution processes demonstrate higher tolerance to fluctuations in hardware parameters, such as wavelength stability of the photon source and detector efficiency (Diamanti and Leverrier 2015), thereby circumventing limitations associated with fiber-optic attenuation and environmental disturbances prevalent in quantum channels. Third, by integrating post-quantum cryptographic algorithms (Alléaume et al. 2014), classical methods can achieve forward secrecy and adequately meet the practical requirements of short- to medium-distance communication scenarios, especially those involving resource-constrained environments such as IoT terminals. This complementarity implies that, until quantum technology matures sufficiently, the coordinated development of classical and quantum key-distribution systems will serve as an essential strategy for building a multi-layered security framework.

In group-oriented services, two primary approaches exist to achieve the sharing of group keys: group key distribution and group key agreement. In group key agreement schemes, a set of entities cooperatively generates the group key. In contrast, group key distribution schemes involve a trusted third party, known as the group manager, who determines the group key and securely distributes it to authorized entities. For scenarios where infrastructure or semi-infrastructure environments are available, group key distribution schemes are generally preferred due to their efficiency, as well as ease of deployment and maintenance (Chien 2021).

In traditional group key distribution schemes, a KDC is typically employed to generate and distribute group keys. However, this approach inevitably introduces the problem of a single point of failure, reducing the robustness of the system. Moreover, existing schemes generally rely on pseudo-random numbers rather than true randomness for generating group keys, posing potential security risks (Cheng et al. 2024). Furthermore, most current group key distribution schemes require all participants to remain continuously online. Nevertheless, in practical scenarios, many battery-powered devices enter sleep mode periodically to extend their operational lifetimes, making continuous online availability challenging to achieve (Rahimi and Chrysostomou 2019). Thus, designing asynchronous group key distribution schemes that accommodate intermittent participant availability remains a challenging research issue. Blockchain technology, an emerging distributed and tamper-resistant data storage approach, has been widely adopted across various domains such as cloud computing and smart grids (Li et al. 2019). It presents a novel solution path for implementing asynchronous group key distribution.

To address the aforementioned challenges, this paper proposes a group quantum key distribution scheme based on blockchain technology and a max heap structure. In our proposed scheme, the KDC is solely responsible for generating temporary group keys, thereby significantly reducing the impact of potential KDC failures on the overall system. Pre-stored quantum keys in the Quantum Key Cloud Server (QKCS) serve as the group keys, while blockchain technology is utilized for secure data storage and transmission, ensuring the system's distributed nature and traceability. Users can asynchronously extract or recover the group quantum key at any time according to their needs. The primary contributions of this paper are as follows:

(1) We propose a temporary group key generation method based on a max heap structure, which allows efficient key updates through simple shift operations. In scenarios where the KDC's performance is insufficient to support a large number of group members concurrently, we utilize node identifiers from the max heap structure as indicators of user capability. Thus, users with higher performance can dynamically share the workload of the key distribution process, effectively balancing the load on the system.
(2) The proposed scheme achieves asynchronous operation. Specifically, after users upload their information to the KDC, they can transition to an offline state. Upon reconnecting, users only need to retrieve the temporary group key from the blockchain to decrypt the group quantum key ciphertext stored in the Quantum Key Cloud Server. Additionally, if users lose their local copy of the group quantum key, they can seamlessly recover the identical key upon reconnecting.
(3) Our scheme minimizes the computational burden on users by primarily employing lightweight XOR operations, significantly reducing the computational cost for resource-constrained participants.

The remainder of this paper is organized as follows. In Section 2, we provide a comprehensive analysis of existing solutions presented in the literature. Section 3 introduces the necessary background knowledge, followed by our notation definitions, system model and threat model presented in Section 4. In Section 5, we describe the details of our proposed scheme. The correctness, security, and security properties of the proposed scheme are analyzed thoroughly in Section 6. Section 7 presents the performance evaluation of our scheme. Finally, we conclude the paper in Section 8.

2 Related works

Group key schemes can generally be classified into two main categories: group key agreement schemes and group key distribution schemes. Group key agreement schemes involve cooperation among all group communication participants to jointly establish a shared group key, independently of any trusted central authority. The group key is collectively determined by group members themselves (Chen et al. 2021). In 2020, Gan et al. (2021) proposed an attribute-based asymmetric group key agreement protocol, in which users are not required to reveal personal information within the group. Instead, they need only satisfy the same number of attributes and perform mutual authentication with the key generation center to ensure legitimacy in mobile-terminal participation. Rawat and Deshmukh (2020) presented a Group Key Agreement Protocol (GKAP) based on tree structures and elliptic curves. Their protocol employs a divide-and-conquer approach, splitting groups into smaller subgroups organized as tree structures. This method exhibits resistance to passive attacks, collaborative attacks, and man-in-the-middle attacks, and ensures both forward and backward secrecy. In 2022, Braeken (2022) proposed a single-round, lightweight elliptic curve-based alternative using Qu-Vanstone elliptic curve certificates, allowing group members to authenticate themselves independently. Naresh et al. (2022) introduced a blockchain-based two-party elliptic curve Diffie-Hellman key agreement protocol, subsequently extending their approach to an n-party key agreement scheme. They presented a blockchain-based dynamic authenticated group key agreement protocol, where a Privacy-Preserving Smart Contract (PPSC) acts as a group controller in the first round, generating pairwise shared keys with each member. In the second round, the PPSC calculates partial group keys and distributes them to respective members. Upon receiving their partial group keys, group members multiply the received product with their own shared keys to derive the final group key. Wang et al. (2022a) proposed a blockchain-based lightweight, computationally efficient, and secure key management method for smart grids. They redesigned the blockchain-enabled smart grid architecture by clearly defining various entity roles and developed a pairing-free authenticated group key agreement scheme. Xu et al. (2022) proposed an anonymous authentication and dynamic group key agreement scheme based on blockchain technology and token mechanisms. Each member can apply for a time-sensitive token during initial authentication, and subsequent authentications require only verification of token validity, significantly reducing computational and communication costs. Wang et al. (2022b) proposed a two-round dynamic authenticated group key agreement protocol based on LWE, proving its security under standard attack models. In 2023, Zhang et al. (2023a) introduced a single-round dynamic authenticated asymmetric group key agreement protocol with sender non-repudiation and privacy. They demonstrated its security against chosen-ciphertext attacks under the assumption that the computational Diffie-Hellman and k-bilinear Diffie-Hellman exponent problems are computationally infeasible. Chhikara et al. (2023) developed a novel blockchain-based group key agreement protocol that RSU as semi-trusted entities, substantially reducing processing delays, queuing delays, and deployment costs. Zhang et al. (2023b) proposed a threshold authenticated group key agreement (TAGKA) protocol capable of handling group member disconnections. Considering the temporary disconnection of drones, they further designed a key recovery protocol that allows temporarily disconnected drones to retrieve the updated group key once their communication links are restored. Yang et al. (2023) proposed a dynamic group key agreement scheme based on short signatures (GKA-SS), utilizing bilinear pairings to execute signature and verification processes. Formal security proofs showed the scheme's resistance to both active attacks within the random oracle model and passive attacks. In 2024, Cui et al. (2024) introduced a C-V2X-based dynamic group authentication and key agreement protocol named V2X-GKA. This protocol leverages cryptographic techniques, such as ECDL and DBDH, to effectively mitigate risks associated with certificate forgery and key theft. By integrating authentication and group key agreement mechanisms, the protocol facilitates dynamic member management and secure key updates without requiring complete re-execution of the protocol.

Group key distribution schemes involve a trusted third party, known as the Group Key Manager (GKM), which determines the group key and securely distributes it to all group members (Xiong et al. 2019). Yıldız et al. (2021) proposed a lightweight group authentication and key distribution (PLGAKD) protocol based on Physical Unclonable Functions (PUF), factorial trees, and the Chinese Remainder Theorem (CRT). In the PLGAKD protocol, PUF provides lightweight identity authentication and key distribution among group members, while factorial trees and CRT help reduce both the number of keys stored in nodes and the volume of transmitted messages during key updates. Harn et al. (2021) introduced a novel key distribution scheme wherein the basic key distribution protocol requires only logical XOR operations. Kumar et al. (2020) proposed an efficient centralized group key distribution protocol designed to minimize the computational cost of the key server during key updates. Additionally, they presented an extended clustering-tree-based CGKD protocol that balances member computational cost during key recovery, exhibiting excellent scalability and effectively accommodating significant fluctuations in group membership. Hougaard and Miyaji (2022) presented the first SIDH-based constant-round tree-type group key exchange protocol, achieving linear communication and storage complexity. Taurshia et al. (2022) proposed a novel Group Key Management scheme for Low Resource Devices (GKM-LRD), employing an SDN-assisted trusted key management server as a central entity providing key management services for groups in IoT applications. Harn et al. (2022) proposed the first lightweight authenticated group key distribution scheme relying solely on logical operations. Their scheme can be constructed upon any existing authenticated pairwise key distribution protocol. Gebremichael et al. (2022) introduced a lattice-based one-way function that can be inverted via appropriately designed lattice trapdoors. Leveraging the concept of "bad/good" lattice bases, they proposed a novel approach to couple multiple private keys into a single public key, subsequently employing this public key to encrypt group messages. The scheme demonstrates clear advantages, notably speculative resistance against potential quantum computing-based attacks. Luo et al. (2023) presented an ITS group authentication scheme that requires fewer keys in QKD networks. In this protocol, multiple QKD network nodes collaborate as a group to achieve authentication. Abdmeziem et al. (2024) proposed an asynchronous ratcheting tree (ART) implementation based on blockchain technology. They introduced a novel asynchronous certificate creation method that leverages smart contracts to realize the distributed nature of blockchain, additionally incorporating a reputation mechanism to address the heterogeneity of resource-constrained IoT devices.

3 Preliminaries

3.1 Max heap

A heap is a specialized form of complete binary tree commonly used in computer science. By definition, a heap is a complete binary tree in which all levels are fully filled except possibly the last, which is filled from left to right as completely as possible. A heap in which the root node contains the maximum value among all nodes is referred to as a max heap. Let $k(i)$ denote the value of the i-th node in the max heap. In this context, $k(1)$ represents the value of the root node. The max heap property ensures that the values satisfy the conditions $k(i) \geq k(2i)$ and $k(i) \geq k(2i+1)$ for all applicable indices i.

