**SCIS 2020** 2020 Symposium on Cryptography and Information Security
Kochi, Japan, Jan. 28 – 31, 2020
The Institute of Electronics, Information and Communication Engineers

# Implementation of Tree-based Dynamic Group Key Exchange with NewHope

Rakyong Choi [*] Dongyeon Hong [†] Kwangjo Kim [*] [†]

**Abstract:** In this paper, we give a generic approach for a group key exchange (GKE) in dynamic setting, with a tree structure. Our construction is always feasible when the group can find the network topology with tree structure efficiently. In our construction, all group members are modelled as the node of the tree. From leaf node of the tree, a child node and his/her parent node run a two-party key exchange to make their common secret key until it reaches the root node. After that, root node can deliver the group secret key to all members as parent node sends the encrypted key to his/her child node.

Then, by adopting NewHope protocol as a building block, we have implemented the first lattice-based GKE, to the best of our knowledge in the open literature. The communication complexity for our GKE protocol is $O(N)$ where $N$ is the number of participants.

**Keywords:** Lattice, RLWE, Dynamic Key Exchange, Tree-based Group Key Exchange

## 1 Introduction

Communication on an insecure network must prevent any attacks that reads transmitted messages. When two parties are communicating, a key exchange (KE) generates a common secret key for secure transmission. For secure group communiation, we require a secure transmission between multiple parties instead of two parties. A group key exchange (GKE) is a well-known cryptographic primitive that establishes a common secret key in which a shared secret key is derived from group members as a function of the information from them.

Moreover, we face the quantum computer era in the near future. Post-quantum cryptographic primitive, which is resistant to quantum computing attack, becomes one of main research areas in cryptography. Indeed, in the quantum computer era, the security of most public key cryptosystems based on number-theoretic hard problems like integer factorization problem, discrete logarithm problem, and elliptic curve discrete logarithm problem becomes vulnerable by Shor's algorithm [1].

It is clear that the effort to develop post-quantum technologies is intensifying. National Institute of Standards and Technology (NIST) requested to submit post-quantum cryptographic algorithms like encryption, KE/ key encapsulation scheme and signature scheme for standardization. While a lot of public implementation is given in public including submissions to NIST PQC competition, we don't know any implementation of lattice-based GKE in the open literature. We suggest a simple construction of GKE for a reference implementation.

In this paper, we give a generic approach for a simple and efficient dynamic GKE protocol without trusted authority and give an instantiation from the well-known lattice-based key exchange, NewHope, with an implementation result.

### 1.1 Outline of the Paper

The rest of this paper is organized as follows. We define basic terms for our scheme in Section 2 and review the related work on GKE and lattice-based KE in Section 3. Then, the design of our dynamic tree-based group key exchange protocol and the security analysis in Sections 4 and 5, respectively. We instantiate our scheme by NewHope, with an implementation result in Section 6. Finally, we give a conclusion and future work in Section 7.

## 2 Preliminaries

### 2.1 Notations

A graph, denoted by $\tilde{G} = (\tilde{V}, \tilde{E})$, is a pair of a set of nodes $\tilde{V} = \{\tilde{v}_0, \tilde{v}_1, \cdots, \tilde{v}_{N-1}\}$ and a set of edges $\tilde{V}$ connecting two nodes in $\tilde{V}$. An edge connecting two nodes $\tilde{u}$ and $\tilde{v}$ is denoted as $e_{\tilde{u},\tilde{v}} = (\tilde{u}, \tilde{v})$.

For a set $A$ and an error distribution $\chi$ sampled from $A$, $x \overset{\$}{\leftarrow} A$ denotes a uniformly random sampling where $x \in A$.

We denote a vector and a matrix as small bold letters (*e.g.*, $\mathbf{x}$, $\mathbf{y}$) and capital bold letters (*e.g.*, $\mathbf{A}$, $\mathbf{B}$), respectively. Each participant in a key exchange protocol is denoted as capital letters with a subscript (*e.g.*,

[*] School of Computing, KAIST. 291, Daehak-ro, Yuseong-gu, Daejeon, South Korea 34141. {thepride, kkj}@kaist.ac.kr
[†] Graduate School of Information Security, KAIST. 291, Daehak-ro, Yuseong-gu, Daejeon, South Korea 34141. {decenthong93, kkj}@kaist.ac.kr

$P_i$, $P_j$). We denote $\mathbb{R}$ and $\log(x)$ for a set of real numbers and $\log_2(x)$, respectively.

For a ring $\mathcal{R}$ and any integer $q \geq 2$, $\mathcal{R}_q$ denotes the quotient ring as $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$. A function $\mu$ is negligible if and only if for all $c \in \mathbb{N}$, there exists $n_c \in \mathbb{N}$ such that $\mu(n) < n^{-c}$ for all $n \geq n_c$.

### 2.2 Basic Operations from Graph Theory

To design a dynamic extension of our tree-based group key exchange framework, we use the following well-known elementary graph operations; node addition/ deletion, edge addition/deletion and edge contraction.

Node addition operation adds a new node $\tilde{w}$ to the original graph $\tilde{G} = (\tilde{V}, \tilde{E})$ so that the new graph $\tilde{G}'$ $= (\tilde{V} \cup \tilde{w}, \tilde{E})$. Edge addition operation adds an edge $\tilde{v}_1, \tilde{v}_2 \in \tilde{V}$ to the original graph $\tilde{G} = (\tilde{V}, \tilde{E})$ so that the new graph $\tilde{G}' = (\tilde{V}, \tilde{E} \cup e_{\tilde{v}_1, \tilde{v}_2})$. Similarly, we can define edge deletion and node deletion.

