sensors

MDPI

Article

Quantum-Safe Group Key Establishment Protocol from Lattice Trapdoors

Teklay Gebremichael [1,*], Mikael Gidlund [1], Gerhard P. Hancke [2] and Ulf Jennehag [3]

[1] Department of Information Systems and Technology, Mid Sweden University, 852 30 Sundsvall, Sweden; mikael.gidlund@miun.se
[2] Department of Computer Science, City University of Hong Kong, Hong Kong 999077, China; gp.hancke@cityu.edu.hk
[3] Division of Industrial Systems, RISE—Research Institutes of Sweden, 852 33 Sundsvall, Sweden; ulf.jennehag@ri.se
* Correspondence: teklay.gebremichael@miun.se; Tel.: +46-010-1428-898

Abstract: Group communication enables Internet of Things (IoT) devices to communicate in an efficient and fast manner. In most instances, a group message needs to be encrypted using a cryptographic key that only devices in the group know. In this paper, we address the problem of establishing such a key using a lattice-based one-way function, which can easily be inverted using a suitably designed lattice trapdoor. Using the notion of a bad/good basis, we present a new method of coupling multiple private keys into a single public key, which is then used for encrypting a group message. The protocol has the apparent advantage of having a conjectured resistance against potential quantum-computer-based attacks. All functions—key establishment, session key update, node addition, encryption, and decryption—are effected in constant time, using simple linear-algebra operations, making the protocol suitable for resource-constrained IoT networks. We show how a cryptographic session group key can be constructed on the fly by a user with legitimate credentials, making node-capture-type attacks impractical. The protocol also incorporates a mechanism for node addition and session-key generation in a forward- and backward-secrecy-preserving manner.

Keywords: IoT group key; quantum-safe cryptography; lightweight cryptography; lattices; lattice-based crypotgraphy; lattice trapdoors; one-way function; learning with errors; LWE; short basis

check for updates

Citation: Gebremichael, T.; Gidlund, M.; Hanck, G.P.; Jennehag, U. Quantum-Safe Group Key Establishment Protocol from Lattice Trapdoors. *Sensors* **2022**, *22*, 4148. https://doi.org/10.3390/s22114148

Academic Editor: Nikos Fotiou

Received: 10 April 2022
Accepted: 27 May 2022
Published: 30 May 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

As the Internet of Things (IoT) becomes an integral component of an ever-increasing number of application areas, the need to secure it is also assuming an ever-increasing level of importance. Security in the IoT is a multi-faceted problem that includes the usual security concerns regarding authentication, integrity, and confidentially; and IoT-specific concerns, such as how to design and implement cryptosystems on small devices; how to prevent, and recover from, attacks that leverage physically accessing an IoT device deployed at a physically accessible place; and preventing, and recovering from, denial-of-service attacks [1].

In this paper, we focus on the problem of how to establish a cryptographic group key that can be used to encrypt or decrypt a group message to ensure confidentiality in a dynamic IoT network. Group communication is an efficient and fast message-sending mechanism, whereby a message is broadcast to a group of users that constitute a given secure network. In contrast to unicast communication, where a sender sends a message to each potential recipient, in group communication, a single message is broadcast to potentially many recipients. This helps minimize the amount of traffic generated and computations performed at the sender's end, since only a single message is broadcast to all potential recipients. In an IoT environment, where resources are generally constrained, minimizing traffic and computations at both the sender's and the receiver's ends is a

desirable requirement. This model of communication would, for example, be desirable in a time-critical IIoT system where a controller must send a command to a group of actuators [2].

Below are some use-case scenarios were group communication would be desirable:

- *Smart lighting*: A smart building may have its lighting devices grouped according to their location and connected to a switch, which acts as a gateway. It is important that the switch is able so send group messages to the devices to control lighting levels and related functions;
- *Software updates*: A gateway downloads a software update and simply broadcasts it to the group so that member nodes patch. The alternative is each device downloads the patch independently, which results in generating unnecessary traffic;
- *Emergency broadcasts*: The control center of some automation may be forced to stop or start many devices in the process with a single command, minimizing time and resource requirements;
- *e-Health*: A sensor implanted in a patient's body may broadcast readings to a group of receivers, such as nurses, doctors, and even chat servers.

Figure 1 shows an example where secure group communication would be preferable to unicast communication. Rather than send a message to each user in the secure multicast group, which is inefficient and computationally expensive, the *central entity* simply broadcasts an encrypted message that each user in the secure multicast group can decipher. The ability to send a single message to multiple recipients is also a desired property in time-sensitive applications, since sending a broadcast message is generally faster than sending a unicast message to each potential recipient.

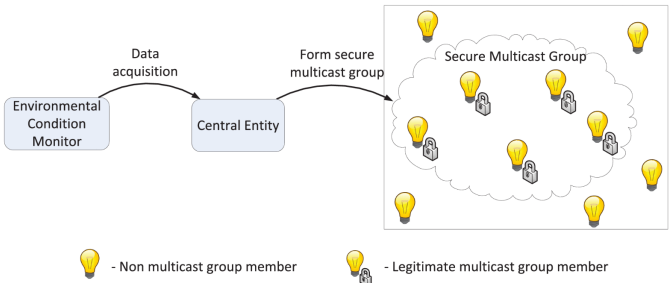

Figure 1. Multicast for light bulbs—Figure 1 of [3].

A critical problem in enabling group communication is designing an efficient mechanism to ensure the confidentiality and integrity of group messages [4]. The standard mechanism to address this problem is to have all users in a group share a secret cryptographic key, so that only a user that has a copy of the secret key can decrypt a group message. This raises the problem of how to establish the secret key among the users in the first place, and how to initiate re-keying securely. There have been various cryptographic group-key establishment protocol proposals that rely on cryptographic constructions that do not have conjectured resistance against quantum attacks [5,6]. Recently, there has been a push towards basing cryptographic constructions on mathematical problems that are believed to be computationally hard, even for a quantum computer [7]. Constructing cryptographic primitives from such mathematical problems has the apparent advantage of both preparing for a future where quantum computers are a reality and having cryptographic systems whose security comes from alternative sources other than the usual ones, such as discrete logarithm [8]. Lattice-based cryptographic constructions in particular have gained an increased level of interest from the research community [9].

A lattice-based cryptographic construction has several advantages. On the security front, it provides an opportunity to build cryptosystems based on a worst-case security assumption, as opposed to constructions based on classical problems, such as integer factorization (IP), that are based on average-case hardness [10]. A cryptosystem based

on worst-case hardness security has the interesting characteristic of being secure as long as a given computational problem is hard in the worst case, even for just one instance of the problem. A lattice-based constructions also has conjectured resistance to quantum-based attacks, as opposed to cryptosystems based on classical problems, which are easily breakable using quantum algorithms, as Shor [11] has demonstrated. In terms of richness, lattices enable the construction of cryptographic primitives and services that are otherwise impossible, such as homomorphic encryption. Regarding efficiency and ease of implementation, lattice-based computations involve simple linear-algebra computations, such as matrix-vector multiplication—which can be computed in parallel—and evaluations of simple linear functions.