3.2 Blockchain

Blockchain is a decentralized and distributed ledger system that operates over a peer-to-peer network. It enables secure storage and verification of transactions without the need for a centralized authority (García et al. 2024). Fundamentally, blockchain is a distributed data structure, conceptually similar to a linked list, composed of a sequence of blocks (or nodes). Each block contains a field that stores the hash value of the preceding block, thereby forming a cryptographic link between blocks (Kemmoe et al. 2023). Blockchain is characterized by decentralization, transparency, security, and immutability. It employs advanced cryptographic algorithms to ensure data privacy and integrity, making it highly resistant to unauthorized access. In addition, blockchain supports smart contracts-self-executing programs whose instructions are stored on the blockchain and are publicly accessible.

3.3 Difficulty problem

Let $E(\mathbb{F}_p)$ be a cyclic elliptic curve group over a finite field $\mathbb{F}_p$, with a base point $g \in \mathbb{F}_p$ of prime order n.

Elliptic Curve Computational Diffie-Hellman (ECCDH) Problem. Let $a, b \in \mathbb{F}_p$ be randomly selected scalars. Given the tuple $(g, [a]g, [b]g)$, it is computationally infeasible to determine $[ab]g$ without knowledge of both a and b.

4 System model

4.1 Notation

To facilitate the performance analysis of the proposed scheme, we first introduce several essential notations. The definitions of the mathematical notations used throughout this paper are summarized in Table 1.

Table 1 Mathematical notations definition

Notations	Definitions
$params$	Parameters published by system initialization
GID_i	Group Identifier
U_i	user i
ID_i	user i's identity information
T	Timestamp
Δt	Time gap
h	Hash Function
K_i, K_s, K_q	public key of user i, KDC and QKCS
k_i, s, q	private key of user i, KDC and QKCS
$\|, \oplus$	Concatenation, XOR operation
CH_i, MAC_{k_i}	Group Authentication Parameters
QH	QKCS obtains data identification on blockchain
sk_i^{ku}, sk^{kq}	The shared key between the KDC and user i, the shared key between the KDC and the QKCS

4.2 System model

The system consists of four entities: (1) users; (2) KDC; (3) blockchain; (4) quantum key cloud server. The system model is illustrated in Fig. 1.

(1) Users: Users are the entities participating in secure group communication. Each user submit his identity and public key information to the KDC and subsequently retrieves the temporary group key from the blockchain. This temporary group key is then used to decrypt the group quantum key ciphertext stored on the QKCS, thereby enabling secure group communication.
(2) KDC: The KDC is responsible for authenticating user identities and generating a temporary group key based on identity information, public keys, and other relevant parameters. The KDC then applies an XOR operation to the temporary group key and uploads the encrypted result to the blockchain, where it can be accessed by users and the QKCS.
(3) Blockchain (BS): The BS is used to store encrypted temporary group keys. It executes smart contracts, and based on the parameters provided by users, returns the corresponding ciphertext of the temporary group key.
(4) QKCS: The QKCS stores the group quantum key generated by the QKD network. It encrypts the group quantum key using the temporary group key provided by the KDC and makes it available for secure retrieval by authorized users.

4.3 Threat model

This subsection outlines the rational assumptions underlying the proposed scheme and defines the adversary's attack capabilities.

Following the attack model presented in García et al. (2024), we assume that the adversary is capable of eavesdropping on and tampering with all messages transmitted over insecure public channels. Therefore, the scheme must ensure that the protocol operates correctly even if the adversary

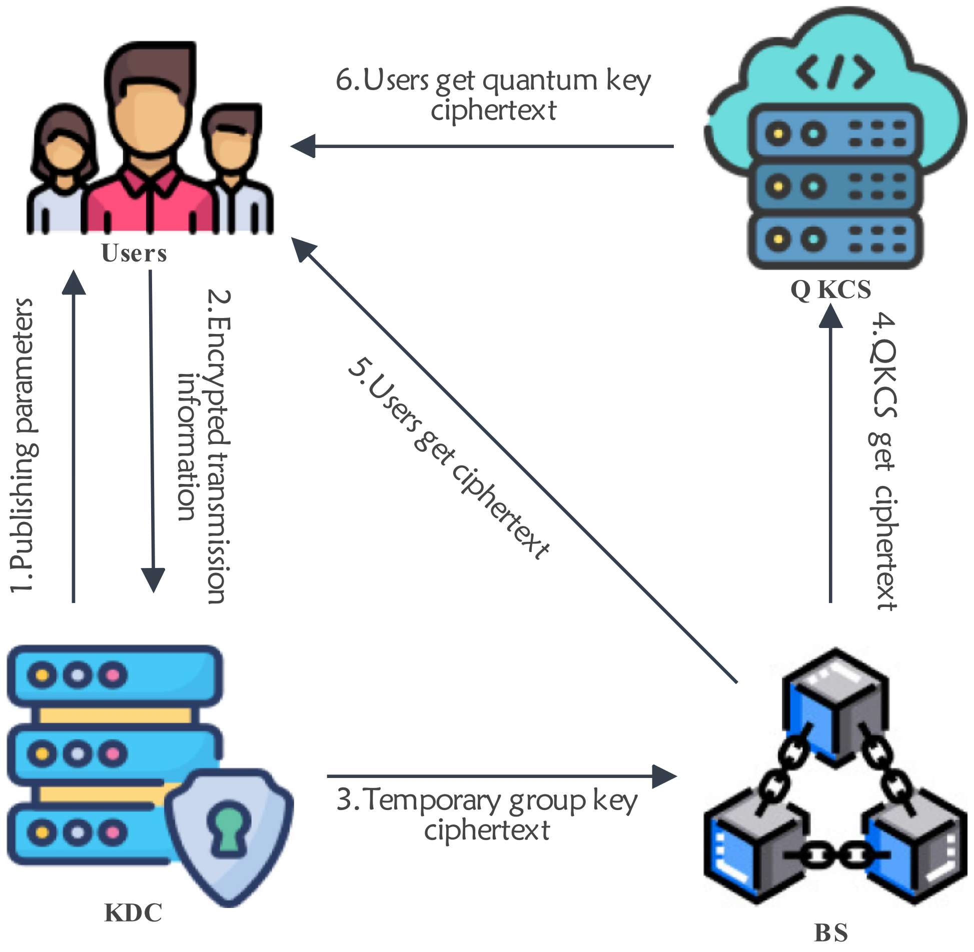
6.Users get quantum key ciphertext
Users
Q KCS
1.Publishing parameters
2.Encrypted transmission information
5.Users get ciphertext
4.QKCS get ciphertext
3.Temporary group key ciphertext
KDC
BS

Fig. 1 System model

obtains such information, and more importantly, that no secret data can be disclosed. In addition, we make the following reasonable assumptions about the system:

(1) Malicious nodes may exist within the group and attempt to impersonate legitimate members. To address this, the scheme is designed so that nodes participate in the group key generation process and verify the legitimacy of other group members using verification information stored on the BS by the KDC. This mechanism effectively prevents forgery.
(2) The KDC and the QKCS are assumed to be fully trusted entities. The KDC is responsible for constructing, but not extracting the temporary group key, while the QKCS can encrypt and transmit the group quantum key but similarly cannot extract it.

5 Specific plan

5.1 System initialization

We denote the n users as $U = \{U_1, U_2, \cdots, U_n\}$, where user i is represented by U_i. Let $ID = \{ID_1, ID_2, \cdots, ID_n\}$ represent the identity information of the users, where ID_i corresponds to the identity of user i. Similarly, let $GID = \{GID_1, GID_2, \cdots, GID_n\}$ denote the group identifiers, with GID_j indicating the identifier for group j. When user i joins group GID_j, his group-specific identity is denoted by ID_i^{GIDj}. The QKCS possesses a private key q and a corresponding public key $K_q = [q]g$. Prior to any user joining a group, the KDC is responsible for pre-storing the users' identity set ID, the corresponding group identifiers GID, and the public key K_q of the QKCS to facilitate group user authentication and key distribution. It is assumed that n users $U_i (1 \leq i \leq n)$ intend to join group GID_j. Figure 2 outlines the overall process from initialization to the key distribution phase. The detailed steps are described below.

(1) During this phase, the KDC executes the following procedures:

Step 1: A large prime number p is selected at random, and an elliptic curve E defined by the equation $y^2 \equiv x^3 + ax + b \bmod p$ is adopted, where $a, b \in \mathbb{F}_p$ and $\mathbb{F}_p$ denotes a finite field of order p, with the discriminant $\Delta = 4a^3 + 27b^2 \neq 0$. Let G be a cyclic additive group over E, and designate a generator g of G.

