SCIS 2019 2019 Symposium on
Cryptography and Information Security
Shiga, Japan, Jan. 22 - 25, 2019
The Institute of Electronics,
Information and Communication Engineers

DHL: Dynamic Key Exchange from Homomorphic Encryption based on Lattice

Rakyong Choi * Kwangjo Kim *

Abstract: Group key exchange protocol produces a common secret key between n parties for secure (group) communication in insecure channel over the internet. In ProvSec Workshop 2018, Choi and Kim designed a novel multi-party key exchange protocol with homomorphic encryption in static setting. In this paper, we discuss the limitation of their scheme and extend this protocol to dynamic setting for managing group membership issue in terms of graph theory. We use the similar graph structure from Kim *et al.*'s key management solution called Tree-based Group Diffie-Hellman (TGDH,) for necessary algorithms providing dynamic key exchange from homomorphic encryption based on lattice problems.

Keywords: Key Exchange, Homomorphic Encryption, Dynamic Key Exchange, Tree-based Group Key Exchange

1 Introduction

1.1 Background and Motivation

Communication between two parties over an insecure channel requires a key exchange protocol to prevent any attacks to read transmitted messages, including unauthorized access, accidental disclosure of the information, *etc.* For secure transmission, a message should be encrypted by an encryption key and sent from one party to another. A challenge for this is how to share the key between two parties securely. To solve this, we must use key exchange protocols which identifies each party to another, create and distribute the key among them securely.

On the other hand, homomorphic encryption (HE) supports some computation on encrypted data without decryption key. After Gentry's breakthrough paper [1] in 2009, there are a number of research on HE schemes based on lattices with Learning With Errors (LWE) and Ring Learning With Errors (Ring-LWE) problems [2–5] and schemes over integers with approximate Greatest Common Divisor (GCD) problem [6, 7]. HE is applicable to various areas using outsourcing computation such as machine learning methods for encrypted data [8, 9] or two-party key exchange protocol [10].

In this paper, we suggest a generic approach to construct a dynamic key exchange protocol from rich cryptographic ingredients and give a post-quantum instantiation.

1.2 Outline of the Paper

The rest of this paper is structured as follows: We give the history of group key exchange protocol in Chapter 2. We give the definition of homomorphic encryption and basic operations from graph theory in Chapter 3. In Chapters 4 and 5, we construct a multi-party key exchange protocol in static and dynamic settings, respectively. Then, Chapter 6 gives an instantiation from HE based on lattice. In Chapter 7, we compare our instantiation with previous quantum-resistant multi-party key exchange protocols like Ding *et al.*'s protocol [11], Boneh *et al.*'s protocol [12], and Choi and Kim's protocol [13]. Finally, we give a concluding remark in Chapter 8.

2 Previous Work

2.1 Lattice-based Key Exchange

In 2012, Ding *et al.* [11] proposed the first lattice-based key exchange protocol and Peikert [14] gave its tweak which is efficient and more suitable as "drop-in" replacement for current Internet standards. Bos *et al.* [15] implemented the lattice-based key exchange protocol in TLS protocol and NewHope protocol [16] improved the performance of Bos *et al.'s protocol* [15] with higher security level. Frodo protocol [17] was suggested to remove the ring structure to reduce the possible risk in the hardness of the problem with more structure. The former practical protocols were designed in the ring structure but Frodo was designed to rely its security on the hardness of LWE problem without ring structure.

Zhang *et al.* [18] designed the first authenticated key exchange based on lattice similar to the well-known HMQV protocol [19]. Kyber protocol [20] is yet another authenticated key exchange protocol recently proposed by Bos *et al.* This protocol is based on a variant of LWE problem called Module-LWE to enhance the performance and they prove the security with the

* School of Computing, KAIST. 291, Daehak-ro, Yuseong-gu, Daejeon, South Korea 34141. {thepride, kkj}@kaist.ac.kr

Quantum-accessible Random Oracle Model (QaROM) instead of classical Random Oracle Model (ROM).

Beyond these, there are many submissions on lattice-based key exchange for NIST post quantum cryptography standards.

2.2 Group Key Exchange

A group key exchange (GKE) protocol is a key exchange protocol in which a shared secret is derived from n parties as a function of the information contributed by each party. In 1996, Steiner *et al.* [21] extended two-party Diffie-Hellman key exchange to group communications with three natural multi-party cases. The 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 [22]. It supports dynamic group operations like adding and deleting group members. They consider two types of GKE protocols as centralized one and contributory one.

Kim *et al.* [23] proposed Tree-based Group Diffie-Hellman (TGDH) key management solution by blending binary key trees with Diffie-Hellman key exchange [24]. There are five initial operations: join, leave, merge, partition, and key refresh. This protocol is dynamic and guarantees group key secrecy, forward/backward secrecy, fault-tolerance, and key independence under passive adversary.

