IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 6 July 2025, accepted 30 July 2025, date of publication 4 August 2025, date of current version 8 August 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3595584

RESEARCH ARTICLE

Dynamic Group Key Exchange Protocol With Lattice Trapdoor and Re-Encryption Mechanisms

NARN-YIH LEE[1], (Member, IEEE), AND YUAN-TING TSAI[2]
[1]Computer and Network Center, National Cheng Kung University, Tainan 70101, Taiwan
[2]Department of Electrical Engineering, National Cheng Kung University, Tainan 70101, Taiwan

Corresponding author: Narn-Yih Lee (nylee@gs.ncku.edu.tw)

This work was supported in part by the National Science and Technology Council (NSTC), Taiwan, under Grant NSTC 113-2221-E-006-227.

ABSTRACT Post-quantum group key exchange protocols have become an increasingly critical area of research due to the potential threat quantum computers pose to classical cryptographic schemes. Traditional group key exchange protocols, often based on Diffie-Hellman and other number-theoretic assumptions, are vulnerable to quantum attacks. To address this challenge, we propose a novel group key exchange protocol that leverages lattice-based cryptography. Specifically, our approach utilizes lattice trapdoors and a re-encryption mechanism based on the Ring Learning With Errors (RLWE) problem. This design not only achieves post-quantum security but also enhances efficiency and flexibility in dynamic group settings. A common challenge in most protocols is the computational cost of deriving new keys as group members join and leave. Our approach effectively reduces this cost, making key updates more efficient and practical. Our protocol is proven secure under standard lattice assumptions, providing a promising solution for quantum-resistant group key exchange.

INDEX TERMS Post-quantum cryptography, ring learning with errors, lattice trapdoors, group key exchange, lattice re-encryption.

I. INTRODUCTION

Key exchange protocols are a fundamental area in cryptography, enabling parties to securely establish a shared session key over an insecure channel. Session keys are crucial for ensuring the confidentiality and integrity of communications. Traditionally, group key exchange protocols have often been designed based on the Diffie-Hellman key exchange, such as the protocol proposed by Ingemarsson et al. [1], which extended the Diffie-Hellman scheme to a group setting for establishing a shared group key. Similarly, Steiner et al. [2] introduced a group key agreement protocol using tree-based Diffie-Hellman, improving efficiency and scalability in large groups. Another notable example is the Burmester-Desmedt protocol [3], which achieves constant rounds and is efficient for dynamic group settings.

However, with the advent of quantum computing, traditional schemes relying on number-theoretic assumptions, such as the discrete logarithm problem, face the risk of being broken. This has driven the need for post-quantum secure protocols. Recent works have explored group key exchange protocols based on lattice problems, such as Learning with Errors (LWE) and Ring Learning with Errors (RLWE). For instance, Yang et al. [13] proposed a two-round group authenticated key exchange scheme leveraging LWE and secure-sketch extraction to achieve mutual authentication and session-key consistency. Similarly, Apon et al. [5] presented a lattice-based authenticated group key exchange protocol, ensuring security under the LWE assumption. Moreover, Samiullah et al. [14] introduced a Saber-based GAKE tailored for IoT, demonstrating sub-millisecond latency.

Despite these advancements, existing lattice-based protocols often suffer from large key sizes, increased computational costs, or complex operations. In particular, when a group member leaves, many existing protocols require costly recomputation of a new group key, leading to significant overhead. In this paper, we propose a new group key exchange protocol that leverages lattice trapdoors and RLWE-based re-encryption techniques to address these challenges. By designing the group key exchange process

The associate editor coordinating the review of this manuscript and approving it for publication was Chi-Yuan Chen.

with member departures in mind, our protocol inherently reduces the computational cost associated with key updates, while achieving quantum resistance and enhancing efficiency and practicality in group key establishment.

This paper presents a novel group key exchange (GKE) protocol that leverages lattice trapdoors and re-encryption techniques to achieve post-quantum security and efficiency in dynamic group settings. The main contributions are summarized as follows:

- An efficient Group Key Exchange protocol is proposed, based on lattice trapdoors and re-encryption techniques, designed to achieve quantum resistance while ensuring practicality in real-world applications.
- A formal security proof is provided, demonstrating the protocol's robustness against quantum attacks under standard lattice-based assumptions, ensuring cryptographic soundness.
- The re-encryption technique facilitates efficient and seamless dynamic group management, particularly when handling member join and leave, with minimal computational cost and disruption to the group key agreement process.
- This paper presents a novel group key exchange approach, offering new insights that may inspire future developments in lattice-based cryptographic key exchange.

II. RELATED WORK

A. LWE-BASED GAKE VIA SECURE SKETCH

Yang et al. [13] propose a passively secure two-round group authenticated key exchange protocol based on LWE and its ring-LWE extension in the random oracle model. Their construction employs a novel robust extractor (secure sketch) for key derivation, achieving mutual authentication and session-key consistency with only $n-1$ unicasts followed by a single broadcast. By offloading the bulk of lattice operations and secure-sketch computations to a trusted authority, client devices are limited to lightweight Gaussian sampling and vector multiplications, making the scheme efficient for resource-constrained settings. However, the protocol assumes static membership and does not support dynamic join/leave.

B. CONSTANT-ROUND RLWE-BASED GKE

Apon et al. [5] propose a constant-round, unauthenticated group key exchange protocol built from the Ring-LWE assumption by generalizing the Burmester-Desmedt design to the polynomial-ring setting. Their scheme completes in two broadcast rounds: parties first exchange RLWE samples with small noise, then perform a linear combination followed by a reconciliation step to derive an identical shared value. Security against a passive eavesdropper is proven in the random-oracle model under Ring-LWE, and applying the Katz-Yung compiler with any post-quantum signature yields an authenticated variant. This work demonstrates scalable, post-quantum security but assumes static group membership and does not support dynamic joins, leaves.

C. SABER-BASED GAKE FOR IOT

Samiullah et al. [14] present the first Saber-based post-quantum group authenticated key exchange protocol tailored for resource-constrained IoT devices and proven secure in the Quantum Random Oracle Model (QROM). Their design leverages the Module-LWR assumption to achieve constant-round key establishment, and a proof-of-concept implementation on ARM Cortex-M4 and M0 demonstrates sub-millisecond latency. A direct comparison to a Kyber-based GAKE ("Compiled Kyber") shows that Saber-GAKE outperforms Kyber in both computational and communication metrics for group sizes up to 2000 participants. While their work compellingly validates the practicality of Saber-KEM in static group settings, it does not address dynamic join/leave operations.

D. OTHER RELATED WORK

Samiullah et al. [15] review 48 IoT group key management schemes (2013-2023), categorizing them by cryptographic primitive, security property, architecture, and performance, and highlight the persistent challenge of efficient dynamic join/leave support in resource-constrained environments.

In a follow-up survey [16], the same authors examine post-quantum GKM protocols under ROM/QROM/MSU models, noting that dynamic group membership remains rudimentary is not yet integrated. These observations prompted us to design a GKE with dynamic support.

III. PRELIMINARIES

A. NOTATIONS

Let $\mathbb{Z}_q$ denote the integers modulo q, let pp be the public parameter using $\mathsf{Gen}(1^\lambda)$ and let $\mathsf{negl}(\lambda)$ be a negligible function in the security parameter λ.