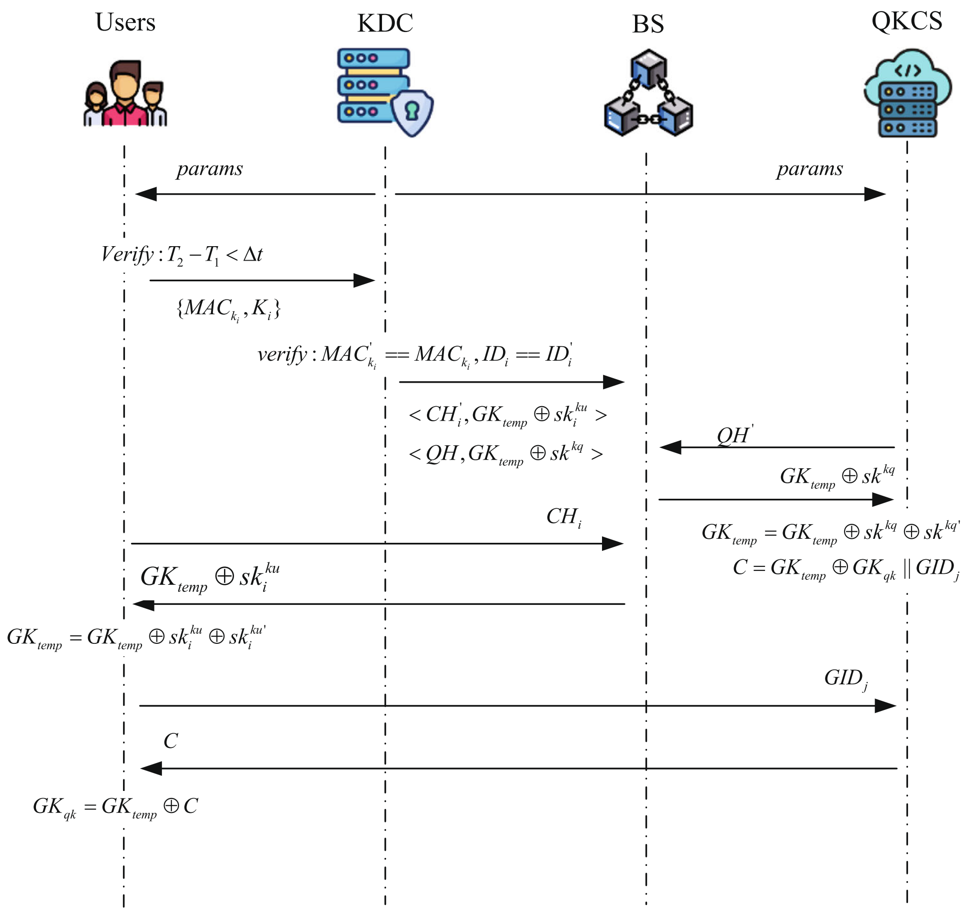
Users
KDC
BS
QKCS
params
params
Verify : $T_2 - T_1 < \Delta t$
$\{MAC_{k_i}, K_i\}$
verify : $MAC'_{k_i} == MAC_{k_i}, ID_i == ID'_i$
$< CH'_i, GK_{temp} \oplus sk_i^{ku} >$
$< QH, GK_{temp} \oplus sk^{kq} >$
QH'
$GK_{temp} \oplus sk^{kq}$
CH_i
$GK_{temp} = GK_{temp} \oplus sk^{kq} \oplus sk^{kq'}$
$C = GK_{temp} \oplus GK_{qk} || GID_j$
$GK_{temp} \oplus sk_i^{ku}$
$GK_{temp} = GK_{temp} \oplus sk_i^{ku} \oplus sk_i^{ku'}$
GID_j
C
$GK_{qk} = GK_{temp} \oplus C$

Fig. 2 Initialization and key distribution

Step 2: A time interval Δt is specified, within which users must complete the group joining process. Requests exceeding this interval are considered invalid.

Step 3: A random number s is selected as the KDC's private key, satisfying $0 < s < p$, and its public key is computed as $K_s = [s]g$.

Step 4: A timestamp T_1 is generated, and a secure hash function $h : \{0, 1\}^* \rightarrow \{0, 1\}^l$ is chosen, where l denotes the output length. The system parameters are then published as $params = \left\{G, g, p, K_s, \Delta t, T_1, GID_j, h\right\}$.

(2) Each user $U_i, (1 \leq i \leq n)$ within group GID_j must deliver their identity ID_i and public key K_i to the KDC within the designated time window. The actions required from each user are as follows:

Step 1: The user obtains the current timestamp T_2 and confirms that $T_2 - T_1 < \Delta t$. If this condition holds, the process continues; otherwise, it terminates.

Step 2: A private key k_i is selected such that $0 < k_i < p$, and the corresponding public key is computed as $K_i = [k_i]g$.

Step 3: By combining their identity ID_i and the group identifier GID_j, the user derives $CH_i = h\left(ID_i \| GID_j\right)$, and then determines $MAC_{k_i} = CH_i \oplus ID_i$. The value CH_i serves as a locator for retrieving necessary parameters on the BS to reconstruct the temporary group key, while MAC_{k_i} supports the KDC in authenticating group membership. The tuple $\{MAC_{k_i}, K_i\}$ is then forwarded to the KDC.

Once the information is submitted, the user can asynchronously obtain the group quantum key without the need to remain continuously online. When access to the group quantum key is required, the user can fetch the encrypted data from both the BS and the QKCS, and subsequently recover the group quantum key through decryption.

5.2 Key distribution phase

In this stage, the KDC engages in lightweight operations to authenticate group members and prepare the temporary group key. The key is derived through the combination of identity information, public credentials, and cryptographic parameters, then concealed via XOR operations and recorded on the BS. Group members retrieve this encrypted material through predetermined access values and reconstruct the temporary group key.

(1) The KDC undertakes the following procedures:

Step 1: The current timestamp T_3 is acquired, and the condition $T_3 - T_1 < \Delta t$ is checked. If satisfied, the process advances.

Step 2: Given each input pair $\{MAC_{k_i}, K_i\}$, the KDC reconstructs $CH_i^{'} = h\left(ID_i \| GID_j\right)$, calculates $MAC_{k_i}^{'} = CH_i^{'} \oplus ID_i$, and restores $ID_i^{'} = CH_i^{'} \oplus MAC_{k_i}$. Verification proceeds only if both $MAC_{k_i}^{'} == MAC_{k_i}$ and $ID_i == ID_i^{'}$ are confirmed.

Step 3: Upon successful validation, a random index d_i satisfying $0 < d_i < p$ is assigned to each user U_i, and all indices are organized into a max heap structure. Define a list L indexed by GID_j, storing entries $[d_i, IKC_i]$, where $IKC_i = \{ID_i^{GID_j}, K_i, CH_i^{'}\}$. For example,

$$GID_j = \{[10, IKC_1], [8, IKC_2], [6, IKC_3], [4, IKC_3], [1, IKC_5]\}$$

complies with the max heap logic (illustrated in Fig. 3).

Step 4: For all $1 \leq i \leq n$, calculate $node_i = h\left([d_i]K_i\right) \oplus ID_i^{GID_j} \| [d_i]g$. The temporary group key is aggregated by the expression: $GK_{temp} = node_1 \oplus node_2 \oplus \cdots \oplus node_n$.

Step 5: Derive the shared secrets $sk_i^{ku} = [s]K_i$ and $sk^{kq} = [s]K_q$, respectively. Construct the hash value $QH = h\left(K_q \| sk^{kq} \| GID_j\right)$. Let $CH_i^{'}$ and QH serve as BS indexing keys, and post the encrypted entries: $< CH_i^{'}, GK_{temp} \oplus sk_i^{ku} >, < QH, GK_{temp} \oplus sk^{kq} >$.

(2) The QKCS proceeds as follows:

Step 1: Determine $sk^{kq^{'}} = [q]K_s$, then compute $QH^{'} = h\left(K_q \| sk^{kq^{'}} \| GID_j\right)$.

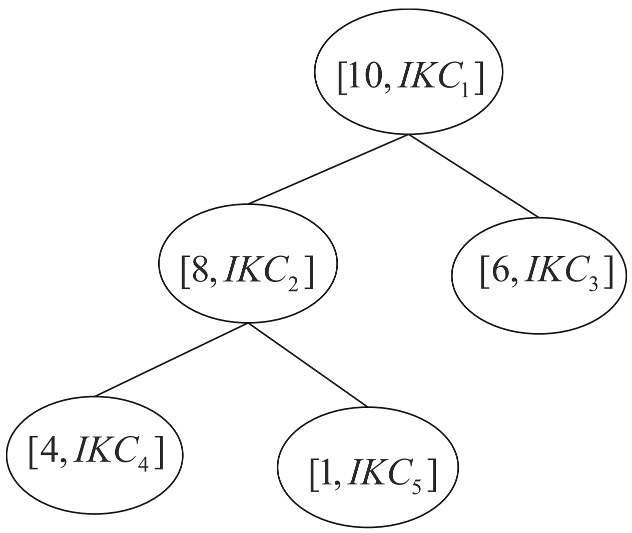

Fig. 3 Logical structure of 5 users in group GID_j

Step 2: With $QH^{'}$ as a reference key, extract $GK_{temp} \oplus sk^{kq}$ from the BS and derive the temporary key by: $GK_{temp} = GK_{temp} \oplus sk^{kq} \oplus sk^{kq^{'}}$.

Step 3: Retrieve a group quantum key GK_{qk}, and produce the ciphertext $C = GK_{temp} \oplus GK_{qk} \| GID_j$.

(3) Each user follows these steps:

Step 1: Compute the shared key $sk_i^{ku^{'}} = [k_i] K_s$.

Step 2: Referencing CH_i as an index, recover $GK_{temp} \oplus sk_i^{ku}$ from the BS and calculate: $GK_{temp} = GK_{temp} \oplus sk_i^{ku} \oplus sk_i^{ku^{'}}$.

Step 3: Retrieve C from the QKDS using GID_j and determine: $GK_{qk} = GK_{temp} \oplus C$.

5.3 Single user join

When a new user U_{n+1} intends to join group GID_j, the KDC must first register the user's identity ID_{n+1} and generate a current timestamp T_4. The user U_{n+1} transmits his identity ID_{n+1} and public key K_{n+1} to the KDC.

(1) The user completes the following steps:

Step 1: Randomly select a private key k_{n+1} such that $0 < k_{n+1} < p$, and compute the corresponding public key $K_{n+1} = [k_{n+1}]g$.

Step 2: Following the same procedure as in Section 5.1 (2) Step 2, compute $CH_{n+1} = h\left(ID_{n+1} \| GID_j\right)$ and $MAC_{k_{n+1}} = CH_{n+1} \oplus ID_{n+1}$. Submit the tuple $\left\{MAC_{k_{n+1}}, K_{n+1}\right\}$ to the KDC.

(2) The KDC performs the following procedures:

Step 1: Acquire the current timestamp T_5 and ensure that $T_5 - T_4 < \Delta t$ holds.

Step 2: Compute $CH_{n+1}^{'} = h\left(ID_{n+1} \| GID_j\right)$, $MAC_{k_{n+1}}^{'} = CH_{n+1}^{'} \oplus ID_{n+1}$, and $ID_{n+1}^{'} = CH_{n+1}^{'} \oplus MAC_{k_{n+1}}$. Verify whether $MAC_{k_{n+1}}^{'} == MAC_{k_{n+1}}$ and $ID_{n+1} == ID_{n+1}^{'}$.