Krendelev and Kuzmin [10] recently proposed a two-party key exchange protocol based on HE and recently, Choi and Kim [13] proposed a multi-party key exchange protocol based on HE with the graph structure.

For quantum-resistant multi-party key exchange protocols, Ding *et al.* [11] constructed the lattice-based interactive multi-party GKE protocol and Boneh *et al.* [12] showed how to use a cryptographic invariant map to construct a non-interactive key exchange protocol in which n parties create a common secret key without any interaction among the n parties. Then, they constructed the multi-party non-interactive key exchange protocol with cryptographic invariant maps from isogenies. But, to the best of our knowledge, there exists no dynamic key exchange protocol with quantum-resistance.

3 Preliminaries

3.1 Notation

We denote vertices as italic small letters with tilde (*e.g.*, $\tilde{u}, \tilde{v}$) and an edge between $\tilde{u}$ and $\tilde{v}$ as $\tilde{e}_{\tilde{u},\tilde{v}} = (\tilde{u}, \tilde{v})$. The set of vertices and the set of edges are denoted as $\tilde{V}$ and $\tilde{E}$, respectively. Then, a graph $\tilde{G} = \left(\tilde{V}, \tilde{E}\right)$ is a collection of vertices and edges. We say a vertex $\tilde{u}$ is incident to an edge e if $\tilde{u} \in e$.

We denote vectors as bold small letters (*e.g.*, $\mathbf{x}$, $\mathbf{y}$) and matrices as bold capital letters (*e.g.*, $\mathbf{A}$, $\mathbf{B}$). Let $\mathbb{R}$ and $\mathbb{Z}$ express the set of real numbers and the set of integers, respectively and italic letters express real numbers (*e.g.*, a, b, c).

For any integer $q \geq 2$, $\mathbb{Z}_q$ denotes the ring of integers modulo q and $\mathbb{Z}_q^{n \times m}$ denotes the set of $n \times m$ matrices with entries in $\mathbb{Z}_q$.

Let $f(a, b)$ be a function f on a and b. We say a function $f : \mathbb{Z} \to \mathbb{R}^+$ is *negligible* when $f = O(n^{-c})$ for all $c > 0$ and denoted by $\mathrm{negl}(n)$. A function $g(m) = \lfloor m \rfloor$ is the floor function from $\mathbb{R}$ to $\mathbb{Z}$ such that $g(m)$ is the largest integer which is smaller than or equal to m.

3.2 Homomorphic Encryption

In 1978, Rivest *et al.* [25] suggested the concept of homomorphic encryption that supports some computation on encrypted data without the knowledge of the secret key. A homomorphic encryption scheme is defined as follows:

Definition 1. (homomorphic encryption) A homomorphic encryption scheme $\mathcal{HE}$ is a tuple of PPT algorithms $\mathcal{HE}$ = (HE.Gen, HE.Enc, HE.Eval, HE.Dec) with the following functionality:

HE.Gen(n, α) :
Given the security parameter n and an auxiliary input α, this algorithm outputs a key triple (pk, sk, evk), where pk is the key used for encryption, sk is the key used for decryption and evk is the key used for evaluation.

HE.Enc(pk, m) :
Given a public key pk and a message m, this algorithm outputs a ciphertext c of the message m.

HE.Eval$(evk, C, c_1, \cdots, c_n)$:
Given an evaluation key evk, a Boolean circuit C, and pairs $\{c_i\}_{i=1}^n$ where c_i is either a ciphertext or previous evaluation results, this algorithm produces an evaluation output.

HE.Dec(sk, c) :
Given a secret key sk and a ciphertext or an evaluation output c, this algorithm outputs a message m.

Depending on what kind of Boolean circuit the scheme supports, we call an encryption scheme as multiplicative (*resp.* additive) HE if it only supports multiplication (*resp.* addition).

Early work on HE was not practical but there are many cryptographic algorithm tools that support HE efficiently such as HElib, FHEW [26, 27].

3.3 Basic Operations from Graph Theory

We define a graph $\tilde{G}$ as a pair of the set of vertices $\tilde{V} = \{\tilde{v}_1, \tilde{v}_2, \cdots, \tilde{v}_n\}$ and the set of edges $\tilde{E}$ between two vertices. Among many terminologies in graph theory, we use vertex/edge addition, vertex/edge deletion, and edge contraction. Since the process of vertex/edge addition and vertex/edge deletion operations is clear,

we give a definition of edge contraction and graph minor.