B. RING-LWE

We work in the cyclotomic ring $R_q = \mathbb{Z}_q[x]/(f(x))$ of degree n. The RLWE problem asks, given $a \in R_q$ and $b = a \cdot s + e$ (with s, e sampled from an error distribution χ), to recover s. Since trapdoor operations incur significant computational complexity (Gaussian Sampling and Ring Multiplication), instantiating our protocol with RLWE effectively mitigates this overhead. First, the ring structure reduces both public key and ciphertext sizes from $O(n^2)$ elements to $O(n)$, yielding much smaller transmissions. Second, RLWE enables fast polynomial multiplication via NTT/FFT, reducing the overall complexity from $O(n^2)$ to $O(n \log n)$. These improvements translate into significant runtime and bandwidth savings in deployments.

C. DISCRETE GAUSSIAN DISTRIBUTION

Let $\sigma > 0$. The one-dimensional discrete Gaussian over $\mathbb{Z}$ with parameter σ is defined by

$$D_{\mathbb{Z},\sigma}(x) = \frac{1}{S(\sigma)} \exp\left(-\frac{x^2}{2\sigma^2}\right), \quad S(\sigma) = \sum_{y \in \mathbb{Z}} \exp\left(-\frac{y^2}{2\sigma^2}\right).$$

In this work, we sample each coordinate e_i of the error vector

$$e = (e_1, \ldots, e_m), \qquad e_i \overset{\text{i.i.d.}}{\sim} D_{\mathbb{Z},\sigma},$$

and then reduce each e_i modulo q to obtain the discrete Gaussian noise over $\mathbb{Z}_q$.

D. LATTICE TRAPDOORS

Based on the Micciancio-Peikert construction [6], let $\mathbf{A}_0 \in \mathbb{Z}_q^{n\times m'}$ and an invertible matrix $\mathbf{H} \in \mathbb{Z}_q^{n\times n}$. Invoking

$$(\mathbf{A}, \mathbf{R}) \leftarrow \mathsf{GenTrap}(\mathbf{A}_0, \mathbf{H})$$

yields a gadget-structured matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and its trapdoor $\mathbf{R} \in \mathbb{Z}_q^{m'\times w}$. The trapdoor satisfies

$$\|\mathbf{R}\| \le \sqrt{m}\,\omega(\sqrt{\log q}),$$

ensuring that preimage sampling produces sufficiently short vectors while preserving security. One can then efficiently sample a preimage

$$x \leftarrow \mathsf{SampleD}(\mathbf{A}, \mathbf{H}, \mathbf{R}, u, \sigma) \quad \text{s.t.} \quad \mathbf{A}x = u,$$

drawing x from a discrete Gaussian-like distribution. Without the trapdoor, solving $\mathbf{A}x = u$ reduces to the Inhomogeneous Short Integer Solution (ISIS) problem, which is conjectured to be hard under standard lattice assumptions.

E. ERROR EXPANSION FACTOR

Let $y_0, \ldots, y_{N-1} \in \mathbb{Z}^{m\times m}$ be the transformation matrices in each hop. We denote by

$$\|A\|_2 \;=\; \max_{\|v\|_2=1} \|A\,v\|_2$$

the *spectral norm* (operator 2-norm, equal to the largest singular value of A). Then the error expansion factor ρ is defined as

$$\rho \;=\; \max_{0\le j<N} \left\|y_j^T\right\|_2,$$

i.e. each hop satisfies $\|y_j^T\|_2 \le \rho$.

Lemma 1 (Single-Hop Error Propagation): Let $e \in \mathbb{Z}^n$ and $y_j \in \mathbb{Z}^{n\times n}$. If $e' = y_j^T e$, then

$$\|e'\| \le \|y_j^T\|\,\|e\|.$$

Lemma 2 (Spectral-Norm of the Product): If $\|y_j^T\| \le \rho$ for each $j = 0, \ldots, N-1$, then

$$\left\|\prod_{j=0}^{N-1} y_j^T\right\| \le \rho^N.$$

Lemma 3 (Gaussian Tail Bound [10], [11]): For $e_1 \sim D_{\mathbb{Z}^n,\sigma}$, we have

$$\Pr\left[\|e_1\| \ge \sigma\sqrt{n}\right] \;\le\; 2^{-n}.$$

IV. SECURITY MODEL

In this section, we adopt the game-based security framework for group key exchange protocols as formalized by Abdalla et al. [17] and Katz and Yung [8], which themselves generalize the two-party AKE model of Bellare and Rogaway [7]. Let $\mathcal{P} = \{P_0, P_1, \ldots, P_{N-1}\}$ be the set of N honest participants and λ be the security parameter. We denote the v-th instance of participant P_i by Π_i^v. The adversary $\mathcal{A}$ is any probabilistic polynomial-time (PPT) algorithm that interacts with the following oracles:

- **Send**(P_i, P_j, m) Simulates the delivery of message m from session instance Π_i^v at party P_i to instance Π_j^v at P_j, and returns the response generated by Π_j^v.
- **Execute**(Π_i^v) Returns the full communication transcript trans_i^v of session instance Π_i^v, assuming it interacts with other parties as defined by the protocol.
- **Reveal**(Π_i^v) Returns the session key sk_i^v of the completed session instance Π_i^v, or $\perp$ if the session has not been completed.
- **Corrupt**(P_i) Returns the secret trapdoor R_i of party P_i. (The ephemeral private key x_i^v used in session instance Π_i^v is deterministically derived from R_i and public parameters u.)
- **Test**(Π_i^v) Used in the KIND and FS experiments. On the first and only query to a *fresh* instance Π_i^v, this oracle returns either the real session key sk_i^v (if $b = 0$) or a random key $sk' \xleftarrow{\$} \{0,1\}^\lambda$ (if $b = 1$), depending on a hidden bit b.

We call a session instance *fresh* if it has not been subject to a prior Reveal, Corrupt, or Test query.

Two completed session instances Π_i^v and Π_j^v are said to *match* if they have identical message transcripts and the send events of one correspond to the receive events of the other.

A. CORRECTNESS

Experiment $\mathsf{Exp}^{\text{Correct}}_{\mathsf{GKE},\mathcal{A}}(\lambda)$**:**

1) **Setup & Protocol Execution:** Challenger runs $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda)$ and gives pp to $\mathcal{A}$. All honest parties execute GKE. Let

$$\mathcal{S} = \{\Pi_i^v \mid i \in \{0,1,\ldots,N{-}1\},\ v \in \mathbb{N},\ \Pi_i^v \text{ completed}\}$$

be the set of all completed instances.
2) **Queries:** $\mathcal{A}$ may adaptively issue Send, Execute, Reveal, and Corrupt queries in any order.
3) **Output:** If there exists a matching pair of completed, unrevealed, and uncorrupted instances $(\Pi_i^v, \Pi_j^v) \in \mathcal{S}$ such that

$$sk_i^v \neq sk_j^v,$$

then the experiment outputs 1; otherwise, it outputs 0.

We say GKE satisfies *Correctness* if for all PPT $\mathcal{A}$,

$$\Pr\left[\mathsf{Exp}^{\text{Correct}}_{\mathsf{GKE},\mathcal{A}}(\lambda) = 1\right] \le \mathsf{negl}(\lambda).$$

B. KEY INDISTINGUISHABILITY (KIND)

Key indistinguishability ensures that to an adversary who does not possess the secrets of the participants, the established session key is computationally indistinguishable from a truly random string.