For edge contraction, if we set a graph $\tilde{G} = \left(\tilde{V}, \tilde{E}\right)$ which contains an edge $e = (\tilde{u}, \tilde{v})$ with $\tilde{u} \neq \tilde{v}$. If $f$ is a function which maps every node in $V \setminus \{\tilde{u}, \tilde{v}\}$ to itself, and maps $\tilde{u}$ and $\tilde{v}$ to a new single node $\tilde{w}$. Figure 1 shows an example of edge contraction operation.

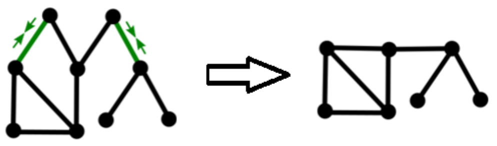


Figure 1: Edge contraction exapmple

The edge contraction of $e_{\tilde{u},\tilde{v}}$ makes a new graph $G' = \left(\tilde{V}', \tilde{E}'\right)$, where $\tilde{V}' = \left(\tilde{V} \setminus \{\tilde{u}, \tilde{v}\}\right) \cup w$ and $\tilde{E}' = \{\tilde{E} \setminus \tilde{E}_{\tilde{u},\tilde{v}}\} \cup \tilde{E}_{\tilde{w}}$. $\tilde{E}_{\tilde{u},\tilde{v}}$ is the set of edges that contains either $\tilde{u}$ or $\tilde{v}$ and $\tilde{E}_{\tilde{w}}$ is the set of edges $(\tilde{w}, \tilde{x}) \in E$ if and only if $\tilde{w} \neq \tilde{x}$ and either $(\tilde{u}, \tilde{x})$ or $(\tilde{v}, \tilde{x})$ is an edge in the original graph.

For two graphs $\tilde{G} = \left(\tilde{V}_G, \tilde{E}_G\right)$ and $\tilde{H} = \left(\tilde{V}_H, \tilde{E}_H\right)$, we say that a graph $\tilde{H}$ is called a minor of the graph $\tilde{G}$ if $\tilde{H}$ can be formed from $\tilde{G}$ by node deletion, edge deletion and edge contraction.

We define a path between two nodes as a sequence of edges $\tilde{P} = \{e_{\tilde{v}_0,\tilde{v}_1}, e_{\tilde{v}_1,\tilde{v}_2}, \cdots, e_{\tilde{v}_{i-1},\tilde{v}_i}\}$ in a graph where $\tilde{v}_0$ is the origin node and $\tilde{v}_i$ is the destination node.

A cycle is a path whose origin and destination nodes are the same and a tree is a graph without a cycle. A graph is connected that there exists a path between all two nodes in a graph.

A spanning tree $\tilde{T}_{\tilde{G}}$ of the given graph $\tilde{G}$ with $N$ nodes is a connected tree which includes all vertices of $\tilde{G}$ and the number of edges are equal to $N - 1$.

### 2.3 Ring Learning with Errors

A lattice $\Lambda$ can defined as a discrete subgroup with its basis $\mathcal{B}$. A basis $\mathcal{B}$ of $\Lambda$ is a set of linearly independent elements $\mathcal{B} = \{\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_m\}$ which generates $\Lambda$ and $\mathbf{B} = (\mathbf{b}_1|\mathbf{b}_2|\cdots|\mathbf{b}_m)$ is called as a basis matrix of $\Lambda$. Learning with Errors (LWE) problem is introduced by Regev [2] in 2005 and Ring Learning with Errors (RLWE) problem is introduced by Lyubashevsky *et al.* [3] in 2010. Both LWE and RLWE problems state that it is hard to find a secret value $s$ from $m$ independent samples $(a_i, a_i s + e_i)$ in $\mathcal{R}_q \times \mathcal{R}_q$ where $\mathcal{R}_q$ is a ring.

Decisional version of RLWE problem is commonly used as a cryptographic primitive like KE, as a building block. This problem states that it is hard to distinguish whether a sample $(a_i, b_i) \in \mathcal{R}_q \times \mathcal{R}_q$ is from RLWE distribution of the form $(a_i, a_i s + e_i) \in \mathcal{R}_q \times \mathcal{R}_q$ where $a \xleftarrow{\$} \mathcal{R}_q$, a secret key $s \leftarrow \chi_s$ and an error $e \leftarrow \chi_e$ or it is sampled from uniform distribution of $\mathcal{R}_q \times \mathcal{R}_q$.

Given a polynomial ring $\mathcal{R}$, a positive integer modulus $q$, a noise distribution $\chi$ of $\mathcal{R}_q$ and $l$ samples from either uniform or RLWE distribution, $\mathsf{Adv}^{RLWE}_{n,q,\chi_s,\chi_e,l}(\mathcal{A})$ is the advantage of algorithm $\mathcal{A}$ in distinguishing RLWE distribution and uniform distribution of $\mathcal{R}_q \times \mathcal{R}_q$ and $\mathsf{Adv}^{RLWE}_{n,q,\chi_s,\chi_e,l}(t)$ is the maximum advantage of any algorithm running in time $t$. If $\chi = \chi_s = \chi_e$, we simply write $\mathsf{Adv}^{RLWE}_{n,q,\chi,l}$.