Definition 2. (edge contraction) Let $\tilde{G} = \left(\tilde{V}, \tilde{E}\right)$ be a graph containing an edge $\tilde{e} = (\tilde{u}, \tilde{v})$ with $\tilde{u} \neq \tilde{v}$. Let f be a function which maps every vertex in $\tilde{V} \setminus \{\tilde{u}, \tilde{v}\}$ to itself, and maps $\tilde{u}$ and $\tilde{v}$ to a new vertex $\tilde{w}$. The edge contraction of e makes a new graph $G' = \left(\tilde{V}', \tilde{E}'\right)$, where $\tilde{V}' = \left(\tilde{V} \setminus \{\tilde{u}, \tilde{v}\}\right) \cup \tilde{w}$, $\tilde{E}' = \tilde{E} \cup \tilde{e}$, and for every $\tilde{v} \in \tilde{V}, \tilde{x}' = f(\tilde{x}) \in \tilde{V}$ is incident to an edge $\tilde{e}' \in \tilde{E}$ if and only if, the corresponding edge, $\tilde{e} \in \tilde{E}$ is incident to $\tilde{x}$ in $\tilde{G}$.

Definition 3. (graph minor) Let $\tilde{G} = \left(\tilde{V}_G, \tilde{E}_G\right)$ and $\tilde{H} = \left(\tilde{V}_H, \tilde{E}_H\right)$ be a graph. If $\tilde{H}$ can be formed from $\tilde{G}$ by deleting vertices/edges and contracting edges, we say that $\tilde{H}$ is called a minor of $\tilde{G}$.

We use these terms for membership addition and revocation in dynamic setting.

4 Our Construction: Static Setting

4.1 Protocol in the Literature: HE-KE

In Figure 1, we state a methodology to build a (non-interactive) multi-party key exchange protocol fro HE by Choi and Kim [13] (HE-KE protocol). All parties pre-share the master secret key sk from $\mathsf{HE.Gen}$ algorithm of HE scheme $\mathcal{HE}$. A circuit C can be public in this protocol.

Assuming that the server is honest but curious, HE-KE protocol runs as follows.

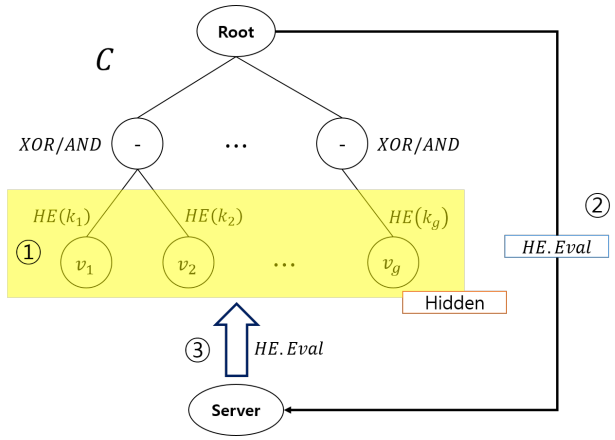

Figure 1: HE-KE protocol

Step 1. Each party pre-shares the master group key $sk \leftarrow \mathsf{HE.Gen}(n, \alpha)$ with each other.

Step 2. Make the Boolean circuit C with g leaf nodes.

Step 3. Each party makes ephemeral session key k_i and encrypts it with its public key pk_i, $c_i = \mathsf{HE.Enc}\ (pk_i, k_i)$. (① from Figure 1.)

Step 4. The server computes $c = \mathsf{HE.Eval}(evk, C, c_1, \cdots, c_g)$ with the Boolean circuit C and broadcasts c. (② and ③ from Figure 1, respectively.)

Step 5. Each party decrypts the evaluated value c and get the session group key $k = \mathsf{HE.Dec}(sk, c)$.

Lemma 1. *[13] If underlying homomorphic encryption $\mathcal{HE}$ is secure, HE-KE protocol is also secure, i.e., it satisfies session key security, known key security, and key privacy.*

4.2 System Model and Security Requirement

HE-KE protocol [13] suggests non-interactive property by pre-sharing the master group key between each party. But, in this protocol, the way each party pre-shares the master secret key is unclear. Even more, since there is no key refresh algorithm, we should assume that the former group member is trustful so that the adversary $\mathcal{A}$ doesn't get the master secret key.

To resolve this, we assume that the server is fully trustful so that the server distributes the ephemeral key as well as the evaluated value of session group key to each party for each membership event (*e.g., join, leave, merge, or partition*). Compared to HE-KE protocol [13], the former group member cannot compute the session group key in our protocol.

We assume that the adversary $\mathcal{A}$ can (i) send messages to some party, (ii) run the protocol to get the appropriate session group key, and (iii) get some information from a previous group member like the former session group keys.

To check the security of our dynamic key exchange protocol, one of the most important security requirements is key freshness. A key is called *fresh* if the generated key is guaranteed to be new to prevent an old key being reused by an adversary. To guarantee key freshness, we have to prove the following security requirements:

1. **Group Key Secrecy**
   If all group members in the protocol are not corrupted, it is computationally infeasible for a passive adversary to discover any session group key.