Experiment $\mathsf{Exp}^{\mathrm{KIND}}_{\mathsf{GKE},\mathcal{A}}(\lambda)$**:**

1) **Setup & Protocol Execution:** Challenger runs $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda)$ and gives pp to $\mathcal{A}$. Honest parties may then execute GKE arbitrarily many times, resulting in a set
$$\mathcal{S} = \left\{\Pi_i^v \mid i \in \{0,1,\ldots,N-1\},\ v \in \mathbb{N},\ \Pi_i^v \text{ completed}\right\}$$
of all completed session instances.
2) **Queries:** $\mathcal{A}$ may issue any sequence of Send, Reveal, and $\mathsf{Corrupt}$ queries, subject to the restriction that the instance selected for Test must remain *fresh*. The adversary may invoke $\mathsf{Test}(\Pi_i^v)$ exactly once, for a fresh session instance $\Pi_i^v \in \mathcal{S}$.
3) **Challenge:** Challenger chooses $b \xleftarrow{\$} \{0,1\}$. On $\mathsf{Test}(\Pi_i^v)$, returns sk_i^v if $b = 0$, else returns $sk' \xleftarrow{\$} \{0,1\}^\lambda$.
4) **Guess:** $\mathcal{A}$ outputs guess b'. The experiment outputs 1 if $b' = b$, else 0.

Define the adversary's advantage
$$\mathsf{Adv}^{\mathrm{KIND}}_{\mathsf{GKE}}(\mathcal{A}) = \left|\Pr[\mathsf{Exp}^{\mathrm{KIND}}_{\mathsf{GKE},\mathcal{A}}(\lambda) = 1] - \tfrac{1}{2}\right|.$$
We say GKE satisfies *KIND* if for all PPT $\mathcal{A}$,
$$\mathsf{Adv}^{\mathrm{KIND}}_{\mathsf{GKE}}(\mathcal{A}) \le \mathsf{negl}(\lambda).$$

C. FORWARD SECRECY (FS)

Forward secrecy ensures that a participant who has left the group cannot compromise the confidentiality of session keys established after they departure.

Experiment $\mathsf{Exp}^{\mathrm{FS}}_{\mathsf{GKE},\mathcal{A}}(\lambda)$**:**

1) **Setup & Protocol Execution:** The challenger runs $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda)$ and gives pp to $\mathcal{A}$. Honest parties execute the GKE protocol, resulting in a set
$$\mathcal{S} = \left\{\Pi_i^v \mid i \in \{0,1,\ldots,N-1\}, v \in \mathbb{N},\ \Pi_i^v \text{ completed}\right\}.$$
2) **Corruption & Revocation:** $\mathcal{A}$ issues $\mathsf{Corrupt}(\Pi_i^v)$, obtaining R_i^v. Party P_i is then revoked, and the remaining honest parties perform protocol $\mathsf{GKE.Leave}$, yielding fresh session instances Π_j^r for each $P_j \neq P_i$.
3) **Queries & Challenge:** $\mathcal{A}$ may invoke Send and $\mathsf{Execute}$ on any unrevoked instance, and may invoke Test exactly once on a fresh rekeyed instance Π_j^v. Challenger picks $b \xleftarrow{\$} \{0,1\}$; on $\mathsf{Test}(\Pi_j^v)$ it returns the real sk_j^v if $b = 0$ or a random key $sk' \xleftarrow{\$} \{0,1\}^\lambda$ if $b = 1$.
4) **Guess:** $\mathcal{A}$ outputs a guess b'. The experiment outputs 1 if $b' = b$, and 0 otherwise.

Define the adversary's advantage
$$\mathsf{Adv}^{\mathrm{FS}}_{\mathsf{GKE}}(\mathcal{A}) = \left|\Pr[\mathsf{Exp}^{\mathrm{FS}}_{\mathsf{GKE},\mathcal{A}}(\lambda) = 1] - \tfrac{1}{2}\right|.$$
We say GKE satisfies *Forward Secrecy* if for all PPT $\mathcal{A}$,
$$\mathsf{Adv}^{\mathrm{FS}}_{\mathsf{GKE}}(\mathcal{A}) \le \mathsf{negl}(\lambda).$$

D. BACKWARD SECRECY (BS)

Backward secrecy ensures that a newly joined participant cannot compromise the confidentiality of past session keys established before they became a member of the group.

Experiment $\mathsf{Exp}^{\mathrm{BS}}_{\mathsf{GKE},\mathcal{A}}(\lambda)$:

1) **Setup & Protocol Execution:** The challenger runs $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda)$ and gives pp to $\mathcal{A}$. Honest parties execute the GKE protocol, resulting in a set
$$\mathcal{S} = \left\{\Pi_i^v \mid i \in \{0,1,\ldots,N-1\},\ v \in \mathbb{N},\ \Pi_i^v \text{ completed}\right\}.$$
2) **Join & Corruption:** A new participant P_k joins the group by executing the $\mathsf{GKE.Join}$ protocol. Immediately after the join protocol completes, the adversary $\mathcal{A}$ issues the query $\mathsf{Corrupt}(P_k)$ to obtain the secret trapdoor R_k of the new member.
3) **Queries & Challenge:** $\mathcal{A}$ may issue any sequence of Send, $\mathsf{Execute}$, and Reveal queries on any session instance, subject to freshness constraints. $\mathcal{A}$ then invokes Test exactly once on a fresh session instance Π_j^v that was completed before P_k joined the group. Challenger picks $b \xleftarrow{\$} \{0,1\}$; on $\mathsf{Test}(\Pi_j^v)$ it returns the real sk_j^v if $b = 0$ or a random key $sk' \xleftarrow{\$} \{0,1\}^\lambda$ if $b = 1$.
4) **Guess:** $\mathcal{A}$ outputs a guess b'. The experiment outputs 1 if $b' = b$, and 0 otherwise.

Define the adversary's advantage
$$\mathsf{Adv}^{\mathrm{BS}}_{\mathsf{GKE}}(\mathcal{A}) = \left|\Pr[\mathsf{Exp}^{\mathrm{BS}}_{\mathsf{GKE},\mathcal{A}}(\lambda) = 1] - \tfrac{1}{2}\right|.$$
We say GKE satisfies *Backward Secrecy* if for all PPT $\mathcal{A}$,
$$\mathsf{Adv}^{\mathrm{BS}}_{\mathsf{GKE}}(\mathcal{A}) \le \mathsf{negl}(\lambda).$$

V. GROUP KEY EXCHANGE PROTOCOL

In this section, we introduce our group key exchange protocol GKE. The protocol involves two security parameters, λ and ρ, where λ is the computational security parameter used to control the protocol's resistance against attacks, and ρ is the statistical security parameter, used in the correctness and stability proofs of the protocol. $\sigma = (\sigma_1, \sigma_2)$ represents the standard deviations of the discrete Gaussian distributions, which are used to generate the secret key χ_s and the error term χ_e.