Step 3: Upon validation, the KDC handles the identity $ID_{n+1}^{GID_j}$ and public key K_{n+1} of user U_{n+1} based on the authentication results, and assigns a new index d_{n+1} with $0 < d_{n+1} < p$. Following the update logic of a max heap, the user is inserted into the list L under group GID_j. Suppose U_{n+1} is placed at position m, the group entry becomes: $GID_j = \{[d_1, IKC_1], \cdots, [d_{m-1}, IKC_{m-1}] [d_{n+1}, IKC_{n+1}], [d_m, IKC_m], \cdots, [d_n, IKC_n]\}$. As an illustrative example, if the group previously holds:

$$GID_j = \{[10, IKC_1], [8, IKC_2], [6, IKC_3], [4, IKC_3], [1, IKC_5]\},$$

and a new user is inserted with index 7, the updated structure (as per Fig. 4(a) and (b)) becomes:

$$GID_j = \{[10, IKC_1], [8, IKC_2][7, IKC_7], [4, IKC_4], [1, IKC_5], [6, IKC_3]\}.$$

Step 4: Update the temporary group key. Compute $node_{n+1} = h\left([d_{n+1}]K_{n+1}\right) \oplus ID_{n+1}^{GIDj} \| [d_{n+1}]g$, and derive $sk_{n+1}^{ku} = [s]K_{n+1}$. Then calculate the updated temporary group key: $GK_{temp}^{'} = GK_{temp} \oplus node_{n+1} \oplus d_{n+1}$. Finally, update the BS with the new key mappings: $< CH_i^{'}, GK_{temp}^{'} \oplus sk_i^{ku} >$, $< QH, GK_{temp}^{'} \oplus sk^{kq} >$, $(1 \le i \le n+1)$.

(3) The QKCS updates the group quantum key as follows:

Step 1: Retrieve $GK_{temp}^{'} \oplus sk^{kq}$ from the BS via $QH^{'}$, then recover the updated key: $GK_{temp}^{'} = GK_{temp}^{'} \oplus sk^{kq} \oplus sk^{kq^{'}}$.

Step 2: Select a new group quantum key $GK_{qk}^{'}$, and compute the ciphertext: $C^{'} = GK_{temp}^{'} \oplus GK_{qk}^{'} \| GID_j$.

(4) The newly joined user U_{n+1} obtains the group quantum key as follows:

Step 1: Compute $sk_{n+1}^{ku^{'}} = \left[k_{n+1}\right] K_s$.

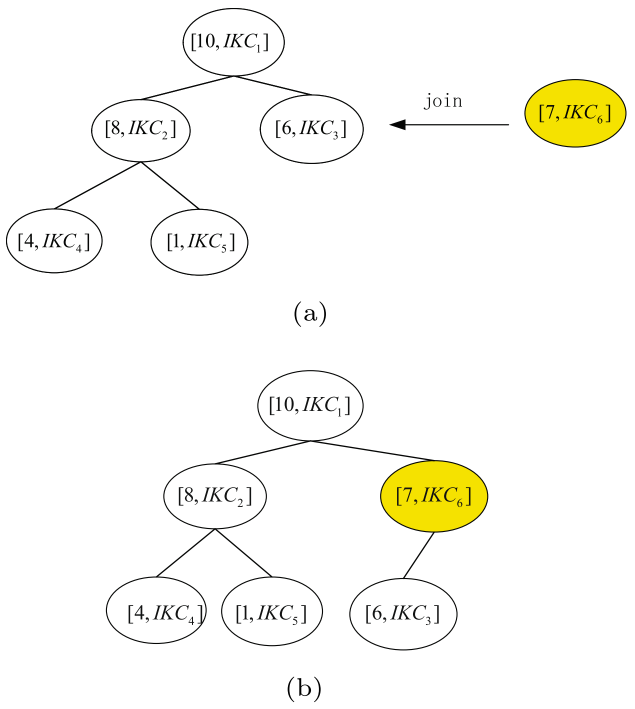

Fig. 4 (a) User number 7 joins group GID_j. (b) User number 7 successfully joins the group GID_j

Step 2: All users (including U_{n+1}) retrieve $GK_{temp}^{'} \oplus sk_i^{ku}$ from the BS using CH_i and derive: $GK_{temp}^{'} = GK_{temp}^{'} \oplus sk_i^{ku} \oplus sk_i^{ku'}$.

Step 3: Based on GID_j, each user accesses $C^{'}$ from the QKCS and computes: $GK_{qk}^{'} = GK_{temp}^{'} \oplus C^{'}$

5.4 Single user leave

When user U_m intends to leave group GID_j, the following procedures are initiated:

(1) User U_m transmits his leaving information to the KDC in encrypted form. Specifically, they utilize the temporary group key GK_{temp} as the encryption key within a symmetric encryption algorithm to protect the message containing $ID_m^{GID_j}$, GID_j, and K_m, forming the ciphertext: $C_m = E_{GK_{temp}}(ID_m^{GID_j} \| GID_j \| K_m)$.

(2) Upon receipt of C_m, the KDC undertakes the following operations:

Step 1: Decrypt C_m with GK_{temp} and verify the information. Once validated, the KDC locates the corresponding record $[d_m, IKC_m]$ in list L for group GID_j, and removes it. A new random integer $d^{'}$ with $0 < d^{'} < p$ is generated. The list for group GID_j is then reorganized to maintain max heap consistency. Assuming the removal does not alter entry order, the group content becomes: $GID_j = \{[d_1, IKC_1], \cdots, [d_{m-1}, IKC_{m-1}], [d_{m+1}, IKC_{m+1}], \cdots, [d_n, IKC_n]\}$. For example, as illustrated in Fig. 5, starting from: $GID_j = \{[10, IKC_1], [8, IKC_2][6, IKC_3], [4, IKC_4], [1, IKC_5]\}$, and removing the user with index 8, the updated group becomes:

$$GID_j = \{[10, IKC_1], [4, IKC_4], [6, IKC_3], [1, IKC_5]\}.$$

Step 2: Compute the new temporary group key by: $GK_{temp}^{'} = GK_{temp} \oplus d^{'}$. Subsequently, the BS is updated with the new entries: $< CH_i^{'}, GK_{temp}^{'} \oplus sk_i^{ku} >$, $< QH, GK_{temp}^{'} \oplus sk^{kq} >$, $(1 \le i < n)$.

(3) The QKCS proceeds as follows:

Step 1: Access $GK_{temp}^{'} \oplus sk^{kq}$ from the BS via QH, and compute: $GK_{temp}^{'} = GK_{temp}^{'} \oplus sk^{kq} \oplus sk^{kq'}$.

Step 2: Retrieve a new group quantum key $GK_{qk}^{'}$, and form the ciphertext: $C^{'} = GK_{temp}^{'} \oplus GK_{qk}^{'} \| GID_j$.

(4) The remaining users derive the updated group key as follows:

Step 1: Each user U_i, $(1 \le i < n)$ retrieves $GK_{temp}^{'} \oplus sk_i^{ku}$ using CH_i and calculates: $GK_{temp}^{'} = GK_{temp}^{'} \oplus sk_i^{ku} \oplus sk_i^{ku'}$.

Step 2: Then, using GID_j, each user accesses $C^{'}$ from the QKCS and computes: $GK_{qk}^{'} = GK_{temp}^{'} \oplus C^{'}$

5.5 Multiple users join

When m users U_i, $(n < i \le n + m)$ wish to join group GID_j, the KDC must pre-register their identity information and generate a timestamp T_6. Each user U_i submits his identity ID_i and public key K_i to the KDC.

(1) Each user U_i, $(n < i \le n + m)$ performs the following:

Step 1: Select a private key k_i such that $0 < k_i < p$, and compute the public key $K_i = [k_i]g$.

Step 2: As in Section 5.1 (2), compute $CH_i = h\left(ID_i \| GID_j\right)$ and $MAC_{k_i} = CH_i \oplus ID_i$, then transmit $\left\{MAC_{k_i}, K_i\right\}$ to the KDC.

(2) The KDC undertakes the following operations:

Step 1: Acquire the current timestamp T_7 and verify that $T_7 - T_6 < \Delta t$.

Step 2: For each user, compute $CH_i^{'} = h\left(ID_i \| GID_j\right)$, $MAC_{k_i}^{'} = CH_i^{'} \oplus ID_i$, and $ID_i^{'} = CH_i^{'} \oplus MAC_{k_i}^{'}$, then validate whether $MAC_{k_i}^{'} == MAC_{k_i}$ and $ID_i == ID_i^{'}$.

Step 3: After successful validation, the KDC constructs a placeholder node $[0, 0, 0, 0]$ and appends it to the GID_j list to separate max heap structures.

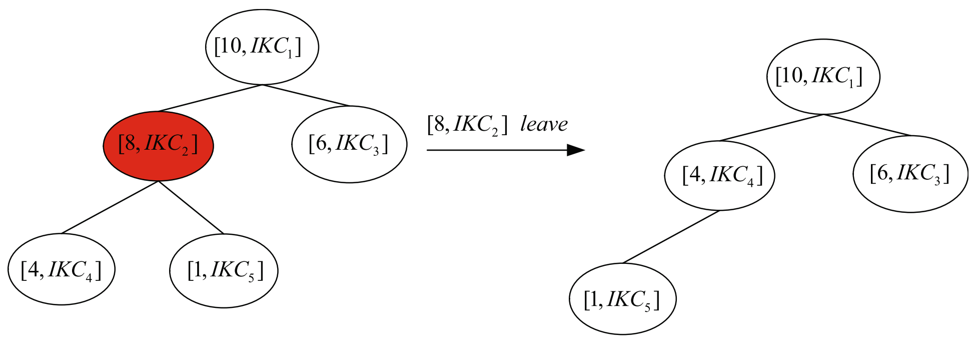

Fig. 5 User number 8 leaves the group GID_j

Then, it builds a local max heap for the new users, randomly assigning indices d_i for each, ensuring $0 < d_i < p$, and updating the group content: $GID_j = \{[d_1, IKC_1], \cdots, [d_n, IKC_n], [0, 0], [d_{n+1}, IKC_{n+1}], \cdots, [d_{n+m}, IKC_{n+m}]\}$.

Step 4: For each new user, compute $node_i = h\left([d_i]K_i\right) \oplus ID_i \| [d_i]g$ and shared key $sk_i^{ku} = [k_i]S$. Then compute the updated temporary group key: $GK_{temp}^{'} = GK_{temp} \oplus node_{n+1} \oplus \cdots \oplus node_{n+m}$. Update the BS with: $< CH_i^{'}, GK_{temp}^{'} \oplus sk_i^{ku} >, < QH, GK_{temp}^{'} \oplus sk^{kq} >, (1 \le i \le n+m)$.