2. **Forward Secrecy**
   Even after a passive adversary $\mathcal{A}$ has acquired some session group keys, new session group keys must remain out of reach of the adversary and former group members.

3. **Backward Secrecy**
   Even after a passive adversary $\mathcal{A}$ has acquired some session group key, previously used session group keys must not be discovered by the adversary and new group members.

4. **Key Independence**
   A passive adversary who knows a proper subset of session group keys cannot discover any other session group keys.

4.3 Parcel-S Protocol

In Figure 2, we give a methodology to build a GKE protocol from HE called Parcel-S for static setting and Parcel-D for dynamic setting, which supports the membership events. As discussed earlier, the server delivers the ephemeral key to all group members to provide forward and backward secrecy. Then, the server broadcasts the evaluated value of ciphertexts from group members and the Boolean circuit C. Assume that we have either XOR or AND operation in C.

All group members pre-share the master secret key sk from $\mathsf{HE.Gen}$ algorithm of HE scheme $\mathcal{HE}$. A circuit C can be public in this protocol.

Under this condition, Parcel-S protocol runs as follows.

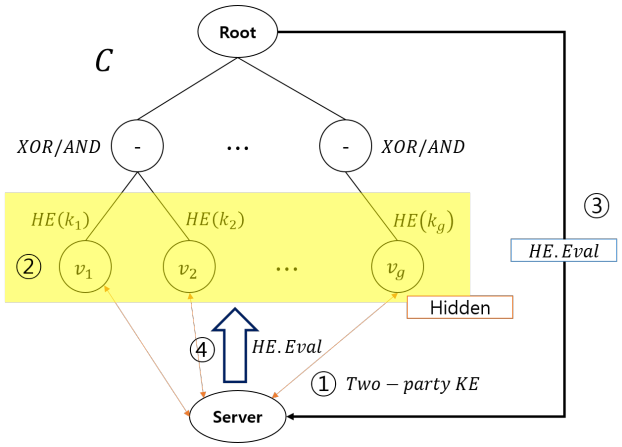

Figure 2: Parcel-S protocol

Step 1. Each party pre-shares the master group key $sk \leftarrow \mathsf{HE.Gen}(n, \alpha)$ with each other and the server runs two-party key exchange protocol $\mathcal{TKE}$ with each party to make the long-term secret key msk_i. (① from Figure 3.)

Step 2. Make the Boolean circuit C with g leaf nodes.

Step 3. Each party makes ephemeral session key k_i and encrypts it with its public key pk_i, $c_i = \mathsf{HE.Enc}\ (pk_i, k_i)$. (② from Figure 3.)

Step 4. The server computes $c = \mathsf{HE.Eval}(evk, C, c_1, \cdots, c_g)$ with the Boolean circuit C and broadcasts c. (③ and ④ from Figure 3, respectively.)

Step 5. The server sends the ephemeral key epk to each party encrypted by encryption scheme $\mathcal{SKE}$ with secret key msk_i.

Step 6. Each group member decrypts the evaluated value c and the ephemeral key epk. Then, each group member gets the proper session group key $k = \mathsf{HE.Dec}(sk, c) \oplus epk$.

4.4 Security Analysis

In this section, we show the correctness and security requirements given in the previous section.

Theorem 1. *If underlying homomorphic encryption scheme $\mathcal{HE}$ is valid, Parcel-S protocol is correct, i.e., it outputs the valid session group key for each session.*

Proof. In Step 4, the server can compute the evaluated value c if $\mathcal{HE}$ is valid and $\mathsf{HE.Eval}(evk, C, c_1, \cdots, c_g)$ is well-defined. Then, each party can get the same value from XOR operation between decryption of c and ephemeral key epk. □

Theorem 2. *If underlying homomorphic encryption scheme $\mathcal{HE}$ and two-party key exchange protocol $\mathcal{TKE}$ are secure, Parcel-S protocol is also secure, i.e., it satisfies group key secrecy, forward/backward secrecy, and key independence.*

Proof. Since $\mathcal{HE}$ is secure, each ciphertext and evaluated value are indistinguishable from random. Thus, all ciphertext c_i of the ephemeral session key k_i from party $\tilde{v}_i$ are indistinguishable from random and so does the ciphertext c of the session group key k, evaluated value of all the ciphertext $c_i's$. Hence, our construction guarantees group key secrecy.

Since the server doesn't have the information of the pre-shared secret key sk, the server cannot know the session group key k because the server doesn't know the decryption value of c. Likewise, the former(*resp.* new) group members cannot know the new(*resp.* previous) session group keys since they don't know the ephemeral key since $\mathcal{TKE}$ is secure. Thus, our protocol provides forward secrecy and backward secrecy.

Thanks to the presence of ephemeral keys, we can show the key independence as well.