Motivated by the aforementioned conjectured assumptions, in this paper, we present a cryptographic group-key establishment mechanism that is based on computational problems on lattices, for which there are no known efficient classical- or quantum-based algorithms. In particular, the cryptographic group-key establishment protocol is based on the concept of a lattice trapdoor, which helps us invert a lattice-based one-way function efficiently. Without the trapdoor, reverting the lattice-based one-way function is as hard as the search version of the well-known learning with errors (LWE) problem [12].

The main contributions of this paper are the following:

- A new method of designing a cryptographic group-key management protocol from lattice trapdoors is presented. Lattice trapdoors have been shown to be extremely versatile for designing various cryptographic primitives such as digital signatures and identity-based encryption (IBE) schemes [13]. The work presented here is a new addition to the list of cryptographic objects that can be built from lattice trapdoors. Since the computations involved are inherently lightweight, the protocol can be implemented and deployed in various IoT environments; hence, they contribute towards preparing the IoT for a future where quantum computers are a reality;
- A new mechanism for cryptographic group-key establishment where the group key is not stored in any of the constituent devices, so that an attacker cannot learn the cryptographic group key by physically examining a given device is also presented here. Moreover, we exhibit efficient mechanisms for adding or removing users from and to a secure group, in a manner that maintains standard security requirements, such as forward and backward secrecy.

The rest of this paper is organized as follows. In Section 2, we briefly discuss related work. In Section 3, we present basic lattice concepts that are essential for our subsequent discussion. In Section 4, we describe our network model for which the group-key establishment is proposed. In Section 5, we present the proposed protocol. In Section 6, we discuss the security and correctness of the proposed scheme. In Section 7, we discuss some implementation and optimization issues. In Section 8, we conclude the paper.

2. Related Work

The design and implementation of cryptographic group-key management protocols for the IoT has received a fair amount of attention from the research community [3,5,14]. The focus thus far has been on designing lightweight cryptographic group-key management protocols based on conventional cryptographic primitives, which are not quantum-safe. Transport layer-specific group-key establishment protocols, such as [14], suffer from the problem that they are not transport-layer-agnostic, in addition to not being quantum-safe. Other proposals, such as those based on elliptic curves [5], also lack the conjectured security against quantum-based attacks and do not scale well since they rely on a trusted anchor. Protocols such as [3,15] are lightweight and very convenient for the IoT, but are designed for one-to-many communication only, and are based on conventional constructions that are not quantum-safe.

Turning to lattice-based constructions, Lei et al. [16] have proposed an NTRU-based key-exchange protocol similar to the well-studied Diffie–Hellman key-exchange protocol [17]. An obvious limitation of this protocol is that it cannot be extended to a group

consisting of more than two users. The identity-based key exchange from the lattices protocol proposed in [18] also cannot be used in settings where there are more than two users in a network. In [19], a quantum-cryptography-based key-management protocol has been proposed without considerations for the resource-limited nature of devices on which it is to be implemented. Likewise, the quantum key distribution proposed in [20] is not intended for resource-constrained devices. In [21], the authors present an NTRU-based key-generation algorithm, but do not consider the problem of generating session keys or adapting the protocol to resource-constrained devices. The work in [22] addresses a related problem of how to secure IoT-like social networks based on the blockchain.

To the best of our knowledge, there does not appear to be any cryptographic group-key-management protocols based on lattices or any other quantum-safe primitive designed for resource-constrained IoT devices in the literature.

3. Preliminaries

3.1. Notation

We use bold lower-case letters, such as $\mathbf{x}$, to denote column vectors; for row vectors, we use the transpose $\mathbf{x}^t$. Bold upper-case letters, such as $\mathbf{A}$, denote matrices, while $\mathbf{A}^t$ represents its transpose. $\mathbf{Ax}$ denotes the usual matrix-vector multiplication, and $\langle \mathbf{a}, \mathbf{b} \rangle$ represents the usual inner product of vectors $\mathbf{a}$ and $\mathbf{b}$.

3.2. Lattice Trapdoors

We recall some basic definitions and theorems—without proofs—that are relevant to our discussion of the protocol. A more rigorous exposition of basic concepts about lattices can be found in [13] or [23], and the computational complexity analyses of various lattice-based computational problems can be found in [24].

Definition 1. *A lattice $\mathcal{L}$ is the set of all integer linear combinations of some linearly independent basis vectors $\mathbf{B} = \{\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_k\}$. Formally,*

$$\mathcal{L} = \mathcal{L}(\mathbf{B}) := \mathbf{B}\mathbb{Z}^k = \{\sum_{i=1}^{k} z_i \mathbf{b}_i \mid z_i \in \mathbb{Z}\},$$

where $\mathbf{B}$ is the matrix whose columns are the n-dimensional column vectors $\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_k$ [13].

A given lattice $\mathcal{L}$ can be generated by infinitely many *bases*. Any two bases $\mathbf{B}_1$ and $\mathbf{B}_2$ of a lattice $\mathcal{L}$ are related by a *unimodular* integer matrix $\mathbf{U}$ such that $\mathbf{B}_1 = \mathbf{B}_2\mathbf{U}$, and vice versa with a different $\mathbf{U}$ [25].

A notion of *goodness* can be associated with lattices: a basis is considered *good* if its vectors are *short* under some reasonable notion of *norm* (typically the Euclidean norm), and are orthogonal or close to orthogonal to each other. A basis is otherwise considered *bad* [26]. As we will see later, some lattice problems, which are computationally hard on their own, become computationally easy if one has access to a good basis of the lattice. It is this important fact that we will mainly use in our construction.

Definition 2. *The minimum distance of a lattice $\mathcal{L}$ is the length of a shortest nonzero lattice vector:*

$$\lambda_1(\mathcal{L}) := \min_{\mathbf{v} \in \mathcal{L} \setminus \{\mathbf{0}\}} \|\mathbf{v}\|.$$

In general, $\lambda_i(\mathcal{L})$ is the smallest r such that $\mathcal{L}$ has i linearly independent norm vectors at most r. Here and everywhere else in this text, $\|.\|$ denotes the Euclidean norm.

Theorem 1 (Intersection of Lattices)**.** *Given two integer lattices $\mathcal{L}(\mathbf{B}_1)$ and $\mathcal{L}(\mathbf{B}_2)$, the set $\mathcal{L}(\mathbf{B}_1) \cap \mathcal{L}(\mathbf{B}_2)$ is also a lattice whose basis can be computed efficiently, i.e., in polynomial time in the dimension of the lattices. It is easy to prove by induction that this can be extended to any finite number of lattice bases [24]. Computing the intersection of two lattices is generally a slow process,*

but in our proof-of-concept implementation, we have devised a relatively fast process described as follows: to compute $\mathcal{L}(\mathbf{B}_1) \cap \mathcal{L}(\mathbf{B}_2)$*, first compute the dual* $\mathcal{L}(\mathbf{B}_1)^*$ *and* $\mathcal{L}(\mathbf{B}_2)^*$ *of* $\mathcal{L}(\mathbf{B}_1)$ *and* $\mathcal{L}(\mathbf{B}_2)$*, respectively. Then,* $HNF(\mathbf{B}_1^* \cap \mathbf{B}_2^*)$ *is the intersection of the two lattices, where HNF is the Hermite normal form, and* $\mathbf{B}_1^*$ *and* $\mathbf{B}_2^*$ *are the bases of* $\mathcal{L}(\mathbf{B}_1)^*$ *and* $\mathcal{L}(\mathbf{B}_2)^*$*, respectively.*