**Protocol 1: NewHope**

| Alice | | Bob |
|---|---|---|
| $\mathsf{seed} \xleftarrow{\$} \{0,1\}^{256}$ | | |
| $\mathbf{a} \leftarrow \mathsf{Parse}(\mathsf{SHAKE\text{-}128}(\mathsf{seed}))$ | | |
| $\mathbf{s}, \mathbf{e} \xleftarrow{\$} \chi$ | | $\mathbf{s}', \mathbf{e}', \mathbf{e}'' \xleftarrow{\$} \chi$ |
| | $\xrightarrow{(b,\mathsf{seed})}$ | $\mathbf{a} \leftarrow \mathsf{Parse}(\mathsf{SHAKE\text{-}128}(\mathsf{seed}))$ |
| | | $\mathbf{u} \leftarrow \mathbf{a}\mathbf{s}' + \mathbf{e}'$ |
| | | $\mathbf{v} \leftarrow \mathbf{b}\mathbf{s}' + \mathbf{e}''$ |
| $\mathbf{v}' \leftarrow \mathbf{u}\mathbf{s}$ | $\xleftarrow{(\mathbf{u},\mathbf{r})}$ | $\mathbf{r} \xleftarrow{\$} \mathsf{HelpRec}(\mathbf{v})$ |
| $\nu \leftarrow \mathsf{Rec}(\mathbf{v}', \mathbf{r})$ | | $\nu \leftarrow \mathsf{Rec}(\mathbf{v}, \mathbf{r})$ |
| $\mu \leftarrow \mathsf{SHA3\text{-}256}(\nu)$ | | $\mu \leftarrow \mathsf{SHA3\text{-}256}(\nu)$ |

### 2.4 NewHope Protocol

Alkim *et al.* [4] proposed RLWE-based KE called NewHope in 2016. Protocol 1 describes the procedure of NewHope. To operate NewHope, we define HelpRec() and Rec() functions.

For a lattice $\hat{D}_4$ generated by a basis $\mathbf{B} = (\mathbf{u}_0, \mathbf{u}_1, \mathbf{u}_2, \mathbf{g})$ where $\mathbf{u}_i$'s are the canonical basis vectors of $\mathbb{Z}^4$ and $\mathbf{g} = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$, $\mathsf{CVP}_{\hat{D}_4}(\mathbf{x} \in \mathbb{R}^4)$ outputs the integer vector $\mathbf{z}$ from a given vector $\mathbf{x}$ such that $\mathbf{x} - \mathbf{B}\mathbf{z} \in \mathcal{V}$ where $\mathcal{V}$ is a Voronoi cell of a lattice, *i.e.*, $x \in \mathcal{V}$ is close to a zero point rather than all other lattice points.

$\mathsf{HelpRec}(\mathbf{x}; b)$ outputs an integer vector from $\frac{2^r}{q}(\mathbf{x} + b\mathbf{g})$ where $\mathsf{HelpRec}(\mathbf{x}; b) = \mathsf{CVP}_{\hat{D}_4}\left(\frac{2^r}{q}(\mathbf{x}+b\mathbf{g})\right) \mod 2^r$ and $b \in \{0, 1\}$ is a uniformly chosen random bit.

$\mathsf{Decode}(\mathbf{x} \in \mathbb{R}^4/\mathbb{Z}^4)$ outputs a bit $k$ such that $k\mathbf{g}$ is a closest vector to $\mathbf{x} + \mathbb{Z}^4$, *i.e.*, $\mathbf{x} - k\mathbf{g} \in \mathcal{V} + \mathbb{Z}^4$.

$\mathsf{Rec}(\mathbf{x}, \mathbf{r})$ is a 1-bit-out-of-4-dimensions reconciliation mechanism where

$$\mathsf{Rec}(\mathbf{x}, \mathbf{r}) := \mathsf{Decode}\Big(\frac{1}{q}\mathbf{x} - \frac{q}{2^r}\mathbf{B}\mathbf{r}\Big).$$

In our experiment, we set parameters $(n, q)$ of NewHope as $(512, 12289)$ and $(1024, 12289)$ with security level 101 and 233, respectively. They use centered binomial distribution in error sampling $\chi$.

## 3 Related Work

### 3.1 Lattice-based Key Exchange

Ding *et al.* [5] suggested the first lattice-based KE in 2012 by extending Diffie-Hellman KE [6] to RLWE setting. Since both LWE and RLWE problems require the error term, we cannot change Diffie-Hellman KE into RLWE setting directly. To control errors between two approximately agreed ring elements in $\mathcal{R}_q$, they suggested the concept of a signal function and a robust extractor, later referred as a key reconciliation mechanism. In a key reconciliation mechanism, one party has output of $\mathsf{rec}$ and a key $k \in \{0, 1\}^\lambda$ for a security parameter $\lambda$ and the other party gets a key value $k' \in \{0, 1\}^\lambda$ from $\mathsf{rec}$ and a ring element $b' \in \mathcal{R}$.

Following this research, numerous publications [4, 7–19] have been focused on constructing KEs based on lattice. Peikert [7] gave an efficient and practical lattice-based KE which is suitable as "drop-in" replacement for current Internet standards. Bos *et al.* [8] designed the more efficient protocol to be implemented in the TLS protocol and NewHope [4] improved the performance with higher security level.

Frodo protocol [9] was suggested to remove the possible risk with more structure in the hardness problem. The former protocols are designed in the ring structure using RLWE problem but Frodo was designed to rely its security on LWE problem without any ring structure.