In the group key exchange setting, there are N participants indexed as $P_0, P_1, \ldots, P_{N-1}$. Assume that P_0 is the initiator of the session. Participants will complete the key exchange protocol using two functions, $\mathsf{GenTrap}$ and $\mathsf{SampleD}$, and ultimately compute a shared common key. During the protocol, participants communicate their parameters via broadcast, and others relay the information in cyclically. This approach not only reduces communication costs but also slightly enhances security. Algorithm 1 defines the

Setup subroutine that generates the public/private key pair and trapdoor from the target vector u. Algorithm 2 defines the GenKey subroutine that samples s, e_1, e_2, computes the ciphertext pair (C_1^0, C_2). Algorithm 3 defines the RecoverKey subroutine that derives the shared key sk from the intermediate value m. Algorithm 4 then presents the full protocol GKE, invoking Setup, GenKey, and RecoverKey. The public parameters are $pp = (N, R_q, \sigma)$, where $R_q = \mathbb{Z}[x]/(x^n + 1)$.

Algorithm 1 Setup(P_i, u, σ)

Input: Party P_i, target u, noise distribution σ
Output: A_i, x_i, R_i

begin
1. Sample $A_i^0 \leftarrow \mathbb{Z}_q^{n \times m'}$
2. Sample invertible $H_i \leftarrow \mathbb{Z}_q^{n \times n}$
3. $(A_i, R_i) \leftarrow$ GenTrap(A_i^0, H_i)
4. $x_i \leftarrow$ SampleD$(A_i, R_i, H_i, u, \sigma)$

return (A_i, x_i, R_i)

Algorithm 2 GenKey$(A_0,\ u,\ \chi_s,\ \chi_e,\ q)$

Input: A_0, target vector u, Gaussian samplers χ_s, χ_e, modulus q
Output: $C_1^0,\ C_2$

begin
1. Sample $s \leftarrow \chi_s$
2. Sample $e_1, e_2 \leftarrow \chi_e$
3. Randomly select $sk \in \{0, 1\}$
4. Compute $C_1^0 = (A_0)^T s + e_1 \mod q$
5. Compute $C_2 = u^T s + e_2 + sk \cdot \lfloor \frac{q}{2} \rfloor \mod q$

return $(C_1^0,\ C_2)$

Algorithm 3 RecoverKey$(m,\ q)$

Input: message m, modulus q
Output: shared key sk

if $(m < q/4)$ or $(m > 3q/4)$ **then**
  $sk \leftarrow 0$
else
  $sk \leftarrow 1$
return sk

Upon completing the protocol, all P_i obtain the same shared secret key sk.

A. MEMBER JOIN AND LEAVE

In GKE protocols, dynamic changes in membership significantly impact both security and efficiency. Member joining and leaving directly influence the group key update mechanism and determine security guarantees such as forward secrecy and backward secrecy. The procedures for member join and member leave are detailed in Algorithm 5 and Algorithm 6, respectively.

Algorithm 4 GKE$\big(P[0, 1, \ldots, N-1],\ N,\ R_q,\ \sigma\big)$

Input: $P[0, 1, \ldots, N-1], N, R_q, \sigma$

Round 1 (for party P_0 only):
1. Select $u \xleftarrow{\$} R_q^n$
2. Broadcast u

Round 2 (for each party P_i, $i \in \{0, 1, \ldots, N-1\}$ in parallel):
1. $(A_i, x_i, R_i) \leftarrow$ Setup(P_i, u, σ)
2. Send A_i to P_{i-1} and P_{i+1}

Round 3 (for each party P_i, $i \in \{0, 1, \ldots, N-1\}$ in parallel):
1. Compute y_i^{i+1} using R_i s.t. $A_i y_i^{i+1} = A_{i+1}$
2. Compute y_i^{i-1} using R_i s.t. $A_i y_i^{i-1} = A_{i-1}$
3. Send y_i^{i+1} to P_{i+1} and y_i^{i-1} to P_{i-1}

Round 4 (for party P_0 only):
1. $(C_1^0, C_2) \leftarrow$ GenKey$(A_0,\ u,\ \chi_s,\ \chi_e,\ q)$
2. Send C_1^0 to P_{N-1} and P_1
3. Broadcast C_2

Round 4 + ℓ, $\ell \in \{1, 2, \ldots, \lfloor N/2 \rfloor\}$ (for each party P_i, $i \in \{1, \ldots, N-1\}$ in parallel):
if *receive* C_1^{i-1} **then**
1. Compute $C_1^i = (y_{i-1}^i)^T C_1^{i-1}$
2. Send C_1^i to P_{i+1}

else if *receive* C_1^{i+1} **then**
1. Compute $C_1^i = (y_{i+1}^i)^T C_1^{i+1}$
2. Send C_1^i to P_{i-1}

Key Computation (for each party P_i, $i \in \{0, \ldots, N-1\}$):
1. Compute $m = C_2 - (x_i)^T C_1^i \mod q$
2. $sk \leftarrow$ RecoverKey(m, q)

1) MEMBER JOIN

Suppose the current group consists of k participants, denoted as $\{P_0, P_1, \ldots, P_{k-1}\}$. If a new member P_k wants to join,, the group key must be updated to ensure forward secrecy.

First, P_0, P_k and P_{k-1} must execute Setup to refresh their new key pair and trapdoor (A, x, R) from the public parameter u, satisfying:

$$Ax = u$$

and send the new A_j to the its neighbor. Next, each party $P_j, j \in \{1, 0, k, k-1, k-2\}$ exchanges and computes new public-key transformation values y with their respective neighbors, so that P_1, P_0, P_k, P_{k-1} and P_{k-2} all obtain updated y.

Finally, P_0 reruns the GenKey to propagate the new key, completing the group rekeying process; all members, including the newly joined participant, then derive the new key following the original protocol.

Algorithm 5 GKE.Join$\big(P[0,1,\ldots,k-1],\ P_k,\ u,\ \sigma\big)$

Input: Current Group $P[0,1,\ldots,k-1]$, new member P_k, u, σ

Round 1 (for each party P_v, $v \in \{0, k-1, k\}$ in parallel):
1. $(A_v, x_v, R_v) \leftarrow \mathsf{Setup}(P_v, u, \sigma)$
2. P_0 send A_0 to P_1 and P_k
3. P_k send A_k to P_0 and P_{k-1}
4. P_{k-1} send A_{k-1} to P_k and P_{k-2}

Round 2 (for each party P_w, $w \in \{1, 0, k, k-1, k-2\}$ in parallel):
1. P_1 computes y_1^0 using R_1, sends it to P_0
2. P_0 computes y_0^1 and y_0^k using R_0, sends the first to P_1, the second to P_k
3. P_k computes y_k^0 and y_k^{k-1} using R_k, sends the first to P_0, the second to P_{k-1}
4. P_{k-1} computes y_{k-1}^k and y_{k-1}^{k-2} using R_{k-1}, sends the first to P_k, the second to P_{k-2}
5. P_{k-2} computes y_{k-2}^{k-1} using R_{k-2}, sends it to P_{k-1}

Round 3 (for party P_0 only):
1. $(C_1^0, C_2) \leftarrow \mathsf{GenKey}(A_0,\ u,\ \chi_s,\ \chi_e,\ q)$
2. Send C_1^0 to P_k and P_1
3. Broadcast C_2

Round $3+\ell$, $\ell \in \{1, 2, \ldots, \lfloor (k+1)/2 \rfloor\}$ (for each party P_i, $i \in \{1, \ldots, k\}$ in parallel):
- **if** *receive* C_1^{i-1} **then**
  1. Compute $C_1^i = (y_{i-1}^i)^T C_1^{i-1}$
  2. Send C_1^i to P_{i+1}
- **else if** *receive* C_1^{i+1} **then**
  1. Compute $C_1^i = (y_{i+1}^i)^T C_1^{i+1}$
  2. Send C_1^i to P_{i-1}

Key Computation (for each party P_i, $i \in \{0, \ldots, k\}$):
1. Compute $m = C_2 - (x_i)^T C_1^i \bmod q$
2. $sk \leftarrow \mathsf{RecoverKey}(m, q)$

2) MEMBER LEAVE

Similarly, suppose the group currently consists of k participants. If the l-th member (P_l) leaves the group, the group key must be updated to ensure forward secrecy.