□

5 Parcel-D Protocol

5.1 Membership Events

As discussed in Chapter 4, a dynamic setting needs to provide key adjustment protocols to cope with any membership changes. Parcel-D protocol includes algorithms to support the following operations:

Join$(\tilde{v}', C)$:
When a new group member $\tilde{v}'$ is added to the group to participate in the group communication, reconstruct the Boolean circuit C_{Join}.

Leave$(\tilde{v}_i, C)$:
When a group member $\tilde{v}_i$ is removed from the group communication, reconstruct the Boolean circuit C_{Leave}.

Merge$(\tilde{V}', C, C')$:
When some group $\tilde{V}'$ is merged with the current group, reconstruct the Boolean circuit C_{Merge}.

Partition$(\tilde{V}_j, C)$:
When a subset of group members $\tilde{V}_j$ are removed from the group communication, reconstruct the Boolean circuit $C_{Partition}$.

After these membership events, we run Parcel-S protocol again to get a new session group key. Note that for each membership event, the only change is the Boolean circuit. A new member can get the same master group key sk since only the existence of ephemeral keys provides forward/backward secrecy. For the remaining of this chapter, we give the detail explanation on each membership event algorithm.

5.2 Join Protocol

We assume the group has g members. The new member $\tilde{v}_{g+1}$ initiates the protocol by sending a 'join' request message to the server. If the server receives this message, the server makes the long-term secret key with $\tilde{v}_{g+1}$. Then, it determines the splitting point in the circuit. The splitting point is the shallowest rightmost node, where the joining of a new member does not increase the depth of the circuit. If the circuit is a fully balanced tree, it chooses a point with more XOR operations in the path to the root node, to minimize the complexity. In the splitting point, we put some Boolean operation like XOR or AND. A new Boolean circuit C_{Join} is constructed as follows:

1. Find the splitting point that does not increase the depth of the circuit or minimize the complexity.
2. Add two vertices to the splitting point and connect these vertices to the splitting point.
3. Put some Boolean operation to the splitting point.
4. Set the leaf nodes as the group members $\tilde{v}_1, \tilde{v}_2, \cdots, \tilde{v}_{g+1}$.

Figure 3 shows an example of Boolean circuit when a new group member is joined to the group.

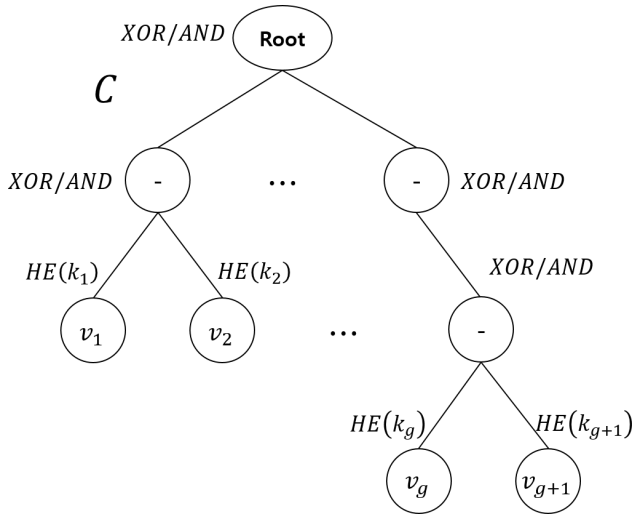

Figure 3: Join algorithm in Parcel-D protocol

5.3 Leave Protocol

Again, we start with g members and assume that member $\tilde{v}_i$ leaves the group. When $\tilde{v}_i$ leaves, the server first deletes the long-term secret key between the server and $\tilde{v}_i$.

From the original Boolean circuit C, the server finds the leaf node marked as $\tilde{v}_i$ and its parent node. Then, we construct a new Boolean circuit C_{Leave} as follows:

1. Find the leaf node marked as $\tilde{v}_i$ and its parent node $\tilde{w}$.
2. Remove $\tilde{v}_i$ node and contract an edge between $\tilde{w}$ node and $\tilde{v}_{i+1}$ node, where $\tilde{v}_{i+1}$ node has the same parent node $\tilde{w}$ with $\tilde{v}_i$ node. (Without loss of generality, we may rename the group members to satisfy this condition.)
3. Rename the leaf nodes as the group members $\tilde{v}'_1, \tilde{v}'_2, \cdots, \tilde{v}'_{g-1}$.

Figure 4 shows an example of Boolean circuit when a group member leaves the group.