Beyond these, there are many publications on lattice-based key exchange and authenticated key exchange protocols including Lizard, Round5, HILA5, RLWE-PAK/PPK and RLWE-SRP [11–13, 15–17, 20].

There exist a small number of publications for lattice-based GKEs. Ding *et al.* [5] suggested the basic idea of GKE protocol, without rigorous proof, based on their first lattice-based key exchange protocol in 2012. Then, Xu *et al.* [21] proposed the first lattice-based 3-party password authenticated key exchange protocol extending Ding *et al.*'s RLWE-PAK protocol [16].

Yang *et al.* [18] suggested the first provably-secure GKE protocol based on the hardness of both LWE and RLWE problems and the security of cryptographic primitive called secure sketch [22] in the random oracle model. But for secure sketch, trusted authority is necessary and this protocol may have the single point of failure issue.

Apon *et al.* [19] proposed the first provably-secure (authenticated) GKE protocol based on the hardness of RLWE problem and Katz-Yung compiler, without trusted authority. They provide a constant-round procedure regardless of the number of participants in a protocol. But to the best of our knowledge, none of previous lattice-based GKE protocols do not suggest parameter sets or an implementation result for their protocol.

### 3.2 Dynamic Group Key Exchange

In 1996, Steiner *et al.* [23] extended two-party Diffie-Hellman key exchange into GKE protcool with group communications. This protocol has no a priori ordering of group members and no synchronization as well. Then, they proposed a protocol that considers the problem of key agreement in a group setting with highly-dynamic group member population [24]. It supports dynamic group operations like adding and deleting group members. They consider two types of group key exchange protocols as centralized and contributory ones.

Kim *et al.* [25, 26] proposed Tree-based Group Diffie-Hellman (TGDH) key management solution by blending binary key trees with Diffie-Hellman key exchange. There are five types of membership changes; join, leave, merge, partition and key refresh. This protocol is dynamic and guarantees group key secrecy, forward and backward secrecy, fault-tolerance, and key independence under passive adversary.

For the security model of dynamic setting, Bresson *et al.* [27, 28] suggested two formal security models for authenticated GKE depending on the power of corruption and the presence of mutual authentication security which ensures that only legitimate participants can compute identical session group secret key.

Compared to the weak corruption model, the strong corruption model enables the adversary $\mathcal{A}$ to reveal the long-term key as well as the short-term ephemeral secrets of the protocol instance. Moreover, the security notion of forward secrecy is also defined in this security model.

## 4 Tree-based GKE Protocol

### 4.1 Basic Construction

In the following construction based on RLWE, we assume that all parties are trustful so the protocol doesn't get influenced by which party is chosen as the root node of the tree. Also, no party reveals the other's ephemeral key.

Network topology can be interpreted as a graph where the connection becomes an edge and each party becomes a node of the graph. From network topology of a given group $G$, we can find a tree structure efficiently if the graph is the connected graph. For the sake of simplicity, we assume that the balanced binary tree is chosen from the network topology and call it as a keygen tree $\tilde{T}$ of our protocol.

Finding the keygen tree $\tilde{T}$ is almost the same as finding the spanning tree $\tilde{T}_{\tilde{G}}$ of the given graph $\tilde{G}$. This can be done efficiently by using path-finding algorithms in graph theory, like well-known Dijkstra's algorithm.

If there are $N$ parties who participate in the communication, we set a keygen tree $\tilde{T}$ with a depth $d = \lceil \log N \rceil$ and define a level of a party $P_{\tilde{u}}$ as $l_{\tilde{u}} = d - l_{\mathsf{root},\tilde{u}}$ where $l_{\mathsf{root},\tilde{u}}$ is the length of a path from the node $\tilde{u}$ to the root node.

We assume that the number of parties are at least three so that the root node always has two nodes as a child. Then, we construct Tree-based Group Key Exchange (TGKE) in static version as follows:

**Step S1. Setup.**
Find a keygen tree $\tilde{T} = (\tilde{V}, \tilde{E}_T)$ with the depth $d = \lceil \log N \rceil$ from the network topology.

**Step S2. Key Construction.**

1. Set the leaf nodes as level 0 party, their parent nodes as level 1 party, till the root node as level $d$ party.
2. Between a parent node $\tilde{v}_p$ and its child node $\tilde{v}_c$, we run a two-party KE $\mathcal{TKE}$ to find the two-party common secret key $epk_{p,c}$ between two parties $P_p$ and $P_c$ as an ephemeral key.
3. We run $\mathcal{TKE}$ between level 0 parry and level 1 node. Then, each level 1 party does XOR operation to get an initial value for the next level. *e.g.*, in Figure 2(a), since party $P_2$ is the parent node of $P_4$ and $P_5$, $P_2$ has two ephemeral keys $epk_{2,4}$ and $epk_{2,5}$ and compute the XORed values $epk_2 = epk_{2,4} \oplus epk_{2,5}$.
4. Similarly, from level 1 party to level $d$ party, we run $\mathcal{TKE}$.
5. Once root node gets the ephemeral keys with his/her child node, it computes the common group secret key $sk_{\mathsf{root}}$ by XOR operation of two-party common secret key.