(3) The QKCS executes:

Step 1: Retrieve $GK_{temp}^{'} \oplus sk^{kq}$ using QH and compute $GK_{temp}^{'} = GK_{temp}^{'} \oplus sk^{kq} \oplus sk^{kq^{'}}$.

Step 2: Generate $GK_{qk}^{'}$ and form: $C^{'} = GK_{temp}^{'} \oplus GK_{qk}^{'} \| GID_j$.

(4) The user proceeds as follows:

Step 1: New users $U_i, (n < i \le n+m)$ calculate $sk_i^{ku^{'}} = [k_i]\,S$.

Step 2: All users $U_i, (1 \le i \le n+m)$ retrieve $GK_{temp}^{'} \oplus sk_i^{ku}$ via CH_i and compute: $GK_{temp}^{'} = GK_{temp}^{'} \oplus sk_i^{ku} \oplus sk_i^{ku^{'}}$.

Step 3: Using GID_j, all users access $C^{'}$ and calculate: $GK_{qk}^{'} = GK_{temp}^{'} \oplus C^{'}$

This paragraph explains the advantages of reconstructing a new max heap for newly joined users. Since user joining events require updates to the heap structure, our scheme generates a local max heap independently for each newly formed user subgroup, rather than reconstructing the entire global heap. User dynamics such as joining are thus confined to operations on the local max heaps. This divide-and-conquer heap management design significantly reduces the time complexity of user modification operations, thereby improving the efficiency of group quantum key updates.

5.6 Multiple users leave

When m users leave the group simultaneously, they transmit their leaving information encrypted under the current temporary key. The KDC decrypts and verifies this information, and after deleting the corresponding entries, generates a new random number $d^{''}$ with $0 < d^{''} < p$ and computes: $GK_{temp}^{'} = GK_{temp} \oplus d^{''}$. Update the BS with: $< CH_i^{'}, GK_{temp}^{'} \oplus sk_i^{ku} >, < QH, GK_{temp}^{'} \oplus sk^{kq} >, (1 \le i \le n-m)$. Then, repeat (3) and (4) from Section 5.4 so that all remaining users acquire the updated group quantum key.

6 Correctness and security analysis

6.1 Correctness analysis

This section provides a correctness analysis of the proposed scheme, demonstrating that, under the assumption of no computational errors, users within the same group can consistently derive an identical group quantum key from the KDC.

Theorem 1 *The proposed scheme ensures correctness.*

Proof Let the public-private key pairs of the users, the KDC, and the QKCS be (K_i, k_i), (K_s, s), and (K_q, q), respectively. Let g be a base point on the elliptic curve. The following relations hold:

$$[k_i]\,K_s = [s]\,K_i = [sk_i]\,g \tag{1}$$

$$[q]\,K_s = [s]\,K_q = [sq]\,g \tag{2}$$

After generating the temporary group key GK_{temp}, the KDC computes the values $GK_{temp} \oplus sk_i^{ku}$ and $GK_{temp} \oplus sk^{kq}$, where $sk_i^{ku} = [s]K_i$ and $sk^{kq} = [s]K_q$, and publishes them to the BS using the access identifiers $CH_i^{'}$ and QH, where $CH_i^{'} = h\left(ID_i \| GID_j\right)$, $QH = h\left(K_q \| sk^{kq} \| GID_j\right)$, to put $< CH_i^{'}, GK_{temp} \oplus sk_i^{ku} >, < QH, GK_{temp} \oplus sk^{kq} >$. Consequently, the following BS entries are made: $< CH_i^{'}, GK_{temp} \oplus sk_i^{ku} >$, $< QH, GK_{temp} \oplus sk^{kq} >$. Both the user and the QKCS can retrieve their respective ciphertexts using their identifiers. Since users compute $CH_i = h\left(ID_i \| GID_j\right)$, and the server computes $QH^{'} = h\left(K_q \| sk^{kq^{'}} \| GID_j\right)$, with $sk^{kq^{'}} = [q]K_s$, both values align with the BS identifiers $CH_i^{'}$ and QH. By combining the retrieved values with the shared secrets (as shown in (1) and (2)) and leveraging the properties of XOR, both the user and the QKCS can derive an identical temporary group key GK_{temp}.

After obtaining GK_{temp}, the QKCS encrypts the group quantum key GK_{qk} with GK_{temp} and appends the group identifier GID_j, resulting in the ciphertext distributed to users. Since all group members obtain the same GK_{temp} and GID_j, they can consistently recover the same group quantum key GK_{qk}. This concludes the proof. □

6.2 Formal security analysis

(1) Security model

We modify the $IND-CPA$ model from Paper (Lee et al. 2024) to suit the security model of our scheme.

Initialization: Let SP be the system parameters. The challenger runs the key generation algorithm to generate a public-private key pair (pk, sk), and sends pk and SP to the adversary. The challenger retains sk to respond to encryption queries from the adversary.

Stage 1: The adversary queries adaptively selected plaintexts, and the challenger returns the encrypted results of these plaintexts to the adversary.

Challenge: The adversary outputs two distinct messages, m_1, m_2, both chosen adaptively. The challenger randomly selects $c \in \{1, 2\}$, then computes the challenge ciphertext $CT^* = E[SP, pk, m_c]$ and sends it to the adversary.

Stage 2: This stage is the same as Stage 1, except that no encryption queries are allowed on message m_1, m_2.

Guess: The adversary outputs a guess c for $c^{'}$. If $c = c^{'}$ is correct, the adversary wins the game.

The advantage of the adversary in winning the game, denoted as ε, is defined as follows:

$$\varepsilon = 2\left(\Pr[c^{'} = c] - \frac{1}{2}\right).$$

(2) Security Proof

This section presents a security analysis and proof of the proposed scheme based on the notion of $IND - CPA$.

The scheme does not consider the issue of long-term private key leakage; instead, we focus on analyzing and proving the security of the temporary group key generation process. We reduce the security of our scheme to the hardness of Problem $ECCDH$. Specifically, if an adversary $\mathcal{A}$ is capable of breaking the proposed scheme, we can leverage this adversary to solve Problem $ECCDH$, thereby demonstrating the security of our scheme.

Theorem 2 *Assume that the hash function H is modeled as a random oracle. If the $ECCDH$ problem is computationally hard, then the proposed scheme is provably secure under the IND-CPA security model, with a reduction loss of $L = q_H$, where q_H denotes the number of hash queries made to the random oracle.*

Proof Assume there exists an adversary $\mathcal{A}$ that can break the scheme with advantage ε in time t under the IND-CPA security model. We construct a simulator $\mathcal{R}$ that uses $\mathcal{A}$ to solve the $ECCDH$ problem.

Given an $ECCDH$ instance $(g, [a]g, [b]g)$ on a cyclic group (G, g, p), the simulator $\mathcal{R}$ interacts with $\mathcal{A}$ as follows while controlling the random oracle:

Initialization: Let the system parameters be $SP = (G, g, p)$, and let H be a random oracle. $\mathcal{R}$ sets the public key of a particular user θ as $K_\theta = [a]g$, with corresponding private key $\alpha = a$, and identity information ID_θ. The public key is directly derived from the given problem instance.

Stage 1: The adversary $\mathcal{A}$ adaptively selects plaintexts for encryption queries, and the challenger returns their corresponding ciphertexts. During this stage, $\mathcal{A}$ also makes hash queries. The simulator $\mathcal{R}$ maintains a hash list to record all queried values and their corresponding responses, initially starting as an empty list.

When $\mathcal{A}$ makes its i-th hash query with input x_i, the simulator checks whether x_i already exists in the hash list. If it does, $\mathcal{R}$ responds with the previously stored value. Otherwise, $\mathcal{R}$ randomly selects $y_i \in \{0, 1\}^n$, sets $H(x_i) = y_i$, returns y_i as the response to the query, and appends the pair (x_i, y_i) to the hash list.

Challenge: $\mathcal{A}$ outputs two messages $m_1 = ID_1$ and $m_2 = ID_2$ of equal length to be used in the challenge. Here, m_1 represents the identity information of user U_1, and m_2 represents the identity information of user U_2. Both U_1 and U_2 join the same group GID_j, each accompanied by the same set of n users. Assume that the target user θ is one of the n users.

The simulator $\mathcal{R}$ randomly selects $R \in \{0, 1\}^n$ and computes the challenge ciphertext CT^* as follows:

User U_1 and the n group users compute the challenge ciphertext $CT^*_{m_1}$ as:

$$node_i = h([d_i]K_i) \oplus ID_i^{GID_j} \| [d_i]g, \quad (1 \le i \le n+2,$$

$$i \ne 2,\ i \ne \theta)$$

$$node_\theta = R \| [d_\theta]g$$

$$GK^{m_1}_{temp} = node_1 \oplus node_3 \oplus \cdots \oplus node_\theta \oplus \cdots \oplus node_{n+2}$$

Let $sk_i^{ku} = [s]K_i$ for all i such that $1 \le i \le n+2$ and $i \ne 2$. Then,

$$CT^*_{m_1} = GK^{m_1}_{temp} \oplus sk_i^{ku}$$

User U_2 and the n group users compute the challenge ciphertext $CT^*_{m_2}$ as:

$$node_i = h([d_i]K_i) \oplus ID_i^{GID_j} \| [d_i]g, \quad (2 \le i \le n+2,$$

$$i \ne \theta)$$

$$node_\theta = R \| [d_\theta]g$$

$$GK^{m_2}_{temp} = node_2 \oplus node_3 \oplus \cdots \oplus node_\theta \oplus \cdots \oplus node_{n+2}$$

Let $sk_i^{ku} = [s]K_i$ for all i such that $2 \le i \le n+2$. Then,

$$CT^*_{m_2} = GK^{m_2}_{temp} \oplus sk_i^{ku}$$

If it holds that $h([b]K_\theta) = R \oplus ID_\theta$, then we can write:

$$node_\theta = h([b]K_\theta) \oplus ID_\theta \| [b]g$$

In this case, the value of $node_\theta$ used in the challenge ciphertext can be interpreted as being computed from the random value b. Therefore, if $[b]K_\theta$ is not queried to the random oracle by the adversary, the challenge ciphertext will be computationally indistinguishable from a valid ciphertext and thus appears legitimate from the adversary's point of view.