First, P_{l-1} and P_{l+1} invoke the Setup to refresh their key pairs and trapdoor (A, x, R). Next, each of them exchanges and computes new public-key transformation values y with each other and with their respective neighbors, so that P_{l-2}, P_{l-1}, P_{l+1}, and P_{l+2} all obtain updated y. Excluding the departed member P_l, P_{l-1}'s next neighbor is now P_{l+1}, and P_{l+1}'s previous neighbor is now P_{l-1}.

Finally, P_0 regenerates the key and repropagates it. Meanwhile, the leaving member P_l has their key pair permanently revoked and can no longer be used.

Algorithm 6 GKE.Leave$\big(P[0,1,\ldots,k-1],\ P_l,\ u,\ \sigma\big)$

Input: Current group $P[0,1,\ldots,k-1]$, departing member P_l, u, σ

Round 1 (for each party P_v, $v \in \{l-1, l+1\}$) in parallel:
1. $(A_v, x_v, R_v) \leftarrow \mathsf{Setup}(P_v, u, \sigma)$
2. P_{l-1} sends A_{l-1} to P_{l-2} and P_{l+1}
3. P_{l+1} sends A_{l+1} to P_{l+2} and P_{l-1}

Round 2 (for each party P_w, $w \in \{l-2, l-1, l+1, l+2\}$) in parallel:
1. P_{l-2} computes y_{l-2}^{l-1} using R_{l-2}, sends it to P_{l-1}
2. P_{l-1} computes y_{l-1}^{l-2} and y_{l-1}^{l+1} using R_{l-1}, sends the first to P_{l-2}, the second to P_{l+1}
3. P_{l+1} computes y_{l+1}^{l-1} and y_{l+1}^{l+2} using R_{l+1}, sends the first to P_{l-1}, the second to P_{l+2}
4. P_{l+2} computes y_{l+2}^{l+1} using R_{l+2}, sends it to P_{l+1}

Round 3 (for party P_0 only):
1. $(C_1^0, C_2) \leftarrow \mathsf{GenKey}(A_0,\ u,\ \chi_s,\ \chi_e,\ q)$
2. Send C_1^0 to P_{k-1} and P_1
3. Broadcast C_2

Round $3+\ell$, $\ell \in \{1, 2, \ldots, \lfloor (k-1)/2 \rfloor\}$ (for each party P_i, $i \in \{1, \ldots, l-1, l+1, \ldots, k-1\}$) in parallel:
- **if** *receive* C_1^{i-1} **then**
  1. Compute $C_1^i = (y_{i-1}^i)^T C_1^{i-1}$
  2. Send C_1^i to P_{i+1}
- **else if** *receive* C_1^{i+1} **then**
  1. Compute $C_1^i = (y_{i+1}^i)^T C_1^{i+1}$
  2. Send C_1^i to P_{i-1}

Key Computation (for each party P_i, $i \in \{0, \ldots, k-1\}$):
1. Compute $m = C_2 - (x_i)^T C_1^i \bmod q$
2. $sk \leftarrow \mathsf{RecoverKey}(m, q)$

VI. SECURITY PROOFS

In this section, we provide formal security proofs for our GKE protocol. The proofs follow the game-based security framework defined in Section IV. We will demonstrate that the protocol achieves Key Indistinguishability (KIND), Forward Secrecy (FS), and Backward Secrecy (BS) under the assumption that the Ring Learning With Errors (RLWE) problem is hard.

A. CORRECTNESS

Theorem 1: [Correctness] Suppose the parameters satisfy

$$\rho^N \sigma \sqrt{n} \le \beta \quad \text{and} \quad B_2 + \|x_i\|\,\beta < \frac{q}{4}.$$

Then all honest parties in protocol GKE recover the same session key sk except with negligible probability.

Proof Sketch: By Lemmas 1, 2, 3 (see Preliminaries) the accumulated error $\|e'\|$ is at most $\rho^N \sigma \sqrt{n} \le \beta$, and the final decryption error $|e_2 - x_i^T e'| \le B_2 + \|x_i\|\beta < q/4$, so the

usual LWE rounding succeeds. The full, detailed proof is in Appendix. □

B. PROOF OF KEY INDISTINGUISHABILITY (KIND)

We prove the key indistinguishability of our GKE protocol through a sequence of games. Let $\mathsf{Adv}_{\mathcal{A}}^{Game\ i}(\lambda)$ be the advantage of a probabilistic polynomial-time adversary $\mathcal{A}$ in winning Game i for a security parameter λ.

- **Game 0:** This is the original KIND experiment $\mathsf{Exp}_{\mathsf{GKE},\mathcal{A}}^{\mathsf{KIND}}(\lambda)$ as defined in Section IV-B. By definition, the adversary's advantage is $\mathsf{Adv}_{\mathsf{GKE},\mathcal{A}}^{\mathsf{KIND}}(\lambda) = \mathsf{Adv}_{\mathcal{A}}^{Game\ 0}(\lambda)$.
- **Game 1:** This game is identical to Game 0, except that the ciphertext (C_1^0, C_2) generated by the P_0 is replaced with a tuple chosen uniformly at random from $R_q^m \times R_q$. This transition is formally justified by the security of the underlying RLWE-based encryption scheme. The GenKey procedure (Algorithm 2), where P_0 computes $(C_1^0, C_2) = ((A_0)^T s + e_1, u^T s + e_2 + sk \cdot \lfloor q/2 \rfloor)$, is a standard construction of an RLWE public-key encryption scheme. The security of such a scheme is defined as Indistinguishability under Chosen-Plaintext Attack (IND-CPA).
  The IND-CPA property, which is based on the hardness of the RLWE problem, guarantees that a ciphertext generated from any specific message (in this case, the session key sk) is computationally indistinguishable from a random element in the ciphertext space. Therefore, an adversary cannot distinguish Game 1 (with a random ciphertext) from Game 0 (with the real ciphertext). The difference in advantage for any PPT adversary $\mathcal{A}$ is negligible:
  $$|\mathsf{Adv}_{\mathcal{A}}^{Game\ 0}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{Game\ 1}(\lambda)| \leq \mathsf{negl}(\lambda)$$
- **Game 2:** This game is identical to Game 1. We only analyze the adversary's advantage from a different perspective. In this game, the tuple (C_1^0, C_2) is completely random and independent of all other protocol parameters, including the session key sk. When the adversary $\mathcal{A}$ makes its single query to the $\mathsf{Test}(\Pi_i^v)$ oracle, the value it receives (either the real key sk_i^v or a random key sk') is information-theoretically independent of the adversary's view (the transcript).
  Therefore, the adversary cannot gain any information about the hidden bit b. The probability of the adversary guessing b correctly is exactly $1/2$. Thus, its advantage is zero.
  $$\mathsf{Adv}_{\mathcal{A}}^{Game\ 2}(\lambda) = |Pr[b' = b] - 1/2| = |1/2 - 1/2| = 0$$