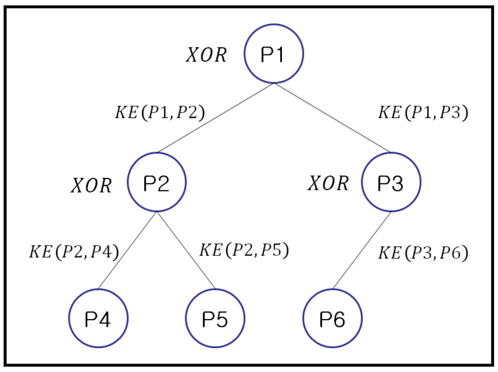
**Step S3. Key Sharing.**

1. The root node sends the encrypted common secret key $k_c$ to its child nodes $\tilde{v}_c$ by computing $k_c = sk_{\mathsf{root}} \oplus epk_{p,c}$
2. All parties get the same value $sk_g$ after $d$ rounds of sending encrypted common secret key.

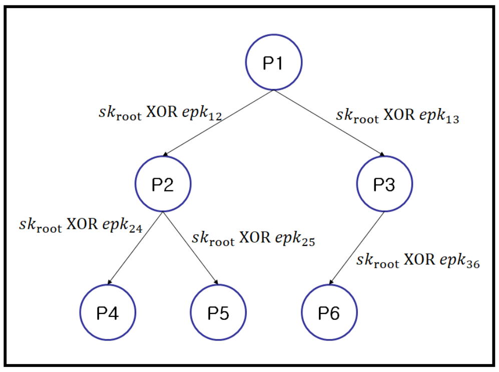
## 4.2 Dynamic TGKE

To extend TGKE into dynamic setting by describing the procedure when a new party is joining or some party is leaving the group communication.

Join describes the process when a party is joining the group communication. Figure 3 shows how a new party is joined to the network.



(a) common key construction example



P1
$sk_{\mathrm{root}}$ XOR $epk_{12}$
$sk_{\mathrm{root}}$ XOR $epk_{13}$
P2
P3
$sk_{\mathrm{root}}$ XOR $epk_{24}$
$sk_{\mathrm{root}}$ XOR $epk_{25}$
$sk_{\mathrm{root}}$ XOR $epk_{36}$
P4
P5
P6

(b) common key distribution example

Figure 2: TGKE protocol in static setting

**Step D1. Join.**

1. Add a node $\tilde{v}_N$ to the original keygen tree $\tilde{T} = \{\tilde{V}, \tilde{E}_T\}$. Then, add an edge between $\tilde{v}_N$ and some node $\tilde{v}_i \in \tilde{V}$ with level 0 or 1, which becomes the balanced binary tree after edge addition. We get a new keygen tree $\tilde{T}' = \{\tilde{V} \bigcup \{\tilde{v}_N\}, \tilde{E}_T \bigcup \{(\tilde{v}_i, \tilde{v}_N)\}\}$.
2. Between parent node $\tilde{v}_i$ and child node $\tilde{v}_N$, we run two-party key exchange protocol $\mathcal{TKE}$ to find the ephemeral key $epk_{i,N}$ between two parties $\tilde{v}_i$ and $\tilde{v}_N$.
3. $\tilde{v}_i$ sends the encrypted common secret key $k_c = sk_{\mathsf{root}} \oplus epk_{i,N}$ to $\tilde{v}_N$ and $\tilde{v}_N$ gets the

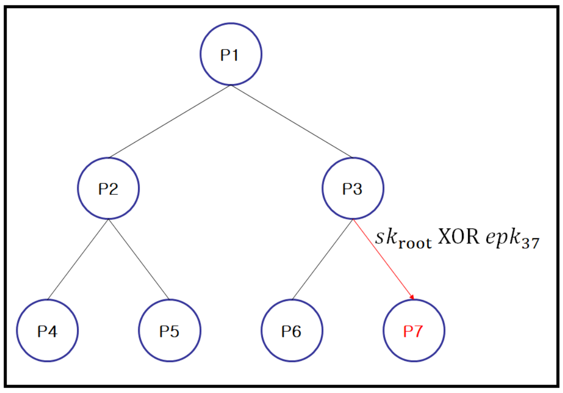
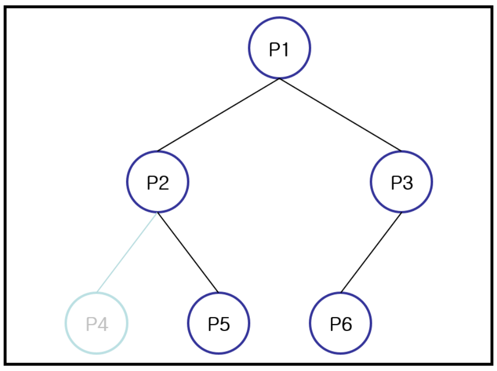
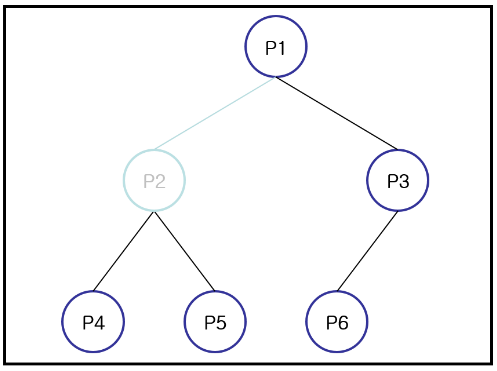


Figure 3: Join example



(a) membership revocation (leaf case)



(b) membership revocation (non-leaf case)

Figure 4: Remove examples for Leave and TreeRefresh

common secret key $sk_{\mathsf{root}}$ by XOR operation.