Guess: $\mathcal{A}$ outputs a guess or returns $\perp$. The challenge hash query is defined as follows:

$$Q^* = [b]K_\theta = [ab]g.$$

The simulator randomly selects a value x from the hash list $(x_1, y_1), (x_2, y_2), \cdots, (x_{q_H}, y_{q_H})$ and treats it as the challenge hash query. It then uses this value to solve the $ECCDH$ problem.

Indistinguishability of the Simulation: The correctness of the simulation process has been demonstrated above. The randomness in the simulation includes all random values generated during key generation, hash oracle responses, and challenge ciphertext construction. From the adversary's perspective, these values appear uniformly random and independent. Therefore, the simulation is computationally indistinguishable from a real attack scenario.

Adversarial Advantage in Breaking the Challenge Ciphertext: If $h([b]K_\theta) = R \oplus ID_\theta$, then the challenge ciphertext $CT^*_{m_1}$ corresponds to message m_1, and $CT^*_{m_2}$ corresponds to message m_2. However, if the adversary does not query $[b]K_\theta$ to the random oracle, the value $h([b]K_\theta)$ remains hidden and uniformly random. As a result, the adversary gains no advantage in distinguishing the challenge ciphertext, and the challenge remains secure.

Reduction Advantage and Time Complexity: Let T_s denote the time required for simulation. Then the simulator $\mathcal{R}$ solves the $ECCDH$ problem within time $(t + T_s)$ with an advantage of (ε/q_H), where ε is the success probability of adversary $\mathcal{A}$ and q_H is the number of hash queries made to the random oracle.

6.3 Non-formal security analysis

(1) Forward Secrecy
In the proposed scheme, the temporary group key GK_{temp} and the group quantum key GK_{qk} are updated whenever a user U_i joins or leaves the group. Even if the current GK_{temp} is compromised, the attacker cannot infer previous keys due to the randomness introduced by the KDC, which assigns each user a fresh random index d_i during initialization, joining, and leaving operations. Moreover, the group quantum key GK_{qk} itself is generated with inherent randomness. Therefore, the proposed scheme satisfies forward secrecy.

(2) Backward Secrecy
Analogous to the forward secrecy case, even if the current GK_{temp} is exposed, the attacker cannot deduce any future temporary group keys. This is due to the randomized index d_i generated by the KDC during each membership update. The randomness embedded in GK_{qk} also ensures that backward secrecy is preserved.

(3) Asynchrony
In our scheme, once user U_i has transmitted their identity ID_i and public key K_i to the KDC, they are no longer required to remain online. When the group quantum key GK_{qk} is needed, the user can retrieve the temporary group key GK_{temp} from the BS and download the ciphertext C from the QKCS. By decrypting C with GK_{temp}, the user obtains GK_{qk}. Hence, the scheme supports asynchronous access.

(4) Resistance to Man-in-the-Middle Attack
In the proposed scheme, the index d_i used to generate GK_{temp} is randomly assigned and never transmitted. Therefore, an attacker cannot forge the group key based on any transmitted information. Moreover, since reconstructing GK_{temp} requires the user's private key k_i, it is infeasible for an attacker to impersonate a legitimate user without possessing this key. As a result, the scheme is resilient to man-in-the-middle attacks.

(5) Anonymity
The scheme guarantees anonymity since user U_i retrieves GK_{qk} independently by accessing parameters from the BS and the QKCS. The process is conducted without revealing or learning the membership status (joining or leaving) of other users U_j, $j \neq i$. Thus, anonymity is preserved.

(6) Immutability As users retrieve the group key GK_{temp} from the BS, the immutability of BS data ensures that this information cannot be altered by malicious parties. Consequently, the integrity of the group key is preserved, and the proposed scheme upholds the immutability property.

7 Performance evaluation

In this section, we assess the performance of our proposed scheme in terms of both storage and computational costs.

Table 2 Computation cost of Scheme

Symbol	Description	Time (ms)
t_h	hash	0.000350
t_{pa}	elliptic curve point addition	0.084967
t_{pm}	elliptic curve point multiplication	1.844129
t_{sym}	symmetric encryption or decryption	0.006487
t_{ex}	shift operation	0.000077
t_{mod}	modular operations	0.000134

Table 3 Storage cost

Scheme	Single User	KDC	BS	QKCS
Yildiz	$t+3$	$5n+3+n_t+\sum_{l=1}^{t-1} l! - (t-1)!$ $n_t = \lceil \{n-(t-1)!\}/(t-1) \rceil$	–	–
Sudheeradh	$t+2$	$6n+2+n_t+\sum_{l=1}^{t-1} l! - (t-1)!$ $n_t = \lceil \{n-(t-1)!\}/(t-1) \rceil$	–	–
Ours	6	$5n+5$	$2n+2$	5

We conducted experiments using the Python programming language. The elliptic curve operations, such as point multiplication and point addition, were performed via the `ECPy` library. Symmetric encryption and decryption were carried out using AES-256 provided by the `cryptography` library. Hash computations were implemented with SHA-256 from Python's `hashlib` module. Shift and modular operations were coded manually. All experiments were run on a platform equipped with an Intel(R) Core(TM) i9-14900HX @2.20 GHz processor, 16GB of RAM, and the 64-bit version of Windows 11 (version 23H2).

We denote the average execution time of each operation as follows: t_h for hash computations, t_{pa} for elliptic curve point addition, t_{pm} for elliptic curve point multiplication, t_{sym} for symmetric encryption or decryption, t_{ex} for shift operation, and t_{mod} for modular operations. Since XOR and concatenation operations incur negligible costs, their computational cost is excluded from our analysis. For storage cost evaluation, we let C represent the cost of storing a single data item. Accordingly, the cost of storing n such items is nC. We conducted 1000 trials for each of the six operations mentioned above and averaged the results. The performance outcomes are summarized in Table 2. Following this, we present a comparative analysis of our scheme against those proposed by Yıldız et al. (2021) and Sudheeradh et al. (2024), focusing specifically on users' storage and computation costs.

In this section, we present a theoretical evaluation of the performance of the proposed scheme.

7.1 Storage cost

We assume the presence of n users and compare the storage cost of corresponding entities in our scheme with those in the schemes proposed by Yıldız et al. (2021) and Sudheeradh et al. (2024). The comparison results are summarized in Table 3. In this table, t denotes the height of the factorial tree used in the schemes of Yıldız et al. (2021) and Sudheeradh et al. (2024), and the unit of data is represented by C.

In our scheme, each user is required to store their private key, group identifier, the public key of the KDC, the temporary group key, the access identifier for retrieving data from the BS, and the user's shared key. Thus, the storage cost per user is $6C$, and the total user-side storage cost is $6nC$. The KDC stores essential information for each user node, including the user's index, identity, and public key, access identifiers for BS retrieval by users and the QKCS, its shared keys with each user and the QKCS, its own private key, group identifier, and the temporary group key. Hence, the KDC's total storage cost amounts to $(5n+5)C$. The BS stores the access identifiers and encrypted parameters required for users and the QKCS, leading to a storage cost of $(2n+2)C$. The QKCS maintains its own private key, group identifier, public

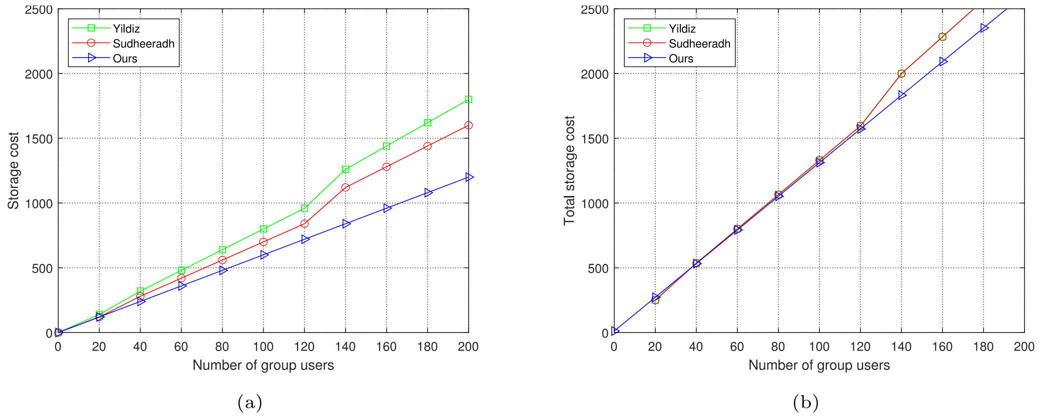

Fig. 6 (a) Comparison of user storage costs. (b) Comparison of user total storage costs

Table 4 Computation cost of initialization and key distribution phase

Scheme	Single User	KDC
Yildiz	$3t_{sym} + 3t_h$	$(3n + 1)t_{sym} + 3nt_h$
Sudheeradh	$4t_{sym} + 3t_h$	$5nt_{sym} + 4nt_h$
Ours	$2t_{pm} + t_h$	$(3n + 1)t_{pm} + (n + 1)t_h$

Table 6 Computation cost of leaving phase

Scheme	Single user leave	Original single user	KDC
Yildiz	–	$3t_{sym} + 3t_h + t_{mod}$	$\frac{1}{2}\left(t^2 - t\right) t_{\text{ mod}} + t_{sym}$
Sudheeradh	–	$t_{sym} + t_h + t_{mod}$	$\frac{1}{2}\left(t^2 - t\right) t_{\text{ mod}} + t_{sym}$
Ours	t_{sym}	–	$t_h + \log(n)t_{ex}$

key of the KDC, the access identifier used to retrieve data from the BS, and its own shared key, resulting in a total cost of $5C$. Therefore, the overall storage cost for our scheme is $(13n + 12)C$. In contrast, the storage cost for the schemes of Yıldız and Sudheeradh are given respectively as:

$$\left((t + 8)n + 3 + \left\lceil \frac{n-(t-1)!}{t-1} \right\rceil + \sum_{l=1}^{t-1} l! - (t - 1)!\right) C,$$

$$\left((t + 8)n + 2 + \left\lceil \frac{n-(t-1)!}{t-1} \right\rceil + \sum_{l=1}^{t-1} l! - (t - 1)!\right) C.$$

Figure 6(a) and (b) presents a comparative analysis of user-side and total storage costs under increasing numbers of users for our scheme versus those of Yıldız and Sudheeradh. As shown in Fig. 6(a) and (b), our scheme exhibits a slower growth rate in both user and overall storage costs as the number of users increases. This efficiency arises from our max heap-based design, where each user occupies a single node, and all nodes collectively contribute to the generation of the temporary group key. In contrast, the other two schemes rely on storing users as leaf nodes in a factorial tree. During group key updates, users and the key server must store additional metadata that records the absolute path from leaf to root, which increases storage costs as the number of users grows.