Conclusion: Combining the arguments above, we have:

$$\begin{aligned}\mathsf{Adv}_{GKE,\mathcal{A}}^{KIND}(\lambda) &= \mathsf{Adv}_{\mathcal{A}}^{Game\ 0}(\lambda)\\ &\approx \mathsf{Adv}_{\mathcal{A}}^{Game\ 1}(\lambda)\\ &= \mathsf{Adv}_{\mathcal{A}}^{Game\ 2}(\lambda)\\ &= 0\end{aligned}$$

Thus, the advantage of any PPT adversary in breaking the key indistinguishability of the protocol is negligible. This concludes the proof of KIND security.

C. PROOF OF FORWARD SECRECY (FS)

The proof for Forward Secrecy is established by analyzing the GKE.Leave procedure within the context of the $\mathsf{Exp}_{\mathsf{GKE},\mathcal{A}}^{\mathrm{FS}}(\lambda)$ experiment. In this scenario, an adversary $\mathcal{A}$ has corrupted a departing member P_l and possesses its secret trapdoor R_l. The adversary's goal is to use this information to compromise a new session key sk' established after P_l has left the group.

Our proof hinges on the fact that the compromised information becomes obsolete due to the key update mechanism. The adversary's knowledge of R_l implies they also know the re-encryption matrices that P_l used in the previous session, specifically y_{l-1}^l and y_{l+1}^l. These matrices were valid only for the old set of public keys, satisfying relations such as $A_{l-1}y_{l-1}^l = A_l$. They allowed P_l to transform a received ciphertext, for instance C_1^{l-1}, into its own required ciphertext C_1^l via the operation $C_1^l = (y_{l-1}^l)^T C_1^{l-1}$, which can then be used to decrypt sk.

The core of the forward secrecy guarantee is as follows:

1) **Key Invalidation:** When P_l leaves, the GKE.Leave protocol mandates that its neighbors, P_{l-1} and P_{l+1}, generate fresh key pairs and trapdoors. Let the new public key of P_{l-1} be $(A_{l-1})'$. The old re-encryption matrix y_{l-1}^l now becomes useless because the foundational relationship is broken: $(A_{l-1})' y_{l-1}^l \neq A_l$.
2) **Ciphertext Flow Disruption:** In the new key exchange session, the initiator P_0 generates a fresh ciphertext tuple $((C_1^0)', (C_2)')$ based on new ephemeral secrets. When the re-encrypted ciphertext $(C_1^{l-1})'$ (which is based on the new public key $(A_{l-1})'$) is propagated, the adversary, acting as the departed P_l, cannot use their obsolete matrix y_{l-1}^l to correctly compute the new required ciphertext $(C_1^l)'$. Applying the old transformation would result in an incorrect value that does not correspond to the new protocol execution.
3) **Inability to Decrypt:** Since the adversary cannot compute the correct intermediate ciphertext $(C_1^l)'$, they cannot correctly compute the message $m' = (C_2)' - (x_l)^T (C_1^l)' \pmod q$. Consequently, they cannot derive the new session key sk'.

In summary, even though the adversary possesses the secrets trapdoor of a past member, the protocol ensures these secrets are invalidated for future sessions by updating the cryptographic context (i.e., the public keys of neighbors). This breaks the re-encryption chain for the departed member and guarantees forward secrecy. Thus, the adversary's advantage in the FS experiment remains negligible.

$$\mathsf{Adv}_{\mathsf{GKE},\mathcal{A}}^{\mathrm{FS}}(\lambda) \leq \mathsf{negl}(\lambda)$$

D. PROOF OF BACKWARD SECRECY (BS)

The proof for Backward Secrecy is demonstrated by analyzing the $\mathsf{Exp}_{\mathsf{GKE},\mathcal{A}}^{BS}(\lambda)$ experiment. An adversary $\mathcal{A}$

compromises a new member P_k immediately after it joins, obtaining its secret trapdoor R_k. The adversary's goal is to use this information to decrypt a session key sk from a session that was completed before P_k joined. The proof's foundation is that the key update mechanism during the join process renders the new member's keys incompatible with any prior session's cryptographic data.

1) **Decryption Mechanics in a Past Session**: In any given session, a participant P_i receives an intermediate ciphertext, for instance C_1^{i-1} from its neighbor P_{i-1}. To correctly continue the re-encryption chain and eventually decrypt the final session key, P_i must compute its own intermediate ciphertext C_1^i. According to Algorithm 4, this is done via the transformation: $C_1^i = (y_{i-1}^i)^T C_1^{i-1}$. The crucial element here is the transformation matrix y_{i-1}^i, which is computed by P_{i-1} using its trapdoor R_{i-1} to satisfy the relation $A_{i-1}y_{i-1}^i = A_i$, where A_i and A_{i-1} are the public keys from that specific past session.
2) **Impact of the Join Protocol**: When the new member P_k joins, it is placed between P_0 and P_{k-1}. According to the GKE.Join protocol (Algorithm 5), this event forces the neighboring members P_0 and P_{k-1} to execute the Setup procedure again. This results in them generating new public keys, let's denote them as $(A_0)'$ and $(A_{k-1})'$, which replace their old keys from the previous session. The new member P_k also generates its own public key, A_k.
3) **The Adversary's Inability to Decrypt**: The adversary having compromised P_k using $\mathsf{Execute}(\Pi_k^v)$ and possesses its transcript trans_k^v. This allows the adversary to gain the transformation matrices from its new neighbors, such as y_0^k which satisfies $(A_0)'y_0^k = A_k$.
The adversary's goal is to compute the correct intermediate ciphertext $(C_1^k)_{old}$ corresponding to the past session. To attempt this, the adversary must apply their newly derived transformation matrix to a ciphertext from that old session, for instance, $(C_1^0)_{old}$. The attempted computation would be $(C_1^k)_{attempt} = (y_0^k)^T(C_1^0)_{old}$.
This attempt will fail. The transformation matrix y_0^k was computed based on the new public key $(A_0)'$. However, the old ciphertext $(C_1^0)_{old}$ belongs to a cryptographic context defined by the old public key A_0. Because $A_0 \neq (A_0)'$, the transformation is invalid. Applying a transformation matrix derived from a new set of public keys to a ciphertext derived from an old set does not produce a valid intermediate ciphertext for the old session. The required mathematical relationship is broken by the key update.
Thus, the adversary cannot compute the correct $(C_1^k)_{old}$.

Since the adversary is unable to derive the correct intermediate ciphertext C_1^k for the past session, the final decryption step, $m = C_2 - (x_k)^T C_1^k$, will produce an incorrect result. Consequently, the adversary cannot recover the old session key sk. This demonstrates that the protocol achieves Backward Secrecy.

$$\mathsf{Adv}^{\mathrm{BS}}_{\mathsf{GKE},\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$$

E. RESILIENCE AGAINST ACTIVE ADVERSARIES

The security proofs provided above establish the protocol's security against a passive (eavesdropping) adversary. To achieve security against active adversaries who can intercept, modify, and inject messages, our protocol can be strengthened using standard cryptographic techniques.