Definition 3 (Learning with errors (LWE))**.** *Let* $\mathcal{X}$ *be a discrete Gaussian of a small width [27]. For a vector* $\mathbf{s} \in \mathbb{Z}_q^n$*, called the secret, the LWE distribution* $\mathbf{A}_{\mathbf{s},\mathcal{X}}$ *over* $\mathbb{Z}_q^n \times \mathbb{Z}_q$ *is sampled by choosing* $\mathbf{a} \in \mathbb{Z}_q^n$ *uniformly at random, choosing* $\mathbf{e} \leftarrow \mathcal{X}$*, and outputting the vector* $(\mathbf{a}, \mathbf{b} = \langle \mathbf{s}, \mathbf{a} \rangle + \mathbf{e} \mod q)$ *[12].*

3.3. Hard Lattice Problems

The following lattice problems—among many other lattice problems—are conjectured to be computationally hard, in the sense that there are no known classical or even quantum algorithms that can efficiently solve them.

Definition 4 (Search-LWE)**.** *Given m LWE samples, find the vector* $\mathbf{s}$*.*

Definition 5 (Decision-LWE)**.** *Given m independent samples* $(\mathbf{a}_i, \mathbf{b}_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$*, where every sample is distributed according to either* $\mathcal{A}_{\mathbf{s},\mathcal{X}}$ *(for a fixed* $\mathbf{s}$*) or the uniform distribution, distinguish which one of the two is the case.*

Definition 6 (Bounded distance decoding problem (BDD_γ))**.** *Given a basis* $\mathbf{B}$ *of an n-dimensional lattice* $\mathcal{L} = \mathcal{L}(\mathbf{B})$*, and a target* $\mathbf{t} \in \mathbb{R}^n$*, with the guarantee that* $dist(\mathbf{t}, \mathcal{L}) < \lambda_1(\mathcal{L})/(2\gamma(n))$*, find the unique vector* $\mathbf{v}$ *such that* $\|\mathbf{v} - \mathbf{t}\| < d$*, where* γ *is some function of n, where n is the dimension of* $\mathcal{L}$*.*

One can easily show that search-LWE can be cast as an *average-case* BDD problem on the following family of lattices:

$$\mathcal{L}(\mathbf{A}) := \{\mathbf{A}^t\mathbf{s} : \mathbf{s} \in \mathbb{Z}_q^n\} + q\mathbb{Z}^m, \tag{1}$$

where $\mathbf{A}$ is the matrix whose columns are the $\mathbf{a}_i \in \mathbb{Z}_q^n$ samples.

Although the BDD problem is computationally hard on its own, with a proper *trapdoor*, it can be easily solved. We refer the reader to [24] for a rigorous analysis of the computational complexity of each problem, and various classical and quantum reductions between different problems.

4. System Model and Security Requirements

The system consists of a set of IoT users, denoted by a universe $\mathcal{U}$, as depicted in Figure 2. Each user in the system is denoted by $\mathcal{U}_i$, with i ranging from 1 to n, where $n = |\mathcal{U}|$. A secure multicast group is a subset $\mathcal{G}$ of $\mathcal{U}$. The aim of the proposed protocol is to enable each user in $\mathcal{G}$ to agree on a shared secret. Note that $\mathcal{G}$ is dynamic, since we want to add or remove users to and from the group.

We assume that each pair $\mathcal{U}_i$ and $\mathcal{U}_j$, $i \neq j$, in $\mathcal{U}$ have an authenticated channel between them. This assumption allows us to rule out an active adversary that attempts to break the protocol by injecting messages by masquerading as a legitimate user during the key-establishment process. Although this assumption abstracts the practical problem of authentication away, it allows us to focus only on aspects related to the secure cryptographic group-key-establishment mechanism.

To simplify the analysis of the security of the system, we assume a reliable communication infrastructure (e.g., no packet loss due to interference).

Regarding security requirements, the first requirement is that a passive adversary—i.e., an adversary that attempts to learn something useful by only observing what is out in the open [28]—cannot learn the shared secret key without solving a computational lattice problem that is conjectured to be hard.

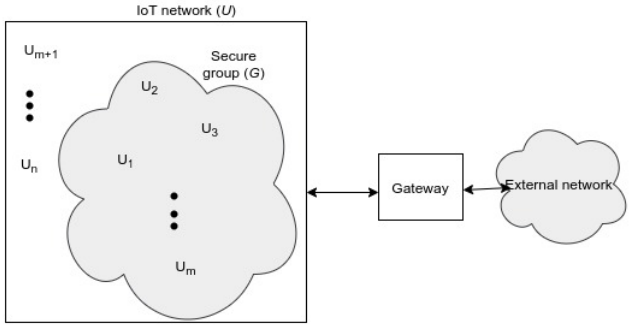

Figure 2. Network model. The network consists of a gateway and a set of nodes, supported by a communication infrastructure. All or a part of the nodes may be members of the secure group at a given time, as shown in the figure (m of n nodes are in the secure group).

The second requirement, which follows from the first as a corollary, is that a passive adversary cannot decrypt a group message that was encrypted using the shared secret key without solving a computationally hard lattice problem. We also require that a new user is added to or removed from the group in a manner that preserves forward and backward secrecy [29]. Finally, we require that group sessions keys be independent from each other to achieve forward and backward secrecy.

In summary, the security requirements are the following:

- **Requirement 1**: Any user $\mathcal{U}_i \notin \mathcal{G}$ cannot learn the group key without solving a hard lattice problem;
- **Requirement 2**: Each session group key is independent from any other session key;
- **Requirement 3**: A user $\mathcal{U}_i$ is added to $\mathcal{G}$ in a forward- and backward-secrecy-preserving manner.

The methodology consists in constructing a scheme and proving that it satisfies the above-mentioned requirements, assuming that some lattice-related problems are computationally hard. To demonstrate feasibility, a toy version of the protocol will be implemented and results analyzed.

5. Proposed Scheme

We start our presentation of the protocol by introducing the concept of a lattice trapdoor.

5.1. Setting Up a Lattice Trapdoor

Informally, a lattice trapdoor is a piece of information that enables one to invert a one-way function defined on a lattice, which is otherwise hard to invert on its own. A lattice trapdoor, in particular, enables one to solve the BDD problem.

Generally, there are two types of lattice trapdoors: *short bases* and *gadget trapdoors* [30]. We describe the proposed protocol using the notion of a short basis as trapdoor, although one can also use gadget trapdoors without having to change much in the protocol construction.

An important property of any given lattice is that it can have arbitrarily many (short) bases. Any one of the short bases can be used to invert a one-way function defined on that particular basis. If one can devise a mechanism for generating a lattice along with many short bases—one for each user—and securely save each short basis in each user, then each user can use its short basis to invert the one-way function and extract a secret value, which can then be used as a group key. That is the main idea behind the proposed scheme.