Leave and TreeRefresh describe the member revocation mechanism since there are two examples when the revoked member is the leaf node (deleting the node $P_4$ in Figure 4(a)) or non-leaf node (deleting the node $P_2$ in Figure 4(b)).

**Step D2-1. Leave** (leaf case).

1. Delete a node $\tilde{v}_i$ from the original keygen tree $\tilde{T} = \{\tilde{V}, \tilde{E}_T\}$. Then, we get a new keygen tree $\tilde{T}' = \{\tilde{V} \setminus \{\tilde{v}_i\}, \tilde{E}_T \setminus \{1 \leq j \leq g | (\tilde{v}_i, \tilde{v}_j)\}\}$.
2. From this new tree, run key construction phase and key sharing phase to get a new common secret key $sk'$.

**Step D2-2. TreeRefresh** (non-leaf case).

1. Delete a node $\tilde{v}_i$ from the original keygen tree $\tilde{T} = \{\tilde{V}, \tilde{E}_T\}$. Then, run the setup phase to construct a new keygen tree.
2. From this new tree, run key construction phase and key sharing phase to get a new common secret key $sk'$.

Note that if we have the connected network topology, we can easily generate the tree by contracting an edge between $\tilde{v}_i$ and its child node. In this case, both leave and tree contraction algorithms make the tree $\tilde{T}'$ which is a minor of the tree $\tilde{T}$.

## 5 Security Analysis

In this section, we give a security analysis of static TGKE. Any probabilistic polynomial-time (PPT) adversary should not distinguish a real common group secret key to a random one even if he/she gets the transcripts of the protocol. We assume that every party is trustful and no party does insider attacks.

We derive the correctness proof and the security proof in **Theorems 1** and **2**, respectively.

**Theorem 1.** *Our protocol has the same common secret key between all parties in the group.*

*Proof. (sketch)* Since we focus on designing conceptually-simpler model for fast and efficient implementation result, we send the common secret key by XORing two-party common secret key. By repeating the same process between parent node and his/her child node from level 0 to level $d$, we always get the correct common secret key for the group. □

Before proving **Theorem 2**, we remark that the root node has the level $d$ and a leaf node has the level 0 where $d = \log N$ is a depth of a keygen tree from the network of the group.

**Theorem 2.** *If underlying two-party key exchange protocol $\mathcal{TKE}$ is secure against the passive adversary, TGKE protocol is also secure.*

*Proof. (sketch)* We prove a security by a hybrid game between the real shared secret key and the random one for each level of the tree.

Then, $\mathbf{Game}_0$ is the real game which the adversary gets the real common group secret key $sk$ and $\mathbf{Game}_d$ is the game which the adversary gets the random value $t_d$. We show that the views of $\mathbf{Game}_0$ and $\mathbf{Game}_d$ are computationally indistinguishable for any PPT adversaries.

$\mathbf{Game}_0$. This game is the real game between protocol challenger and the passive adversary $\mathcal{A}$, adversary obtains the common secret key $sk$ from our protocol described in Section 4.

$\mathbf{Game}_1$. This game is identical to $\mathbf{Game}_0$ except level 1 nodes changes his initial key as a random value instead of two-party common secret key with level 0 nodes.

Since the underlying two-party key exchange protocol is secure against passive adversary $\mathcal{A}$. Random values are indistinguishable from the key generated by two-party KE for any PPT adversaries. Thus, the adversary cannot distinguish $\mathbf{Game}_0$ and $\mathbf{Game}_1$.

**Game**$_i$ $(2 \leq i \leq d-1)$. This game is identical to **Game**$_{i-1}$ except level $i$ nodes changes his initial key as a random value $r_i$ instead of two-party common secret key with level $i-1$ nodes. Since the underlying two-party key exchange protocol is secure against passive adversary $\mathcal{A}$. Random values are indistinguishable from keys generated by two-party KE for any PPT adversaries. Thus, the adversary cannot distinguish **Game**$_{i-1}$ and **Game**$_i$.

**Game**$_d$. This game is identical to **Game**$_{d-1}$ except that encrypted values from parties with level $d-1$ to the root node are replaced by random value $t_d$ during the protocol. Then, similar to the game between **Game**$_{i-1}$ and **Game**$_i$, the views between **Game**$_{d-1}$ and **Game**$_d$ are computationally indistinguishable for any PPT adversaries.

Now, the claim follows since the real game **Game**$_0$ and the random game **Game**$_d$ are computationally indistinguishable assuming the security of underlying two-party key exchange protocol.

□

## 6 Implementation

### 6.1 TGKE with NewHope

To apply NewHope into our TGKE framework, we slightly modify the original construction. For setup phase, we use the AVL tree for balanced binary tree.

For key construction with NewHope, we modify the way to use XORed value of ephemeral keys by two parties. In our experiment, we use XORed value as a seed value of NewHope in the next level. The detail is described below:

**Step I2. Key Construction with NewHope.**

1. Set the leaf node as level 0 party, its parent node as level 1 party, till the root node is set as level $d$ party.
2. Between a parent node $\tilde{v}_p$ and its child node $\tilde{v}_c$, we run NewHope to find the two-party common secret key $epk_{p,c}$ between two parties $P_p$ and $P_c$ as an ephemeral key.
3. We run NewHope between level 0 node and level 1 node. Then, each level 1 node does XOR operation to obtain a seed from the protocol 1. *e.g.*, in Figure 2, since party $P_2$ is the parent node of $P_4$ and $P_5$, $P_2$ has two ephemeral keys $epk_{2,4}$ and $epk_{2,5}$ and compute the XORed values $epk_2 = epk_{2,4} \oplus epk_{2,5}$.
4. Similarly, from level 1 party to level $d$ party, we run NewHope between a child node and its parent node.
5. Once root node gets the ephemeral keys with his/her child node, it computes the common secret key $sk_{\mathsf{root}}$ by XOR operation of two-party common secret key. We assume that the number of parties are at least three so that the root node always has two nodes as a child.