7.2 Computation cost

We assume the number of users ranges from 1 to 2000, and we summarize the computational costs of our proposed scheme as well as those of Yıldız et al. (2021) and Sudheeradh et al. (2024) during the initialization and key distribution phase, the user joining phase, and the user leaving phase. The detailed comparisons are presented in Tables 4, 5, and 6, respectively.

In the proposed scheme, during the initialization and key distribution phases, each user performs two elliptic curve point multiplications: one to generate their public key and one to compute the shared key, along with a single hash operation. Consequently, the per-user computational cost is $2t_{pm} + t_h$, and the overall user-side computational cost is $2nt_{pm} + nt_h$. During the key distribution phase, the KDC performs n hash operations. To construct the temporary group key, it performs $2n$ elliptic curve point multiplications, and encrypting the temporary group key requires an additional $n + 1$ point multiplications and one hash operation. Thus, the KDC's computational cost is $(3n + 1)t_{pm} + (n + 1)t_h$. The BS merely stores parameters and does not incur any computational cost. The QKCS performs one elliptic curve point multiplication and one hash operation, resulting in a computational cost of $t_{pm} + t_h$. Hence, the total computational cost of our scheme is $(5n + 2)t_{pm} + (2n + 2)t_h$. For comparison, the total computational costs of the schemes by Yıldız and Sudheeradh are $(6n + 1)t_{sym} + 6nt_h$ and $9nt_{sym} + 7nt_h$, respectively.

In the new user joining phase of our scheme, the newly joining user performs two elliptic curve point multiplications and one hash operation, while the existing users only perform XOR operations, which are negligible in cost. Thus, the total user-side computational cost is $2t_{pm} + t_h$. The KDC performs three point multiplications, one hash operation, and approximately $\log(n)$ exchange operations, leading to a cost

Table 5 Computation cost of joining phase

Scheme	Newly added single user	Original single user	KDC
Yildiz	$3t_{sym} + 3t_h$	$t_{sym} + 2t_h$	$5t_{sym} + 5t_h$
Sudheeradh	t_{sym}	t_{sym}	$t_{sym} + t_h$
Ours	$2t_{pm} + t_h$	–	$3t_{pm} + t_h + \log(n)t_{ex}$

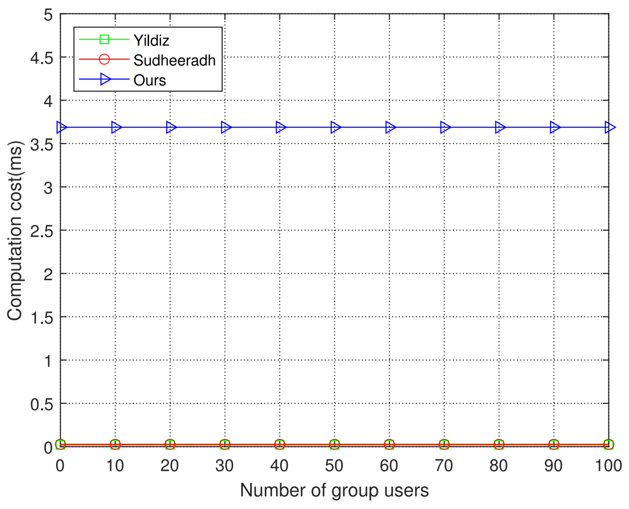

Fig. 7 Computation cost of a single user in Initialization and key distribution phase

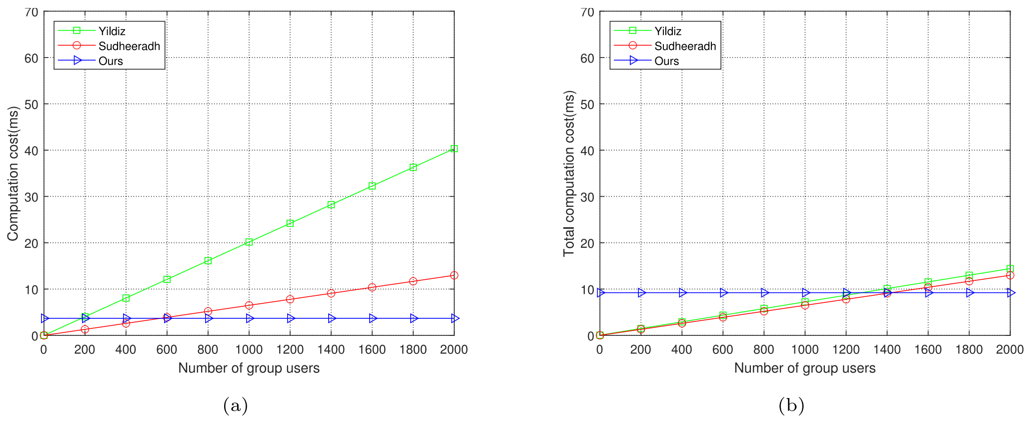

Fig. 8 (a) Computation cost during joining phase. (b) Total computation cost during the joining phase

of $3t_{pm} + t_h + \log(n)t_{ex}$. The BS and the QKCS are only involved in data storage and XOR operations, respectively, both incurring negligible computational costs. Therefore, the total cost of this phase is $5t_{pm} + 2t_h + \log(n)t_{ex}$. The corresponding costs in the schemes of Yıldız and Sudheeradh are $(n+7)t_{sym} + (2n+6)t_h$ and $(n+1)t_{sym} + t_h$, respectively.

During the user leaving phase, the departing user encrypts their own information using the current temporary group key, yielding a cost of t_{sym}. Non-departing users only perform XOR operations, which are negligible. The KDC performs one decryption and $\log(n)$ shift operations, resulting in a cost of $t_{sym} + \log(n)t_{ex}$. Thus, the total cost of the user leaving phase is $2t_{sym} + \log(n)t_{ex}$. The schemes of Yıldız and Sudheeradh have respective computational costs of:

$$\frac{1}{2}\left(t^2 - t\right) t_{\text{mod}} + t_{sym} + (n-1)(3t_{sym} + 3t_h + t_{\text{mod}}),$$

$$\frac{1}{2}\left(t^2 - t\right) t_{\text{mod}} + t_{sym} + (n-1)(t_{sym} + t_h + t_{\text{mod}})$$

Figure 7 presents a comparison of the per-user computational cost during the initialization and key distribution phases for the three schemes. As observed in the figure, the per-user computational cost of our scheme is slightly higher than that of the other two. This is primarily because our scheme adopts elliptic curve point multiplication in the construction of the shared key, whereas the other two schemes utilize symmetric encryption, which is computationally less intensive.

Figure 8(a) and (b) illustrate the comparison of user-side and total computational costs during the user joining phase for the three schemes. In our scheme, only the newly added user needs to perform elliptic curve point multiplications and a hash operation, while existing group members only execute XOR operations to update the group key. As a result, the user's computational cost remains independent of the total group size. Furthermore, the KDC computes a new shared key for the joining user and updates the temporary group key, while the BS and QKCS perform update operations with negligible computational cost. Thus, the total computational

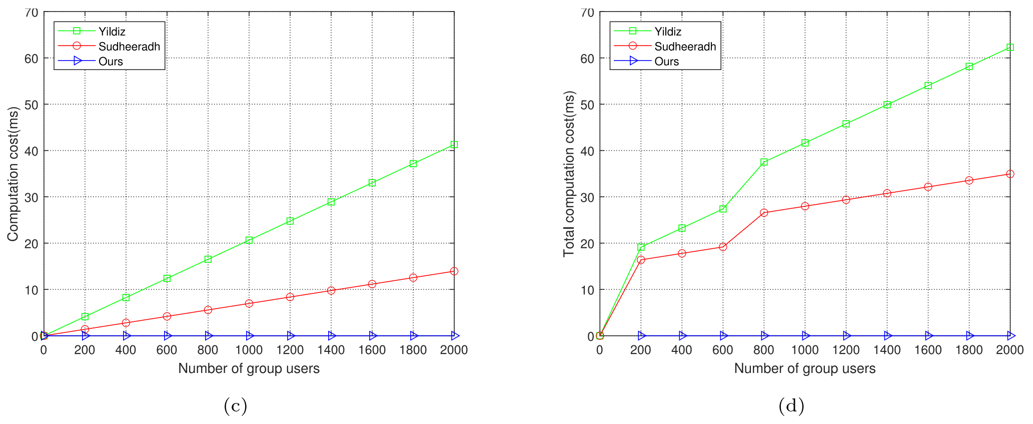

Fig. 9 (a) Computation cost during leaving phase. (b) Total computation cost during the leaving phase

cost is only related to the newly added member. In contrast, in the schemes proposed by Yıldız and Sudheeradh , the KDC must generate parameters and encrypt them for distribution to all users. Each user must also participate in updating the group key. Consequently, as the number of group members increases, both the user-side and total computational costs increase accordingly.

Figure 9(a) and (b) illustrate the comparison of user-side and total computational costs during the user leaving phase for the three schemes. In our proposed scheme, when a user leaves, the departing user only needs to encrypt their information with the temporary group key and transmit it to the KDC. After decryption, the KDC updates the temporary group key. Non-departing users simply perform lightweight XOR operations to complete the key update process. Therefore, both user-side and total computational costs remain independent of the number of users in the group. In contrast, the schemes proposed by Yıldız and Sudheeradh rely on the Chinese Remainder Theorem (CRT) to update keys upon the user leaving. Their computational costs are directly tied to the height of the tree structure employed, and thus both user-side and total computational costs increase proportionally with the number of users.