The first part of the protocol deals with setting up a trapdoor for each user $\mathcal{U}_i$ in the group $\mathcal{G}$. Although it is likely that each user will end up having a different trapdoor, we will show that each trapdoor will enable each user to invert a common one-way function. This is at the heart of the protocol—the fact that different but related trapdoors can be used to invert a given one-way function.

The lattice-based one-way function that we will rely on is the following LWE function:

$$g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) := \mathbf{s}^t\mathbf{A} + \mathbf{e}^t \mod q, \tag{2}$$

where $\mathbf{s} \in \mathbb{Z}_q^n$ is chosen uniformly at random, and $\mathbf{e} \in \mathbb{Z}^m$ is a small vector chosen from the LWE error distribution $\mathcal{X}^m$. Inverting this function is equivalent to the search-LWE problem [12]. With a trapdoor information, the function can easily be inverted. The trapdoor is a short basis $\mathbf{S}$ of the lattice

$$\mathcal{L}(\mathbf{A}) := \{\mathbf{A}^t\mathbf{s} : \mathbf{s} \in \mathbb{Z}_q^n\} + q\mathbb{Z}^m.$$

With a short basis $\mathbf{S}$, the function in Equation (2) can easily be inverted (i.e., $\mathbf{s}$ and $\mathbf{e}$ can be recovered), given that the the parameters are chosen appropriately [30].

There are two challenges associated with setting up a trapdoor corresponding to the function in Equation (2). The first challenge is how to generate a random lattice $\mathcal{L}(\mathbf{A})$ and its corresponding short basis $\mathbf{S}$. The second challenge is how to generate different short bases $\mathbf{S}_i$ for each $\mathcal{U}_i \in \mathcal{G}$, each corresponding to the lattice $\mathcal{L}(\mathbf{A})$.

In order to generate a random lattice along with a short basis, we use Ajtai's method [31]. We recall it here as a theorem:

Theorem 2. *There is an efficient randomized algorithm that, given positive integers n, q and $m \geq Cn \log q$ for some constant C, outputs a nearly uniformly random matrix $\mathbf{A} \in \mathbb{Z}^{n \times m}$ specifying the integer lattice $\mathcal{L} = \mathcal{L}^{\perp}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^{\mathbf{m}} : \mathbf{A}\mathbf{x} = \mathbf{0} \mod q\} \subseteq \mathbb{Z}^m$, along with a basis $\mathbf{S} \in \mathbb{Z}^{m \times m}$ of $\mathcal{L}$ whose vectors have norms bounded by poly $(n, \log q)$.*

It is worth noting that the lattice in Theorem 2 corresponds to the short integer solution (SIS) problem [32], and is a dual [33] of the lattice in Equation (1). Ajtai's algorithm is slow and complex for small IoT users. However, Micciancio et al. [30] have improved upon Ajtai's algorithm. Due to [30], it is now possible to generate a random SIS lattice along with a corresponding short basis $\mathbf{S}$ efficiently and easily. This addresses the first challenge.

Regarding the second problem, assume that the users in $\mathcal{G}$ are indexed $\mathcal{U}_1$ through $\mathcal{U}_k$, where $k = |\mathcal{G}|$. Without loss of generality, we may assume that $\mathcal{U}_1$ is the group leader, by which we mean that it starts the trapdoor-setup process.

User $\mathcal{U}_1$ starts the trapdoor-setup process by broadcasting a *group-join* message. The *group-join* message consists of a group ID chosen by user $\mathcal{U}_1$ and a short description of the group. The *group-join* broadcast message serves as a signal to to each user $\mathcal{U}_i \in \mathcal{U}$ to join the group. We assume that a subset $\mathcal{G}$ of the users in $\mathcal{U}$ will respond to the *group-join* message positively, i.e., will want to join the group.

Each user that wants to join the group simply broadcasts its ID. After a predefined amount of time has elapsed, the leader broadcasts the set $\mathcal{G} = \{ID_1, ID_2, \cdots ID_k\}$ of all received IDs. The set $\mathcal{G}$ gives each user $\mathcal{U}_i$ information about the user whose ID precedes it. Broadcasting the set $\mathcal{G}$ helps the users in $\mathcal{G}$ form a logical ring structure in which each user knows the user before it, assuming that a notion of ordering can be associated with respect to the IDs.

User $\mathcal{U}_1$ starts the trapdoor-setup process by generating a lattice $\mathcal{L}_1 = \mathcal{L}_1^{\perp}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A}\mathbf{x} = 0 \mod q\} \subseteq \mathbb{Z}^m$, along with a short basis $\mathbf{S}_1$, using the mechanism stated in Theorem 2. It keeps $\mathbf{S}_1$ secret and makes $\mathcal{L}_1^{\perp}(\mathbf{A})$ public. Note that this corresponds to the idea of generating a *good* and *bad* basis pair, so that the *good* basis is used as a trapdoor to a one-way function based on the *bad* basis.

Each user $\mathcal{U}_i$, where i ranges from 2 to k, does the following: upon receiving $\mathcal{L}_{i-1}$, it generates the $\mathcal{L}_i$ and $\mathbf{S}_i$ pair as before, but with the additional requirement that $\mathcal{L}_{i-1} \cap \mathcal{L}_i \neq \emptyset$. The problem of how to generate such lattice pairs has been addressed in [25]. We use a slightly modified version of the algorithm discussed in [25] (please see Algorithm 1). Given a lattice with a bad basis, Algorithm 1 generates a lattice with a Hadamard ratio [34] of 0.75, which is considered a good basis. Figure 3 depicts a lattice with a good basis gener-

ated from a lattice with a bad basis (a Hadamard ratio of 0.37). In our proof-of-concept implementation, we have used a less-optimized version of one of the algorithms proposed in [25]. User $\mathcal{U}_i$ keeps $\mathbf{S}_i$ private and makes $\mathcal{L}_i \cap \mathcal{L}_{i-1}$ public. This round finishes with $\mathcal{U}_k$ generating $\mathbf{S}_k$ and its corresponding $\mathcal{L}_k$, with the requirement stated above. At the final round, $\mathcal{U}_k$ makes $\mathcal{L}_k \cap \mathcal{L}_{k-1}$ public. We call this last lattice $\mathcal{L}_\mathbf{G}$. The trapdoor setup process is shown in Algorithm 2.

$$\begin{bmatrix} 31 & 17 & 93 & 22 & 36 \\ 47 & 43 & 24 & 67 & 53 \\ 78 & 129 & 36 & 21 & 85 \\ 56 & 43 & 7 & 28 & 5 \\ 93 & 8 & 89 & 11 & 36 \end{bmatrix} \; - - > \; \begin{bmatrix} 91 & 16 & 7 & 3 & 12 \\ 67 & 52 & 80 & 32 & 98 \\ 67 & 43 & 36 & 67 & 90 \\ 89 & 74 & 54 & 21 & 32 \\ 37 & 5 & 7 & 85 & 8 \end{bmatrix}$$

Figure 3. Public lattice → new private lattice.