A common method to upgrade a passively secure key exchange protocol to an actively secure one is to apply a compiler, such as the one implicitly used by Katz and Yung [8]. This typically involves having each participant digitally sign the messages they send. By authenticating all communication flows, any unauthorized modification of messages by an adversary can be detected.

Specifically, each party P_i would use a post-quantum digital signature scheme to sign its messages, such as the public matrix A_i and the transformation matrix y_i. Upon receiving a message, the recipient verifies the signature before processing it. If any verification fails, the protocol is aborted. This ensures that an adversary cannot impersonate another participant or tamper with the keying material without possessing the legitimate participant's signing key.

While a full formalization of the actively secure variant is beyond the scope of this paper's primary contributions, the path to achieving it is standard and relies on well-established cryptographic principles. Our protocol's core design, which focuses on efficient dynamic membership, serves as a solid foundation for such an extension.

VII. COMPARISON WITH OTHER LATTICE-BASED GKE PROTOCOLS

Table 1 compares our proposed scheme with three existing lattice-based group key exchange protocols. The evaluation criteria include scalability, communication rounds, dynamic group, computation complexity and the number of rounds required for members to join or leave.

A. ANALYSIS OF THE COMPARISON

Table 1 highlights key trade-offs among representative lattice-based GKE protocols. Yang et al.'s ring-based construction uses only two broadcasts but assumes a fixed group, and depends on a semi-trusted collector for secure unicast, making it impractical for fully decentralized settings. Apon et al. achieve constant-round rekeying via and generic reconciliation, but still do not support member join or leave. Samiullah et al. further optimizes to four constant rounds and concrete authentication in the QROM, yet likewise treats the group as static.

While our key-exchange phase requires $4 + \lfloor N/2 \rfloor$ interaction rounds, each participant's computation and communication remain constant (i.e. $O(1)$ per-party). More

TABLE 1. Comparison of computation complexity and features.

Method	Yang et al. [13]	Apon et al. [5]	Samiullah et al. [14]	Our
Scalability	$O(1)$	$O(1)$	$O(1)$	$O(1)$
Communication Round	2	3	4	$4+\lfloor\frac{N}{2}\rfloor$
Dynamic Setting	X	X	X	**O**
Gaussian Sampling	2	$3+\frac{1}{N}$	5	$(m'w)+\frac{3}{N}$
Ring Multiplication	$2+\frac{2}{N}$	$2+\frac{1}{N}$	6	$(2m+7)+\frac{2}{N}$
Member Join & Leave	-	-	-	$3+\lfloor\frac{N}{2}\rfloor$

- **Scalability**: O indicates that each participant's computational and communication workload remains constant as the group size grows; X means it does not.
- **Communication Rounds**: Total number of interactive rounds required for the key-exchange phase.
- **Dynamic Setting**: O indicates support for dynamic member join/leave; X means the protocol does not support dynamic changes.
- **Gaussian Sampling, Ring Multiplication**: Average number of online Gaussian-sampling calls and ring-multiplications per participant in a single protocol execution.
- **Member Join & Leave**: "-" indicates the operation is not applicable; numeric entries give the number of rounds required to perform that operation.

importantly, our protocol simultaneously achieves fully dynamic membership. Any party can join and leave in $3+\lfloor N/2\rfloor$ rounds, while preserving both forward and backward secrecy.

Due to the trapdoor mechanism, each participant's key-pair generation incurs a fixed cost proportional to the trapdoor dimension $m'\times\omega$, which is inherent and unavoidable, however, this is typically a one-time or infrequent operation. In the computation of each matrix y, we invoke SampleD to sample one column of length m, which requires 1 online ring multiplications. Since $y\in\mathbb{Z}_q^{m\times m}$ has m columns, constructing one full y costs m multiplications. Each participant must compute y for both its two neighbors, resulting in $2m$ multiplications per participant. All other minor overhead is RLWE key generation and computation of the encryption and decryption steps.

Moreover, our protocol is the lattice-based GKE to combine both full dynamic join/leave support and constant-state per round. This fills a crucial gap between fully static, highly scalable constructions and single-round but static schemes. We thus target scenarios where runtime performance is secondary to the ability to securely and efficiently rekey a live group under post-quantum assumptions.

VIII. CONCLUSION AND FUTURE WORK

In this work, we proposed a dynamic Group Key Exchange (GKE) protocol for a cyclic group topology, based on lattice trapdoors and re-encryption techniques. By leveraging the strong mathematical structure of lattice-based trapdoors, we designed a secure and efficient key distribution mechanism. Furthermore, the integration of re-encryption enhances the protocol's flexibility in key updates and delegation, improving its overall applicability and practicality.

This research introduces a novel GKE model with resistance to quantum attacks while also considering computational efficiency and security analysis. Compared to traditional number-theoretic approaches, our protocol offers better scalability and adaptability, making it a promising candidate for secure communication in the post-quantum era.

For future work, the key generation process can be further refined by shifting from a model where P_0 determines the key to a collaborative computation approach involving all participants. This design not only enhances decentralization and security, but also serves as a foundation for more efficient communication architectures. For instance, by adopting a tree-based aggregation and dissemination structure, the communication rounds can be reduced from $O(N)$ to $O(\log N)$. Furthermore, to achieve a truly constant-round protocol, a key future direction would be to extend our GKE protocol to a broadcast-based system. In such a model, all participants could broadcast their keying material, potentially in parallel, allowing the key agreement to complete in a fixed number of rounds independent of the group size. This adaptation would be critical for realizing a constant-round solution.

Further optimizations to the protocol's implementation can be achieved by first investigating the integration of our trapdoor-enabled re-encryption primitive into existing lattice libraries such as Kyber, NTRU and FrodoKEM. If this proves feasible, we will develop a prototype within these frameworks and experimental evaluations to assess computational overhead and validate real-world performance. Security analysis can be strengthened by validating the protocol under standard models or stricter security assumptions, particularly in the presence of an active adversary. Another potential direction is to explore its applicability in extended scenarios such as blockchain systems, decentralized identity authentication, and secure multi-party computation.

In conclusion, this work presents an innovative GKE protocol based on lattice cryptography, offering a viable solution for secure communication in the post-quantum era. With potential optimizations in key generation, implementation efficiency, security analysis, and extended applications, future research will focus on the practical feasibility and security optimization of the protocol to ensure its real-world applicability.

APPENDIX
CORRECTNESS PROOF

Proof of Theorem 1: The goal is to show that for any honest participant P_i, the final decryption step correctly recovers the session key sk with overwhelming probability.