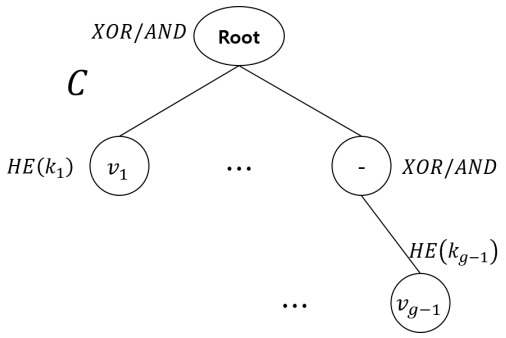

Figure 4: Leave algorithm in Parcel-D protocol

5.4 Merge Protocol

Network faults may partition a group into several subgroups. In the meantime, they communicate inside the subgroups only. After the network recovers, subgroups need to be merged into a single group. In this case, since all group members already exist in the group communication, we don't need to make a new long-term secret key between the server and each group member.

To build a new Boolean circuit, the server checks the connecting point in the circuit C. The connecting point is chosen similarly to the splitting node in Join algorithm. We assume that the original group $\tilde{V}$ consists of g_1 members and the merged group $\tilde{V}'$ consists of g_2 members, where $g_1 \geq g_2$. Then, a new Boolean circuit C_{Merge} is processed as follows:

1. Find the connecting point from a Boolean circuit C that does not increase the depth of the circuit or minimize the complexity.
2. Connect the edge $\tilde{e}_{Merge}$ between connecting point from C and the root node of a circuit C', which is the Boolean circuit of group $\tilde{V}'$.
3. Contract the edge $\tilde{e}_{Merge}$ and put some Boolean operation to the node after this edge contraction.

4. Set the leaf nodes as the group members $\tilde{v}_1, \tilde{v}_2, \cdots, \tilde{v}_{g_1+g_2}$.

Figure 5 shows an example of Boolean circuit when two groups merge.

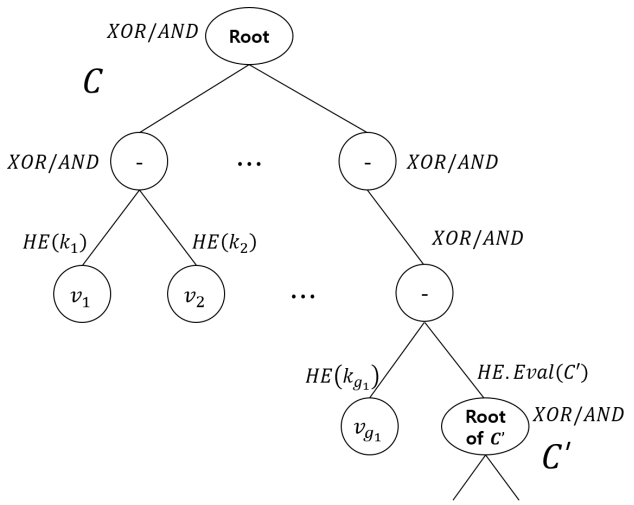

Figure 5: Merge algorithm in Parcel-D protocol

5.5 Partition Protocol

Assume that a network fault causes a partition of the group with g members. From the remaining member, this event seems to be a concurrent 'leave' of multiple members.

Starting from the leftmost leaf node of the Boolean circuit C, we run $\mathsf{Leave}(\tilde{v}_i, C)$ if $\tilde{v} \in \tilde{V}_j$. But, instead of removing all long-term secret keys of the vertices in $\tilde{V}_j$, the server keeps those long-term secret keys in separate box.

5.6 Relation on Boolean Circuit

For all membership events, one Boolean circuit is the minor of the other Boolean circuit. For addition event like Join and Merge algorithms, an original circuit C is the minor of the new circuits C_{Join} and C_{Merge}. Similarly, for revocation event like Leave and Partition algorithms, new circuits C_{Leave} and $C_{Partition}$ are the minor of the original circuit C. With this property, we can check the validity of the circuit after each membership event.

6 Instantiation

6.1 GSW13 Encryption Scheme

Gentry *et al.* [4] describes a comparatively simple fully homomorphic encryption (FHE) scheme based on LWE problem. They propose a new technique for building FHE schemes with the approximate eigenvector method. In GSW13 scheme, homomorphic addition and multiplication are just matrix addition and multiplicatio since the secret key is an *approximate eigenvector* of the ciphertext matrix $\mathbf{C}$, and the message μ is the *eigenvalue*. We briefly introduce how the basic GSW13 encryption scheme works.

BitDecomp$(\mathbf{a})$ operates the binary representation of given vector $\mathbf{a}$, where bits are ordered from least significant to most significant.

BitDecomp$^{-1}(\mathbf{a})$ is the inverse of BitDecomp algorithm.

Flatten$(\mathbf{a})$ outputs BitDecomp (BitDecomp^{-1})$(\mathbf{a})$ for a vector $\mathbf{a}$.

Powersof2$(\mathbf{a})$ computes $(a_1, 2a_1, \cdots, 2^{l-1}a_1, \cdots, a_k, 2a_k, \cdots, 2^{l-1}a_k)$ from $\mathbf{a} = (a_1, \cdots, a_k)$.