Algorithm 1: A Short-Basis Lattice from a Long-Basis Lattice

Input: Bad-basis lattice $\mathbb{A}^{m \times n}$
Result: Short-basis lattice $\mathbb{A}^{*m \times n}$
$line = (random()\%m/2) + 1$
$\mathbb{A}^{'} = gram - schmidt(\mathbb{A})$
for $i = 1$ *to* m **do**
  for $j = 1$ *to* n **do**
    $v = random()$
    $\mathbb{A}^{'}[i][j] = \mathbb{A}^{'}[i][j]/v$
  end
end
while $\frac{det(\mathbb{A}^{'})}{||\mathbb{A}_1||_2 \cdot ||\mathbb{A}_2||_2 \cdots ||\mathbb{A}_n||_2})^{1/n} \leq 0.75$ **do**
  $\mathbb{A}^{'} = gram - schmidt(\mathbb{A}^{'})$
end
Output $\mathbb{A}^{'}$ as the lattice with a good basis

Algorithm 2: Trapdoor Setup

Result: $\mathbf{S}_i$, $1 \leq i \leq k$; $\mathbf{A} \in \mathbb{Z}^{n \times m}$ defining $\mathcal{L}_\mathbf{G}$
Input: $m, q, n, k, \mathcal{C}$
$\mathcal{U}_1$ broadcasts *group-join* message and sets a timer;
Each $\mathcal{U}_i \in \mathcal{U}$ that wants to join the group to which it sends its ID;
$\mathcal{U}_1$ creates $\mathcal{G} = \{ID_1, ID_2, \cdots ID_k\}$ and broadcasts it;
$\mathcal{U}_1$ generates the $\mathbf{S}_1$ and $\mathcal{L}_1$ pair, and broadcasts $\mathcal{L}_1$;
for $i = 2$ *to* $k - 1$ **do**
  $\mathcal{U}_i$ generates the $\mathbf{S}_i$ and $\mathcal{L}_i$ pair, such that $\mathcal{L}_{i-1} \cap \mathcal{L}_i \neq \emptyset$
  $\mathcal{U}_i$ keeps $\mathbf{S}_i$ secret, and broadcasts $\mathcal{L}_i$
end
$\mathcal{U}_k$ generates the $\mathbf{S}_k$ and $\mathcal{L}_k$ pair
$\mathcal{U}_k$ keeps $\mathbf{S}_k$ secret, and broadcasts $\mathcal{L}_\mathbf{G} := \mathcal{L}_k$ as the public parameter

We remark that $\mathcal{L}_\mathbf{G}$ is represented by the matrix $\mathbf{A} \in \mathbb{Z}^{n \times m}$, which defines a *bad* basis for the lattice

$$\mathcal{L}(\mathbf{A}) := \{\mathbf{A}^t \mathbf{s} : \mathbf{s} \in \mathbb{Z}_q^n\} + q\mathbb{Z}^m.$$

The matrix $\mathbf{A}$ is a public parameter, and can be suitably stored in a public directory which each user $\mathcal{U}_i$ can access, or be stored locally in each user.

See Figure 4 for an instance of the trapdoor-initialization process with four users.

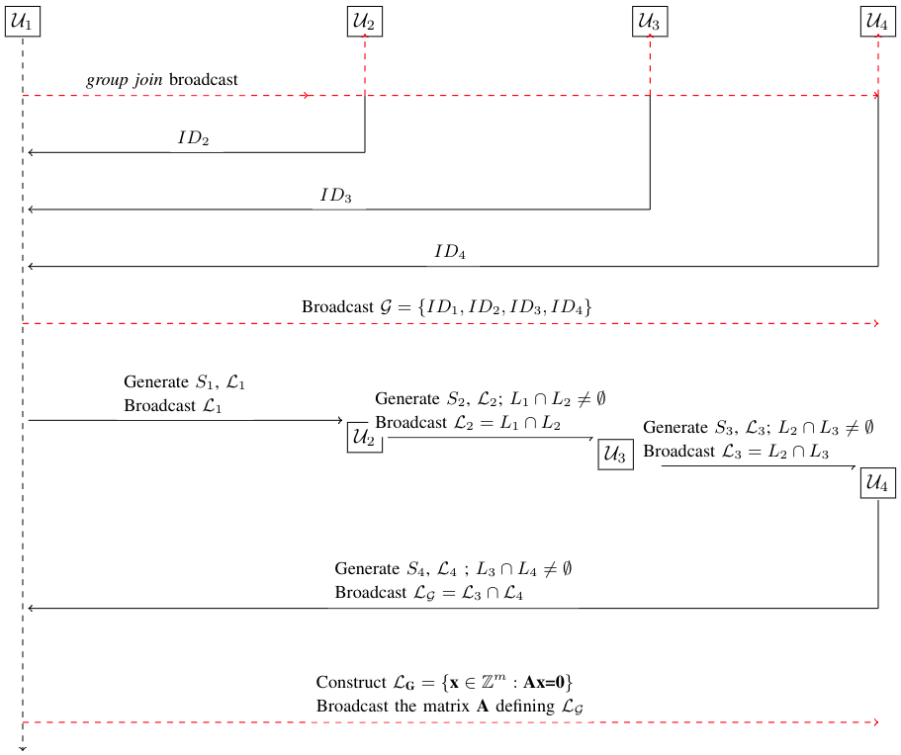

Figure 4. Trapdoor-setup procedure with four users.

5.2. Group Key from a One-Way Function with a Trapdoor

First, we note that $\mathcal{L}_\mathbf{G} \subseteq \mathcal{L}(\mathbf{S}_i), 1 \leq i \leq k$. This is true because $\mathcal{L}_\mathbf{G} = \mathcal{L}(\mathbf{S}_1) \cap \mathcal{L}(\mathbf{S}_2) \cap \cdots \cap \mathcal{L}(\mathbf{S}_m)$, which is a subset of $\mathcal{L}(\mathbf{S}_i)$, for each $i \in \{1, 2, \cdots, k\}$.

By construction, $\mathcal{L}_\mathbf{G}$ is a lattice of the form

$$\mathcal{L}_\mathbf{G} = \{\mathbf{x} : \mathbf{Ax} = 0\}$$

with parameters as specified in Theorem 2, and is specified by the parity-check [35] integer matrix $\mathbf{A}$.

Our one-way function is the following

$$g_\mathbf{A}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t \mod q, \tag{3}$$

where the matrix $\mathbf{A}$ is constructed as defined in Section 5.2, and $\mathbf{e}, \mathbf{s}$ and q are chosen according to the distributions specified in Definition 5.

5.3. The Group Key and Group-Message Encryption

The group key is some function f of $\mathbf{s} \in \mathbb{Z}_q^n$. The function f could be a pseudo-random function that takes $\mathbf{s}$ as its seed and expands it to a required bit-length suitable for the encryption algorithm agreed upon in the group. It could well be a function that truncates certain bits from $\mathbf{s}$.

More formally, let $\mathcal{E}$ be the symmetric-key-encryption algorithm used to encrypt group messages, and let $\mathcal{M}$ be the message space. To encrypt a message $m \in \mathcal{M}$, $\mathcal{U}_i$ chooses $\mathbf{s} \in \mathbb{Z}_q^n$ and an error $\mathbf{e} \in \mathbb{Z}^m$, according to the distributions specified in Equation (1), and outputs

$$c = \mathcal{E}(f(s), m); \tag{4}$$

$\mathcal{U}_i$ then computes the one-way function $g_{(\mathbf{s},\mathbf{e})}$ and appends it to c. The final combined message is then simply broadcasted. Algorithm 3 describes how the encryption process works.