Compared to our design, we use XORed value as a seed of NewHope in the next level. To do that, we can avoid the possible risk that the secret key is not a ring element or suitable for RLWE problem.

For Join and two removal steps Leave and TreeRefresh of TGKE, while we simply add a node to the tree and adjust the tree when a party is joining, a huge computation is required for removing a party since either we initialize all values of all nodes or reconstruct a tree.

### 6.2 Implementation Result

Our experiment was run on a computer with AMD Ryzen 2600x six-core processor, 3.8GB of RAM, running Ubuntu 18.04.3 LTS. We summarize the running time for static TGKE in Table 1 when the number of participants $N$ = 10, 30, 100 and 300, the dimension $n = 512$ or 1024 and the modulus $q = 12289$ for both dimension. We run the protocol 300 times for each condition and check the average time to avoid the biased data.

Table 1: Running time for static TGKE (unit: ms)

| Static TGKE with NewHope | | Number of Parties | | | |
|---|---|---|---|---|---|
| | | 10 | 30 | 100 | 300 |
| Dimension | 512 (101) | 0.755 | 2.404 | 8.458 | 25.550 |
| (Security Level) | 1024 (233) | 1.546 | 5.005 | 16.787 | 50.661 |

From Table 1, we check that the running time for TGKE in static setting depends on the number of participants almost linearly.

We also check the running time for Join and Remove steps of TGKE, as membership changes by adding parties to the group or removing parties from the group, in Tables 2 and 3, respectively. We set the parameters of NewHope for these two experiments as $(n, q) = (512, 12289)$.

As we expected from our design of Join, Table 2 shows the similar running time if the number of original parties are the same.

But for removal steps, since we have to redesign the tree when we remove the party out of the group, the running time increases almost linearly as the number of removal parties is increased.

### 6.3 Comparison with Other Protocols

In Table 4, we compare our construction with other lattice-based GKEs in a theoretic point of view.

Compared to other protocols, our construction has advantages that we get an implementation result and moreover, it can be extended to the dynamic setting.

Table 2: Running time for Join step (unit: ms)

| Join of TGKE with NewHope | | Number of Parties | | | |
|---|---|---|---|---|---|
| | | 10 | 30 | 100 | 300 |
| Number of Join Parties | 5 | 0.759 | 2.389 | 8.163 | 24.557 |
| | 30 | 0.781 | 2.445 | 8.551 | 26.340 |

Table 3: Running time for removal steps (unit: ms)

| Remove of TGKE with NewHope | | Number of Parties | | | |
|---|---|---|---|---|---|
| | | 10 | 30 | 100 | 300 |
| Number of Remove Parties | 5 | 3.419 | 13.037 | 47.075 | 145.198 |
| | 15 | - | 30.654 | 121.741 | 391.755 |
| | 30 | - | - | 223.297 | 747.587 |

But we have disadvantages that our security model is much weaker than the standard model suggested by Bresson *et al.* [27, 28] or other security model. Indeed, there is a trivial attack in our protocol if an adversary can get all two-party common secret keys when one of the participants has leaked his information.

## 7 Conclusion and Future Work

In this paper, we construct a simple and theoretically-efficient approach to design a GKE protocol using keygen trees and give a quantum-resistant instantiation with the well-known NewHope protocol.

Then, we compare this method with other lattice-based key exchange protocols. Our construction relies on the security of underlying two-party key exchange protocol and the communication complexity of lattice-based GKE protocol is $O(N)$ where $N$ is the number of group members.

Compared to Kim *et al.*'s Tree-based Group Diffie-Hellman key management [26], our keygen tree considers all nodes to be parties in the protocol while only leaf node is the communication party in [26]. Hence, our tree has smaller number of vertices and edges. On the other hand, our method covers single member addition and revocation while Kim *et al.*'s method provides group addition and revocation too.

Currently, our GKE model is vulnerable to insider attacks since any party can impersonate other parties by achieving their two-party common secret keys after XORing with the group key $sk_{\mathsf{root}}$. Also, our protocol is vulnerable to active adversaries who can corrupt a party.

As future work, we consider how to control this issue without increasing a lot of computation cost. Then, we check the security under active adversaries who can modify the scheme.

Table 4: Comparison with other lattice-based GKEs

| Method | DXL12 [5] | YMZ15 [18] | ADGK19 [19] | Ours |
|---|---|---|---|---|
| Communication Complexity[a] | O($N^2$) | O($N$) | O($N$) | O($N$) |
| Trusted Authority | X | O | X | △[b] |
| Public Implementation | No | No | No | Yes |
| Dynamic Setting | No | No | No | Yes |

[a] $N$ is the number of participants for GKE.
[b] There is no trusted authority but all parties must be very honest to prevent any attacks.

## Acknowledgement

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. 2017-0-00555, Towards Provable-secure Multi-party Authenticated Key Exchange Protocol based on Lattices in a Quantum World).