8 Conclusion

In this paper, we propose an asynchronous group quantum key distribution scheme over classical networks based on a max heap structure. Leveraging the inherent randomness of quantum keys, we designate them as the communication group key. On this basis, we construct a temporary group key using a max heap derived from user contributions, which enables dynamic load balancing across heterogeneous devices in networked environments such as the Internet of Things (IoT). Furthermore, we utilize BS to store intermediate parameters, thereby achieving the asynchronous nature of the proposed scheme. Finally, we provide a comprehensive security and performance analysis, demonstrating that our scheme outperforms existing approaches in terms of efficiency and adaptability.

Author Contributions Dexin Zhu: Methodology, Software, Writing - Original Draft. Zilong Zhao: Software, Writing - Original Draft. Huanjie Zhang: Software, Visualization. Zhiqiang Zhou: Software. Yuanbo Li: Software. Jian Zhao: Investigation. Lijun Song: Methodology. Jun Zheng: Conceptualization, Methodology, Writing - Original Draft.

Funding This work is funded by the National Key Research and Development Program of China under Grant 2023YFC3305404, the Education Department of Jilin Province with grants No. JJKH20251102KJ, the Science and Technology Department of Jilin Province with grants No. 20250102045JC, 20250102055JC.

Data Availability Enquiries about data availability should be directed to the authors.

Declarations

Conflicts of Interest The authors declare no conflict of interest.

References

Abdmeziem MR, Nacer AA, Deroues NM (2024) Group key management in the internet of things: handling asynchronicity. Future Gener Comput Syst 152:273–287. https://doi.org/10.1016/J.FUTURE.2023.10.023

Alléaume R, Branciard C, Bouda J, Debuisschert T, Dianati M, Gisin N, Godfrey M, Grangier P, Länger T, Lütkenhaus N, Monyk C, Painchault P, Peev M, Poppe A, Pornin T, Rarity JG, Renner R, Ribordy G, Riguidel M, Salvail L, Shields AJ, Weinfurter H, Zeilinger A (2014) Using quantum key distribution for cryptographic purposes: a survey. Theor Comput Sci 560:62–81. https://doi.org/10.1016/J.TCS.2014.09.018

Braeken A (2022) Pairing free asymmetric group key agreement protocol. Comput Commun 181:267–273. https://doi.org/10.1016/J.COMCOM.2021.10.011

Chen C, Deng X, Gan W, Chen J, Islam SH (2021) A secure blockchain-based group key agreement protocol for iot. J Supercomput 77(8):9046–9068. https://doi.org/10.1007/S11227-020-03561-Y

Cheng T, Liu Q, Shi Q, Yang Z, Wang C, Zhang X, Xu P (2024) Efficient anonymous authentication and group key distribution scheme based on quantum random numbers for vanets. IEEE Internet Things J 11(13):23544–23560. https://doi.org/10.1109/JIOT.2024.3384993

Chhikara D, Rana S, Singh G, Mishra D, Kumar N (2023) Blockchain-based partial group key agreement protocol for intelligent transportation systems. IEEE Trans Veh Technol 72(12):16701–16710. https://doi.org/10.1109/TVT.2023.3299705

Chien H (2021) Self-healing group key distribution facilitating source authentication using block codes. Secur Commun Networks 2021:2942568–1294256811. https://doi.org/10.1155/2021/2942568

Cui B, He W, Cui Y (2024) A dynamic C-V2X anonymous authentication and group key agreement protocol. Int J Inf Sec 23(4):2977–2989. https://doi.org/10.1007/S10207-024-00876-2

Diamanti E, Leverrier A (2015) Distributing secret keys with quantum continuous variables: principle, security and implementations. Entropy 17(9):6072–6092. https://doi.org/10.3390/E17096072

Gan Y, Wang B, Zhuang Y, Gao Y, Li Z, Zhang Q (2021) An asymmetric group key agreement protocol based on attribute threshold for internet of things. Trans Emerg Telecommun Technol 32(5). https://doi.org/10.1002/ETT.4179

García JCP, Braeken A, Benslimane A (2024) Blockchain-based group key management scheme for iot with anonymity of group members. IEEE Trans Inf Forensics Secur 19:6709–6721. https://doi.org/10.1109/TIFS.2024.3414663

Gebremichael T, Gidlund M, Hancke GP, Jennehag U (2022) Quantum-safe group key establishment protocol from lattice trapdoors. Sensors 22(11):4148. https://doi.org/10.3390/S22114148

Harn L, Hsu C, Xia Z (2021) Lightweight and flexible key distribution schemes for secure group communications. Wirel Networks 27(1):129–136. https://doi.org/10.1007/S11276-020-02449-2

Harn L, Hsu C, Xia Z (2022) General logic-operation-based lightweight group-key distribution schemes for internet of vehicles. Veh Commun 34:100457. https://doi.org/10.1016/J.VEHCOM.2022.100457

Hougaard HB, Miyaji A (2022) Authenticated logarithmic-order supersingular isogeny group key exchange. Int J Inf Sec 21(2):207–221. https://doi.org/10.1007/S10207-021-00549-4

Houzhen W, Wanying Q, Qin L, Chunwu Y, Zhidong S (2023) Identity based group key distribution scheme. J Comput Res Dev 60(10):2203–2217. https://doi.org/10.7544/issn1000-1239.202330457

Kemmoe VY, Kwon Y, Hussain R, Cho S, Son J (2023) Leveraging smart contracts for secure and asynchronous group key exchange without trusted third party. IEEE Trans Dependable Secur Comput 20(4):3176–3193. https://doi.org/10.1109/TDSC.2022.3189977

Kumar V, Kumar R, Pandey SK (2020) A computationally efficient centralized group key distribution protocol for secure multicast communications based upon RSA public key cryptosystem. J King Saud Univ Comput Inf Sci 32(9):1081–1094. https://doi.org/10.1016/J.JKSUCI.2017.12.014

Lee J, Oh J, Kwon DK, Kim M, Kim K, Park Y (2024) Blockchain-enabled key aggregate searchable encryption scheme for personal health record sharing with multidelegation. IEEE Internet Things J 11(10):17482–17494. https://doi.org/10.1109/JIOT.2024.3357802

Li X, Wang Y, Vijayakumar P, He D, Kumar N, Ma J (2019) Blockchain-based mutual-healing group key distribution scheme in unmanned aerial vehicles ad-hoc network. IEEE Trans Veh Technol 68(11):11309–11322. https://doi.org/10.1109/TVT.2019.2943118

Lo H, Curty M, Tamaki K (2014) Secure quantum key distribution. Nature Photon 8(149):595–604. https://doi.org/10.1038/nphoton.2014.149

Luo Y, Mao H, Li Q, Chen N (2023) An information-theoretic secure group authentication scheme for quantum key distribution networks. IEEE Trans Commun 71(9):5420–5431. https://doi.org/10.1109/TCOMM.2023.3280561

Naresh VS, Allavarpu VVLD, Reddi S (2022) Provably secure blockchain privacy-preserving smart contract centric dynamic group key agreement for large WSN. J Supercomput 78(6):8708–8732. https://doi.org/10.1007/S11227-021-04175-8

Pirandola S, Andersen UL, Banchi L, Berta M, Bunandar D, Colbeck R, Englund D, Gehring T, Lupo C, Ottaviani C et al (2020) Advances in quantum cryptography. Adv Opt Photon 12(4):1012–1236

Rahimi P, Chrysostomou C (2019) Improving the network lifetime and performance of wireless sensor networks for iot applications based on fuzzy logic. In: 15th International conference on distributed computing in sensor systems, DCOSS 2019, Santorini, Greece, May 29-31, 2019. IEEE, Piscataway, NJ, USA, pp 667–674. https://doi.org/10.1109/DCOSS.2019.00120

Rawat AS, Deshmukh M (2020) Tree and elliptic curve based efficient and secure group key agreement protocol. J Inf Secur Appl 55:102599. https://doi.org/10.1016/J.JISA.2020.102599

Sudheeradh K, Jahnavi NN, Chine PN, Kasbekar GS (2024) Efficient and secure group key management scheme based on factorial trees for dynamic iot settings. IEEE Access 12:5659–5671. https://doi.org/10.1109/ACCESS.2024.3350780

Taurshia A, Kathrine GJW, Souri A, Vinodh SE, Vimal S, Li K, Ilango SS (2022) Software-defined network aided lightweight group key management for resource-constrained internet of things devices. Sustain Comput Informatics Syst 36:100807. https://doi.org/10.1016/J.SUSCOM.2022.100807

Wang Z, Huo R, Wang S (2022a) A lightweight certificateless group key agreement method without pairing based on blockchain for smart grid. Future Internet 14(4):119. https://doi.org/10.3390/FI14040119

Wang Z, Yang Z, Li F (2022b) A two rounds dynamic authenticated group key agreement protocol based on LWE. J Syst Archit 133:102756. https://doi.org/10.1016/J.SYSARC.2022.102756

Xiong H, Wu Y, Lu Z (2019) A survey of group key agreement protocols with constant rounds. ACM Comput Surv 52(3):57–15732. https://doi.org/10.1145/3318460

Xu Z, Liang W, Li K, Xu J, Zomaya AY, Zhang J (2022) A time-sensitive token-based anonymous authentication and dynamic group key agreement scheme for industry 5.0. IEEE Trans Ind Informatics 18(10):7118–7127. https://doi.org/10.1109/TII.2021.3129631

Yang Z, Wang Z, Qiu F, Li F (2023) A group key agreement protocol based on ECDH and short signature. J Inf Secur Appl 72:103388. https://doi.org/10.1016/J.JISA.2022.103388

Yildiz H, Cenk M, Onur E (2021) PLGAKD: A puf-based lightweight group authentication and key distribution protocol. IEEE Internet Things J 8(7):5682–5696. https://doi.org/10.1109/JIOT.2020.3032757

Zhang R, Zhang L, Choo KR, Chen T (2023a) Dynamic authenticated asymmetric group key agreement with sender non-repudiation and privacy for group-oriented applications. IEEE Trans Dependable Secur Comput 20(1):492–505. https://doi.org/10.1109/TDSC.2021.3138445

Zhang Z, Li X, Wang Y, Miao Y, Liu X, Weng J, Deng RH (2023b) TAGKA: threshold authenticated group key agreement protocol against member disconnect for UANET. IEEE Trans Veh Technol 72(11):14987–15001. https://doi.org/10.1109/TVT.2023.3287487