In the next section, we prove that only a user in $\mathcal{G}$ can recover $\mathbf{s}$ and, hence, decrypt the encrypted group message c.

Algorithm 3: Group-Message Encryption

Result: Encrypted group message c, n, q
Input: Encryption algorithm $\mathcal{E}$, message m, pseudo-random function f
Choose $s \in \mathbb{Z}_q^n$ randomly
Compute $key = f(s)$
Compute $c = \mathcal{E}(key, m)$
Broadcast c as the encrypted group message

5.4. Decrypting a Group Message

Decrypting a group message c involves first recovering s and then using the agreed-upon decryption algorithm to recover m.

To recover s, each user $\mathcal{U}_i \in \mathcal{G}$ uses its short-basis trapdoor $\mathbf{S}_i$ to compute:

$$\mathbf{x}^t = \mathbf{b}^t(\mathbf{S}_i) = e^t(\mathbf{S}_i) \mod q,$$

where

$$\mathbf{b}^t = g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t(\mathbf{A}) + \mathbf{e}^t \mod q$$

The vector $\mathbf{x}$ is then lifted to its canonical representative [26] $\overline{\mathbf{x}} \in [-\frac{q}{2}, \frac{q}{2})^m$, from which we obtain

$$\mathbf{e} = \overline{\mathbf{x}}^t(\mathbf{S}^{-1})$$

Computing $\mathbf{s}$, given $\mathbf{e}$, is straightforward, as shown in [13].

5.5. Adding a New Node to a Group

In a dynamic IoT network, there could arise a need to add a new user $\mathcal{U}_i \in \mathcal{U} \setminus \mathcal{G}$ to $\mathcal{G}$. A standard security requirement in such a scenario is that $\mathcal{U}_i$ does not learn any of the $\mathbf{s}$ values used before it joins the group, nor is any group message exchanged prior to joining. This is the usual backward-secrecy requirement [36].

Node addition is effected as follows: $\mathcal{U}_i$ reads off the public parameter $\mathcal{L}_\mathbf{G}$. It then generates an $\mathbf{S}_i$ and $\mathcal{L}_i$ pair, with the requirement that $\mathcal{L}_i \cap \mathcal{L}_\mathbf{G} \neq \emptyset$. Finally, node $\mathcal{U}_i$ broadcasts $\mathcal{L}_i \cap \mathcal{L}_\mathbf{G}$ as the new $\mathcal{L}_\mathbf{G}$. All other nodes in $\mathcal{G}$ update their copy of $\mathcal{L}_\mathbf{G}$ accordingly. The node-addition process is shown in Algorithm 4.

Notice that each user $\mathcal{U}_j \in \mathcal{G} \setminus \{\mathcal{U}_i\}$ does not update its copy of $\mathbf{S}_j$ when $\mathcal{U}_i$ is added to $\mathcal{G}$.

Algorithm 4: Adding a Node $\mathcal{U}_i$ to $\mathcal{G}$

Result: A new $\mathcal{L}_\mathbf{G}$, representing the new group parameter
$\mathcal{U}_i$ requests group leader $\mathcal{U}_1$ to send it a copy of $\mathcal{L}_\mathbf{G}$;
$\mathcal{U}_i$ generates $\mathbf{S}_i$ and $\mathcal{L}_i$, such that $\mathcal{L}_i \cap \mathcal{L}_\mathbf{G} \neq \emptyset$
$\mathcal{U}_i$ sends $\mathcal{L}_i \cap \mathcal{L}_\mathbf{G}$ to $\mathcal{U}_1$
$\mathcal{U}_1$ broadcasts $\mathcal{L}_i \cap \mathcal{L}_\mathbf{G}$ as the new $\mathcal{L}_\mathbf{G}$
for *each* $\mathcal{U}_j \in \mathcal{G}$ **do**
  $\mathcal{U}_j$ updates its $\mathcal{L}_\mathbf{G}$ value to $\mathcal{L}_i \cap \mathcal{L}_\mathbf{G}$
end

5.6. Removing a Node from a Group

Once a secure group has been established, the need to remove a node $\mathcal{U}_i \in \mathcal{G}$ could emerge for various reasons [37]. A common security requirement in such a scenario is that $\mathcal{U}_i$ be unable to read future messages or have access to any of the future session keys [38].

In the proposed protocol, the only way to remove a node while maintaining the said security requirement is to re-run the protocol from scratch, excluding the node to be removed. This is a straightforward but inefficient method. An improved solution is left for future work.

5.7. Generating Session Keys

Generating session keys in such a way that security requirements such as forward and backward secrecy are maintained is not an easy problem. The proposed protocol easily solves these problems since the session key is a value derived from $\mathbf{s}$, where $\mathbf{s}$ is chosen uniformly at random at each round. That is, if we let t represent the round number, the $\mathbf{s} \in \mathbb{R}^n$ used at round t_i is independent from the $\mathbf{s} \in \mathbb{R}^n$ used at round t_{i-1} or t_{i+1}. So, if an attacker ever succeeds in learning the value of $\mathbf{s}$ at any particular round, the attacker does not learn anything about previous or future values of $\mathbf{s}$.

In each round t, a user $\mathcal{U}_i$ that intends to broadcast a group message m chooses $s_t \in \mathbb{Z}_q^n$ randomly, computes $c = \mathcal{E}(f(s_t), m)$, and broadcasts c. A user $\mathcal{U}_j$ uses its short basis S_j to recover s_t and runs the decryption algorithm using $f(s_t)$ as the decryption key. If t and $t+1$ represent two consecutive rounds, or group sessions, then knowledge of s_t does not help an attacker learn anything about s_{t+1}, nor does knowledge of s_{t+1} reveal anything useful about s_t. If each s_i is chosen sufficiently randomly and the LWE function is indeed hard, then independence of session keys from each other follows easily.

6. Proof of Correctness and Security

By *correctness*, we mean that each $\mathcal{U}_i$ in $\mathcal{G}$ ends up recovering the same value of $\mathbf{s}$. By *secure*, we mean that no user $\mathcal{U} \notin \mathcal{G}$ can either classically or quantumly recover $\mathbf{s}$. We state each claim as a theorem and provide a proof for each.

Theorem 3 (Correctness)**.** *The group-key-agreement protocol is correct.*

Proof. This amounts to proving that each $\mathbf{S}_1$ is a good (that is, short) basis for $\mathcal{L}_\mathbf{G}$. Strictly speaking, this is generally not the case since $\mathcal{L}(\mathbf{S}_i) \supset \mathcal{L}_\mathbf{G}$, for each $i \in \{1, 2, \cdots, m\}$. However, since $\mathcal{L}_\mathbf{G} \subseteq \mathcal{L}(\mathbf{S}_i)$, and since each $\mathbf{S}_i$ is a short basis by construction, each $\mathbf{S}_i$ is a trapdoor for $g_{(\mathbf{s},\mathbf{e})}$. The details of how $\mathbf{s}$ can be recovered given a short basis $\mathbf{S}$ can be seen in [13] or [39]. □

Theorem 4 (Security)**.** *The group-key-agreement protocol meets the security requirements specified in Section 4.*

We only consider a passive adversary; that is, we consider an adversary that can attempt to learn the group key or some partial information about the group key by intercepting data that is in the open. We do not consider an active adversary.