In our multi-hop protocol, the first-hop ciphertext from P_{i-1} is

$$C_1^{i-1}=(A_{i-1})^T s+e_1,$$

and applying the transform $(y_{i-1}^i)^T$ yields the new ciphertext for the next hop. This process is detailed as follows:

$$\begin{aligned}C_1^i&=(y_{i-1}^i)^T C_1^{i-1}\\&=(y_{i-1}^i)^T\left((A_{i-1})^T s+e_1\right)\end{aligned}$$

$$= (y_{i-1}^i)^T (A_{i-1})^T s \; + \; (y_{i-1}^i)^T e_1$$
$$= (A_i)^T s \; + \; (y_{i-1}^i)^T e_1.$$

Iterating this transformation over N hops defines the accumulated error term in the final ciphertext received by any party P_i:

$$e' \; = \; \left(\prod_{j=0}^{N-1} y_j^T \right) e_1.$$

Thus, the final decryption at participant P_i computes the message m as:

$$m = sk \cdot \left\lfloor \frac{q}{2} \right\rfloor + \left(e_2 - x_i^T e'\right) \pmod q.$$

For correct decryption, the noise term $\left(e_2 - x_i^T e'\right)$ must be smaller than $q/4$. We proceed to bound its magnitude.

From 2, we know that the norm of the product of transformation matrices is bounded by $\| \prod_{j=0}^{N-1} y_j^T \| \leq \rho^N$.

From our updated 3, the norm of the initial error vector $e_1 \sim D_{\mathbb{Z}^n,\sigma}$ is bounded by $\|e_1\| < \sigma\sqrt{n}$ with probability at least $1 - 2^{-n}$. This probability is overwhelmingly high for any cryptographic value of n.

Combining these, we can bound the norm of the accumulated error $\|e'\|$ over N hops:

$$\|e'\| \leq \left\| \prod_{j=0}^{N-1} y_j^T \right\| \cdot \|e_1\| \leq \rho^N \sigma \sqrt{n}$$

This inequality holds with probability at least $1 - 2^{-n}$.

We can now bound the magnitude of the final noise term using the triangle inequality and Cauchy-Schwarz inequality:

$$|e_2 - x_i^T e'| \leq \|e_2\| + \|x_i\| \|e'\|$$

Let's assume the norm of the second error term $\|e_2\|$ is bounded by B_2 (a standard assumption for LWE noise). The condition for correct decryption becomes:

$$B_2 + \|x_i\| \cdot \|e'\| < \frac{q}{4}$$

Substituting our bound for $\|e'\|$, we get the final condition:

$$B_2 + \|x_i\| \left(\rho^N \sigma \sqrt{n}\right) < \frac{q}{4}$$

This matches the condition given in **Theorem 1** (after setting $\beta = \rho^N \sigma \sqrt{n}$). Since this condition holds by the theorem's hypothesis, the noise term is guaranteed to be smaller than $q/4$, and the standard LWE rounding in 'RecoverKey' correctly recovers sk. This holds for all honest participants, meaning they all derive the same session key, except with a negligible probability of 2^{-n} from the tail bound. This concludes the proof of correctness. □

REFERENCES

[1] I. Ingemarsson, D. Tang, and C. Wong, "A conference key distribution system," *IEEE Trans. Inf. Theory*, vol. IT-28, no. 5, pp. 714–720, May 1982.
[2] M. Steiner, G. Tsudik, and M. Waidner, "Diffie–Hellman key distribution extended to group communication," in *Proc. 3rd ACM Conf. Comput. Commun. Secur.*, 1996, pp. 31–37.
[3] M. Burmester and Y. Desmedt, "A secure and efficient conference key distribution system," in *Proc. Workshop Theory Appl. Cryptograph. Techn.*, Perugia, Italy. Cham, Switzerland: Springer, 1994, pp. 275–286.
[4] J. Zhang, Z. Zhang, J. Ding, M. Snook, and Ö. Dagdelen, "Authenticated key exchange from ideal lattices," in *Proc. 34th Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, Sofia, Bulgaria. Cham, Switzerland: Springer, 2015, pp. 719–751.
[5] D. Apon, D. Dachman-Soled, H. Gong, and J. Katz, "Constant-round group key exchange from the ring-lwe assumption," in *Proc. 10th Int. Conf. Post-Quantum Cryptogr.*, Chongqing, China. Switzerland: Springer, 2019, pp. 189–205.
[6] D. Micciancio and C. Peikert, "Trapdoors for lattices: Simpler, tighter, faster, smaller," in *Proc. Annual Int. Conf. Theory Appl. Cryptograph. Techn.* Cham, Singapore: Springer, 2012, pp. 700–718.
[7] M. Bellare and P. Rogaway, "Entity authentication and key distribution," in *Proc. Annu. International Cryptol. Conf.* Cham, Switzerland: Springer, 1993, pp. 232–249.
[8] J. Katz and M. Yung, "Scalable protocols for authenticated group key exchange," in *Proc. Annu. Int. Cryptol. Conf.* Cham, Switzerland: Springer, 2003, pp. 110–125.
[9] K. Cohn-Gordon, C. Cremers, and L. Garratt, "On post-compromise security," in *Proc. IEEE 29th Comput. Secur. Found. Symp. (CSF)*, Jun. 2016, pp. 164–178.
[10] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *J. ACM*, vol. 56, no. 6, pp. 1–40, Sep. 2009.
[11] C. Peikert, "Public-key cryptosystems from the worst-case shortest vector problem," in *Proc. 41st Annu. ACM Symp. Theory Comput.*, May 2009, pp. 333–342.
[12] J. Ding, X. Xie, and X. Lin. (2012). *A Simple Provably Secure Key Exchange Scheme Based on the Learning with Errors Problem*. Accessed: Jan. 29, 2023. [Online]. Available: https://eprint.iacr.org/2012/688
[13] X. Yang, W. Ma, and C. Zhang, "Group authenticated key exchange schemes via learning with errors," *Secur. Commun. Netw.*, vol. 8, no. 17, pp. 3142–3156, Nov. 2015.
[14] F. Samiullah, M.-L. Gan, S. Akleylek, and Y. Aun, "Quantum resistance saber-based group key exchange protocol for IoT," *IEEE Open J. Commun. Soc.*, vol. 6, pp. 378–398, 2025, doi: 10.1109/OJCOMS.2024.3516005.
[15] F. Samiullah, M.-L. Gan, S. Akleylek, and Y. Aun, "Post-quantum group key management in IoTs," in *Proc. 25th Int. Multitopic Conf. (INMIC)*, Lahore, Pakistan, Nov. 2023, pp. 1–6, doi: 10.1109/inmic60434.2023.10466001.
[16] F. Samiullah, M.-L. Gan, S. Akleylek, and Y. Aun, "Group key management in Internet of Things: A systematic literature review," *IEEE Access*, vol. 11, pp. 77464–77491, 2023, doi: 10.1109/ACCESS.2023.3298024.
[17] B. Poettering, P. Rösler, J. Schwenk, and D. Stebila, "SoK: Game-based security models for group key exchange," in *Proc. Cryptographers' Track RSA Conf.* Cham, Switzerland: Springer, pp. 148–176.

NARN-YIH LEE (Member, IEEE) received the Ph.D. degree in information engineering from National Cheng Kung University, Taiwan, in 1996. He is currently a Professor with the Computer and Network Center, National Cheng Kung University. His research interests include cryptography, blockchain, network security, post-quantum cryptography, and chip security.

YUAN-TING TSAI received the B.S. degree in computer science from National Taiwan Ocean University, Keelung, Taiwan, in 2022. He is currently pursuing the M.S. degree in cybersecurity intelligence with National Cheng Kung University, Tainan, Taiwan. His research interests include lattice-based cryptography, group key exchange protocols, and post-quantum security.