GSW.Gen$(params)$ generates the secret key $\mathbf{s}$ and public key $\mathbf{A}$.

GSW.Enc(pk, μ) encrypts a message $\mu \in \mathbb{Z}_q$, sample a uniform matrix $\mathbf{R} \in \{0,1\}^{N \times m}$ and output the ciphertext $C = \mathsf{Flatten}(\mu)I_N + \mathsf{BitDecomp}(\mathbf{R} \cdot \mathbf{A}) \in \mathbb{Z}_q^{N \times N}$.

GSW.Dec$(sk, \mathbf{C})$ computes $x_i \leftarrow \langle \mathbf{c}_i, \mathbf{v} \rangle$ where $\mathbf{c}_i$ is the i-th row of $\mathbf{C}$ and $\mathbf{v}$=$(v_1, v_2, \cdots, v_l) = (1, 2, \cdots, 2^{l-1})$. Then, it outputs $\mu' = \lceil x_i / v_i \rfloor$.

GSW.Eval$(\mathbf{C}_1, \cdots, \mathbf{C}_n, C)$ simply computes the matrix operation with the Boolean circuit C.

Note that no evaluation key is generated from GSW.Gen algorithm.

6.2 DingKE Protocol

Ding *et al.* [11] gives two two-party key exchange protocols based on lattice. Among them, LWE-based key exchange protocol runs as follows:

Step 1. (Setup Phase) From the security parameter n, q, α where $q > 2$ is prime, sample a uniformly random matrix $\mathbf{M} \leftarrow \mathbb{Z}_q^{n \times n}$.

Step 2. (Key Exchange Phase)

- Alice: Choose a secret vector $\mathbf{s}_A \leftarrow \mathcal{D}_{\mathbb{Z}^n, \alpha q}$. Then, Alice computes $\mathbf{p}_A = \mathbf{M}\mathbf{s}_A + 2\mathbf{e}_A \bmod q$, where $\mathbf{e}_A \leftarrow \chi$. Send $\mathbf{p}_A$ to Bob.
- Bob: Choose a secret element $\mathbf{s}_B \leftarrow \mathcal{D}_{\mathbb{Z}^n, \alpha q}$ and an error $e'_B \leftarrow \mathcal{D}_{\mathbb{Z}, \alpha q}$. Then, compute $K_B = \mathbf{p}_A^T \cdot \mathbf{s}_B + 2e'_B \bmod q$ and $\sigma \leftarrow S(K_B)$. Sample $\mathbf{e}_B \leftarrow \mathcal{D}_{\mathbb{Z}^n, \alpha q}$ and compute $\mathbf{p}_B = \mathbf{M}\mathbf{s}_B + 2\mathbf{e}_B \bmod q$.
- Bob: Send $(\mathbf{p}_B, \sigma)$ to Alice and obtain the shared key $SK_B = E(K_B, \sigma)$.
- Alice: Sample $e'_A \leftarrow \mathcal{D}_{\mathbb{Z}, \alpha q}$ and compute $K_A = \mathbf{s}_A^T \mathbf{p}_B + 2e'_A \bmod q$. Then, obtain $SK_A = E(K_A, \sigma)$.

6.3 DHL Protocol

To instantiate a Dynamic key exchange protocol using HE scheme based on Lattice (DHL protocol), we replace $\mathcal{HE}$, $\mathcal{TKE}$, and $\mathcal{SKE}$ with GSW13 scheme, DingKE protocol, and a simple XOR operation, respectively. Then, DHL protocol runs as follows:

Table 1: Comparison of quantum-resistant multi-party key exchange protocols

Method	Ding *et al.* [11]	Boneh *et al.* [12]	Choi and Kim [13]	Ours
Underlying Assumption[a]	LWE & Ring-LWE assumption	Cryptographic invariant maps from isogenies	any HE scheme	any HE scheme
Dynamic Setting[b]	X	X	X	**O**
Quantum Resistance[c]	X	**O**	△ [d]	△ [d]

[a] Cryptographic invariant maps from isogenies and protocols from LWE problem and Ring-LWE problem are all believed to be quantum-resistant.
[b] O: protocol supports membership addition and revocation, X: protocol does not support them.
[c] O: quantum-resistant, X: vulnerable to quantum computing attacks.
[d] △: our design is quantum-resistant if the underlying homomorphic encryption scheme was designed to be quantum-resistant.

Step 1. Each party pre-shares the master group key $sk \leftarrow \mathsf{GSW.Gen}(n, \alpha)$ with each other and the server runs DingKE protocol with each party to make the long-term secret key msk_i.

Step 2. Make the Boolean circuit C with g leaf nodes.