- **Requirement 1**: An attacker that attempts to recover one of the short bases $\mathbf{S}_i$ or generate a new short basis for the lattice $\mathcal{L}_\mathbf{G}$ has to solve a version of the (approximate) shortest vector problem (SVP) [40], which we conjecture to be computationally hard. An attacker that attempts to directly recover the secret $\mathbf{s}$ by inverting the one-way function g has to solve a version of the BDD problem, since the two problems are computationally equivalent. The BDD problem is again conjectured to be computationally hard. Therefore, a passive adversary cannot recover $\mathbf{s}$ without solving either one of the two lattice problems. Since we conjecture the problems to be hard even for quantum computers, we conclude that the protocol is quantum-safe. The confidentiality of a message m encrypted using s follows as a corollary, assuming that the encryption

scheme $\mathcal{E}$ is quantum-safe. The encryption scheme could be a standard protocol, such as the AES, with a proper key length to account for possible quantum-based attacks;
- **Requirement 2**: This easily follows from the fact that, at each round, a user $\mathcal{U}_i$ chooses **s** randomly;
- **Requirement 3**: By design, the addition of a new node $\mathcal{U}_j$ into $\mathcal{G}$ forces each node $\mathcal{U}_i \in \mathcal{G}$ to acquire a new $\mathcal{L}_{\mathbf{G}}$ that is not related to the previous $\mathcal{L}_{\mathbf{G}}$ or a future $\mathcal{L}_{\mathbf{G}}$.

7. Results and Performance Analysis

We have implemented a proof of concept of the protocol on a simulated IoT network on Contiki OS [41]. Using a 300×300 full-rank lattice, which would be insecure for practical purposes due to the small parameter sizes, each node takes 4.94 s on average to set up a trapdoor. The restriction to small lattices was motivated by the memory-size-related limits of the IoT-device simulators provided by Contiki [22]. One can extrapolate the encryption and decryption times for larger lattices for the results obtained for smaller ones.

Encryption takes 0.023 s on average and decryption takes 3.04 s on average. With proper optimization, these numbers can be improved, making the protocol fast.

Figure 5 depicts the running times for setting up a trapdoor on each node, encrypting, and decrypting (128-bit key, 128-bit message) an IoT network consisting of 10 Raspberry Pi 3Bs [42].

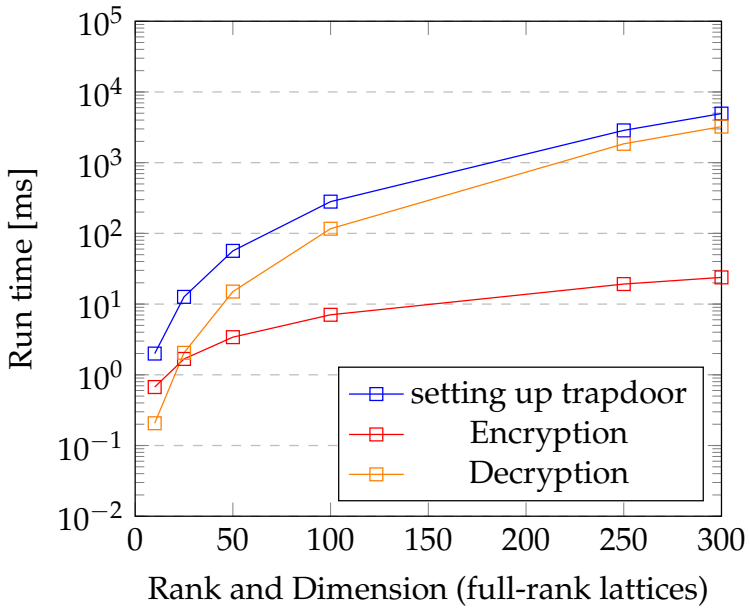

Figure 5. Running time.

Regarding error rate, each node recovers the same s value—on average—92 percent of the time, meaning that 8 times out of 100, decryption does not work. This is not a significant problem, however, since one can use a reconciliation technique [43] to resolve any discrepancies in decryption.

The main positive aspect about the performance of the protocol is that once a trapdoor is set up on each node, all the other functions—encryption, decryption, session-key generation, and node addition—are effected in constant time, regardless of the size of the network. Setting up trapdoors takes $\Theta(n)$, where n is the number of nodes in $\mathcal{G}$. This is so because the setting up of trapdoors is conducted in a sequential manner, but in principle, it can be effected in parallel. The fact that most of the main functions of the protocol can be achieved in constant time is a considerable improvement over other protocols available in the literature, which take at least linear time in the size of the input [6]. Operations that take constant time are especially desirable in larger networks.

In concrete terms, setting up a trapdoor, the most time-consuming function, took less that 5 s, albeit in a network with 10 nodes and a small lattice (300 by 300 dimensional lattice). The other key-agreement-related functions took less than a second to complete.

Analyzing the protocol in relation to other protocols that solve the same or a related problem is impractical due to the specificity of the design—different primitives, different setups, and different construction. At an abstract level, our protocol outperforms related protocols with respect to the running time and amount of storage required. See Table 1 for the running times of each function of the proposed protocol. All the protocols discussed in Section 2 that deal with the same problem have at least a $\Theta(n)$ running time for every function. This is, of course, a crude comparison, because the operations involved in each function are different. Regardless, a constant running time is a considerable improvement, especially in large networks. With respect to storage, our protocol does not require a user to store anything permanently since a session is constructed on the fly.

Table 1. The table shows the running time for each function as a function of n, where n is the number of users in the secure group.

Function	Running Time
Setting up trapdoor	$\Theta(n)$
Encrypting a group message	$\Theta(1)$
Decrypting a group message	$\Theta(1)$
Generating a session key	$\Theta(1)$
Adding a node	$\Theta(1)$
Removing a node	$\Theta(n)$

In this work, we only presented the basic building blocks of the protocol, without regard to how it can actually be implemented efficiently. One issue with lattice-based cryptographic constructions in general is the size of the security parameters. For a reasonable security level, such as a 128-bit security level, the dimension of the lattice should be greater than 500 [26]. This means that each device would need to store at least 2 matrices, each in the order of 100s of kilobytes. This could be an issue for extremely small devices with highly constrained memory sizes. This problem can be partially solved by using the Hermite normal form (HNF) of the matrix $\mathbf{A}$ that describes the lattice $\mathcal{L}_{\mathcal{G}}$ [25]. This optimization reduces the size of the matrix by a factor of n, where n is the dimension of the lattice. A further improvement both in terms of storage and computation can be achieved by defining the one-way function on the ring-LWE problem, which is more compact and allows for faster computations using fast Fourier transform (FFT) techniques [44]. One can then use Micciancio's compact one-way function, which is syntactically similar to the one presented here [45].

8. Conclusions and Direction for Future Work

In this paper, we have presented a new cryptographic group-key-management protocol based on lattice trapdoors. The protocol enables a group of IoT devices to define a trapdoor-based one-way function that only a device in the network can revert. We have shown how one can construct a cryptographic group-key from the one-way function.