## References

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[2] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," in *Annual ACM Symposium on Theory of Computing*, pp. 84–93, ACM, 2005.

[3] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 1–23, Springer, 2010.

[4] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "Post-quantum key exchange-a new hope," in *USENIX Security Symposium*, pp. 327–343, 2016.

[5] J. Ding, X. Xie, and X. Lin, "A simple provably secure key exchange scheme based on the learning with errors problem," *IACR Cryptology ePrint Archive 2012/688*, 2012.

[6] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[7] C. Peikert, "Lattice cryptography for the internet," in *International Workshop on Post-Quantum Cryptography*, pp. 197–219, Springer, 2014.

[8] J. W. Bos, C. Costello, M. Naehrig, and D. Stebila, "Post-quantum key exchange for the TLS protocol from the ring learning with errors problem," in *IEEE Symposium on Security and Privacy*, pp. 553–570, IEEE, 2015.

[9] J. Bos, C. Costello, L. Ducas, I. Mironov, M. Naehrig, V. Nikolaenko, A. Raghunathan, and D. Stebila, "Frodo: Take off the ring! practical, quantum-secure key exchange from LWE," in *ACM SIGSAC Conference on Computer and Communications Security*, pp. 1006–1018, ACM, 2016.

[10] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS–Kyber: a CCA-secure module-lattice-based KEM," in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 353–367, IEEE, 2018.

[11] H. Baan, S. Bhattacharya, S. R. Fluhrer, O. Garcia-Morchon, T. Laarhoven, R. Rietman, M.-J. O. Saarinen, L. Tolhuizen, and Z. Zhang, "Round5: Compact and fast post-quantum public-key encryption.," pp. 83–102, 2019.

[12] J. H. Cheon, D. Kim, J. Lee, and Y. Song, "Lizard: Cut off the tail! a practical post-quantum public-key encryption from LWE and LWR," in *International Conference on Security and Cryptography for Networks*, pp. 160–177, Springer, 2018.

[13] M.-J. O. Saarinen, "HILA5: On reliability, reconciliation, and error correction for Ring-LWE encryption," in *International Conference on Selected Areas in Cryptography*, pp. 192–212, Springer, 2017.

[14] J. Zhang, Z. Zhang, J. Ding, M. Snook, and Ö. Dagdelen, "Authenticated key exchange from ideal lattices," in *Advances in Cryptology–EUROCRYPT 2015*, pp. 719–751, 2015.

[15] J. Zhang and Y. Yu, "Two-round PAKE from approximate SPH and instantiations from lattices," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 37–67, Springer, 2017.

[16] J. Ding, S. Alsayigh, J. Lancrenon, R. Saraswathy, and M. Snook, "Provably secure password authenticated key exchange based on RLWE for the post-quantum world," in *Cryptographers' Track at the RSA Conference*, pp. 183–204, Springer, 2017.

[17] X. Gao, J. Ding, J. Liu, and L. Li, "Post-quantum secure remote password protocol from rlwe problem," in *International Conference on Information Security and Cryptology*, pp. 99–116, Springer, 2017.

[18] X. Yang, W. Ma, and C. Zhang, "Group authenticated key exchange schemes via learning with errors," *Security and Communication Networks*, vol. 8, no. 17, pp. 3142–3156, 2015.

[19] D. Apon, D. Dachman-Soled, H. Gong, and J. Katz, "Constant-round group key exchange from the ring-LWE assumption.," in *International Conference on Post-Quantum Cryptography*, pp. 189–205, Springer, 2019.

[20] J. Katz and V. Vaikuntanathan, "Smooth projective hashing and password-based authenticated key exchange from lattices," in *Advances in Cryptology–ASIACRYPT 2009*, vol. 5912, pp. 636–652, Springer, 2009.

[21] D. Xu, D. He, K.-K. R. Choo, and J. Chen, "Provably secure three-party password authenticated key exchange protocol based on ring learning with error," *IACR Cryptology ePrint Archive 2017/360*, 2017.

[22] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in *International conference on the theory and applications of cryptographic techniques*, pp. 523–540, Springer, 2004.

[23] M. Steiner, G. Tsudik, and M. Waidner, "Diffie-Hellman key distribution extended to group communication," in *Proceedings of the 3rd ACM conference on Computer and communications security*, pp. 31–37, ACM, 1996.

[24] M. Steiner, G. Tsudik, and M. Waidner, "CLIQUES: A new approach to group key agreement," in *Distributed Computing Systems, 1998. Proceedings. 18th International Conference on*, pp. 380–387, IEEE, 1998.

[25] Y. Kim, A. Perrig, and G. Tsudik, "Simple and fault-tolerant key agreement for dynamic collaborative groups," in *Proceedings of the 7th ACM conference on Computer and communications security*, pp. 235–244, ACM, 2000.

[26] Y. Kim, A. Perrig, and G. Tsudik, "Tree-based group key agreement," *ACM Transactions on Information and System Security (TISSEC)*, vol. 7, no. 1, pp. 60–96, 2004.

[27] E. Bresson, O. Chevassut, and D. Pointcheval, "Provably authenticated group Diffie-Hellman key exchange – the dynamic case," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 290–309, Springer, 2001.

[28] E. Bresson, O. Chevassut, and D. Pointcheval, "Dynamic group Diffie-Hellman key exchange under standard assumptions," in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 321–336, Springer, 2002.