Step 3. Each party makes ephemeral session key k_i and encrypts it with its public key pk_i, $c_i = \mathsf{GSW.Enc}\ (pk_i, k_i)$.

Step 4. The server computes $c = \mathsf{GSW.Eval}(C, c_1, \cdots, c_g)$ with the Boolean circuit C and broadcasts c.

Step 5. The server sends the ephemeral key epk XOR-ed by the long term secret key msk_i.

Step 6. Each group member decrypts the evaluated value c and the ephemeral key epk. Then, each group member gets the proper session group key $k = \mathsf{GSW.Dec}(sk, c) \oplus epk$.

Note that we don't need an evaluation key to run GSW.Eval algorithm.

7 Comparison with Other Methods

In Table 1, we compare our construction with other quantum-resistant multi-party key exchange protocols like Ding *et al.*'s protocol [11], Boneh *et al.*'s protocol [12], and Choi and Kim's protocol [13].

Boneh *et al.*'s protocol misses the concrete design of cryptographic invariant maps from isogeny while other methods can be implemented. Also, our method can become a quantum-resistant multi-party key exchange protocol if we adopt any quantum-resistant HE scheme from the literature, like [4], as we described in Chapter 6.

Our method can be extended to dynamic setting while the other three methods only cover the static setting.

8 Conclusion and Future Work

In this paper, we construct a novel method to design a quantum-resistant dynamic key exchange protocol using HE scheme and compare it with other quantum-resistant multi-party protocols. Our protocol is secure against a passive adversary and it satisfies forward secrecy, backward secrecy, and key independence as well. For this, we give a methodology to describe each membership event in a circuit in terms of graph minor.

As future work, we consider the implementation of DHL protocol using some lattice-based libraries with HE tools like HElib and FHEW [26, 27] and evaluate the performance. Embedding multi-key HE scheme, to avoid the additional key exchange protocol between the server and the group member, is left for future work.

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] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Annual ACM on Symposium on Theory of Computing*, pp. 169–178, ACM, 2009.

[2] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE," *SIAM Journal on Computing*, vol. 43, no. 2, pp. 831–871, 2014.

[3] Z. Brakerski and V. Vaikuntanathan, "Fully homomorphic encryption from ring-LWE and security for key dependent messages," in *Advances in Cryptology–CRYPTO 2011*, pp. 505–524, Springer, 2011.

[4] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with er-

rors: Conceptually-simpler, asymptotically-faster, attribute-based," in *Advances in Cryptology–CRYPTO 2013*, pp. 75–92, Springer, 2013.

[5] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," *ACM Transactions on Computation Theory (TOCT)*, vol. 6, no. 3, p. 13, 2014.

[6] M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 24–43, Springer, 2010.

[7] J. H. Cheon, J.-S. Coron, J. Kim, M. S. Lee, T. Lepoint, M. Tibouchi, and A. Yun, "Batch fully homomorphic encryption over the integers," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 315–335, Springer, 2013.

[8] J. H. Cheon, M. Kim, and K. Lauter, "Homomorphic computation of edit distance," in *International Conference on Financial Cryptography and Data Security*, pp. 194–212, Springer, 2015.

[9] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *International Conference on Machine Learning*, pp. 201–210, 2016.

[10] S. Krendelev and I. Kuzmin, "Key exchange algorithm based on homomorphic encryption," in *Computer Science and Information Systems (FedCSIS), 2017 Federated Conference on*, pp. 793–795, IEEE, 2017.

[11] 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.

[12] D. Boneh, D. Glass, D. Krashen, K. Lauter, S. Sharif, A. Silverberg, M. Tibouchi, and M. Zhandry, "Multiparty non-interactive key exchange and more from isogenies on elliptic curves," *arXiv preprint arXiv:1807.03038*, 2018.

[13] R. Choi and K. Kim, "A novel non-interactive multi-party key exchange from homomorphic encryption," in *ProvSec Workshop 2018*, 2018.

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

[15] 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.

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

[17] 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.

[18] 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.

[19] H. Krawczyk, "HMQV: A high-performance secure Diffie-Hellman protocol (extended abstract)," in *Advances in Cryptology–CRYPTO 2005*, pp. 546–566, Springer, 2005.

[20] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, and D. Stehlé, "CRYSTALS–Kyber: a CCA-secure module-lattice-based KEM," *IACR Cryptology ePrint Archive 2017/634*, 2017.

[21] 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.

[22] 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.

[23] 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.

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

[25] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," in *Foundations of secure computation 4.11*, pp. 169–180, 1978.

[26] S. Halevi and V. Shoup, "Algorithms in HElib," in *International Cryptology Conference*, pp. 554–571, Springer, 2014.

[27] L. Ducas and D. Micciancio, "FHEW: bootstrapping homomorphic encryption in less than a second," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 617–640, Springer, 2015.