The results show that the protocol can be implemented on resource-constrained IoT devices. Moreover, the fact that most of the functions of the protocol can be effected in constant time is an important improvement on similar protocols that attempt to solve the same problem.

The advantage of the protocol is twofold: First, it is highly lightweight since the computations involved are simple linear-algebra operations that are parallelizable and evaluations of simple linear functions. Second, it provides conjectured security against potential quantum-based attacks and has security based on worst-case hardness assumptions. We have also demonstrated how to add a user to a secure group dynamically, while

maintaining the necessary security requirements, all in constant time, regardless of the size of the network. The protocol can be potentially deployed on various IoT networks, and help secure such networks against future quantum-based attacks.

The protocol can be improved in future work by expanding the security model to include an active adversary, devising a parallel bad/good lattice basis-generation mechanism, and optimizing the implementation.

Author Contributions: The conceptualisation and design of the protocol and writing of the article was done by T.G. and M.G. supervised the conceptualisation and writing process. G.P.H. reviewed and validated the work. U.J. assisted during initial phases of the project by providing guidance and feedback. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by the Research Grants Council of Hong Kong under project CityU 11218419 and partially by the Knowledge Foundation in the project Next generation Industrial IoT (NIIT) at Mid Sweden University.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Stankovic, J.A. Research directions for the internet of things. *IEEE Internet Things J.* **2014**, *1*, 3–9. [CrossRef]
2. Zhang, Z.K.; Cho, M.C.Y.; Wang, C.W.; Hsu, C.W.; Chen, C.K.; Shieh, S. IoT security: Ongoing challenges and research opportunities. In Proceedings of the 2014 IEEE 7th International Conference on Service-Oriented Computing and Applications, Matsue, Japan, 17–19 November 2014; pp. 230–234.
3. Porambage, P.; Braeken, A.; Schmitt, C.; Gurtov, A.; Ylianttila, M.; Stiller, B. Group key establishment for enabling secure multicast communication in wireless sensor networks deployed for IoT applications. *IEEE Access* **2015**, *3*, 1503–1511. [CrossRef]
4. Tiloca, M.; Nikitin, K.; Raza, S. Axiom: DTLS-based secure IoT group communication. *ACM Trans. Embed. Comput. Syst.* **2017**, *16*, 1–29. [CrossRef]
5. Gebremichael, T.; Jennehag, U.; Gidlund, M. Lightweight iot group key establishment scheme using one-way accumulator. In Proceedings of the 2018 International Symposium on Networks, Computers and Communications (ISNCC), Rome, Italy, 19–21 June 2018; pp. 1–7.
6. Ferrari, N.; Gebremichael, T.; Jennehag, U.; Gidlund, M. Lightweight group-key establishment protocol for IoT devices: Implementation and performance Analyses. In Proceedings of the 2018 Fifth International Conference on Internet of Things: Systems, Management and Security, Valencia, Spain, 15–18 October 2018; pp. 31–37.
7. Gisin, N.; Ribordy, G.; Tittel, W.; Zbinden, H. Quantum cryptography. *Rev. Mod. Phys.* **2002**, *74*, 145. [CrossRef]
8. Bernstein, D.J. Introduction to post-quantum cryptography. In *Post-Quantum Cryptography*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 1–14.
9. Micciancio, D.; Regev, O. Lattice-based cryptography. In *Post-Quantum Cryptography*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 147–191.
10. Gentry, C. Toward basing fully homomorphic encryption on worst-case hardness. In Proceedings of the Annual Cryptology Conference, Santa Barbara, CA, USA, 15–19 August 2010; pp. 116–137.
11. Shor, P.W. Algorithms for quantum computation: Discrete logarithms and factoring. In Proceedings of the 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM, USA, 20–22 November 1994; pp. 124–134.
12. Regev, O. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM* **2009**, *56*, 1–40. [CrossRef]
13. Peikert, C. A decade of lattice cryptography. *Found. Trends Theor. Comput. Sci.* **2016**, *10*, 283–424. [CrossRef]
14. Raza, S.; Seitz, L.; Sitenkov, D.; Selander, G. S3K: Scalable security with symmetric keys—DTLS key establishment for the Internet of Things. *IEEE Trans. Autom. Sci. Eng.* **2016**, *13*, 1270–1280. [CrossRef]
15. Halford, T.R.; Courtade, T.A.; Chugg, K.M. Energy-efficient, secure group key agreement for ad hoc networks. In Proceedings of the 2013 IEEE Conference on Communications and Network Security (CNS), Washington, DC, USA, 14–16 October 2013; pp. 181–188.
16. Lei, X.; Liao, X. NTRU-KE: A Lattice-based Public Key Exchange Protocol. *Cryptol. ePrint Arch.* **2013**, *2013*, 718.
17. Diffie, W.; Hellman, M. New directions in cryptography. *IEEE Trans. Inf. Theory* **1976**, *22*, 644–654. [CrossRef]
18. Banerjee, U.; Chandrakasan, A.P. Efficient Post-Quantum TLS Handshakes using Identity-Based Key Exchange from Lattices. In Proceedings of the ICC 2020-2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020; pp. 1–6.

19. Arul, R.; Raja, G.; Almagrabi, A.O.; Alkatheiri, M.S.; Chauhdary, S.H.; Bashir, A.K. A quantum-safe key hierarchy and dynamic security association for LTE/SAE in 5G scenario. *IEEE Trans. Ind. Inform.* **2019**, *16*, 681–690. [CrossRef]
20. Murugan, G. An Efficient Algorithm on Quantum Computing With Quantum Key Distribution for Secure Communication. *Int. J. Commun.* **2020**, *5*.
21. Banupriya, S.; Kottursamy, K.; Bashir, A.K. Privacy-preserving hierarchical deterministic key generation based on a lattice of rings in public blockchain. *Peer-to-Peer Netw. Appl.* **2021**, *14*, 2813–2825. [CrossRef]
22. Yi, H. Secure Social Internet of Things Based on Post-Quantum Blockchain. *IEEE Trans. Netw. Sci. Eng.* **2021**, *9*, 950–957. [CrossRef]
23. Regev, O. The learning with errors problem. *Invit. Surv.* **2010**, *7*, 30.
24. Micciancio, D.; Goldwasser, S. *Complexity of Lattice Problems: A Cryptographic Perspective*; Springer: Berlin/Heidelberg, Germany, 2012; Volume 671.
25. Alwen, J.; Peikert, C. Generating shorter bases for hard random lattices. *Theory Comput. Syst.* **2009**, *48*, 535–553. [CrossRef]
26. Hoffstein, J.; Pipher, J.; Silverman, J.H. NTRU: A ring-based public key cryptosystem. In *International Algorithmic Number Theory Symposium*; Springer: Berlin/Heidelberg, Germany, 1998; pp. 267–288.
27. Peikert, C. An efficient and parallel Gaussian sampler for lattices. In Proceedings of the Annual Cryptology Conference, Santa Barbara, CA, USA, 15–19 August 2010; pp. 80–97.
28. Naoui, S.; Elhdhili, M.E.; Saidane, L.A. Security analysis of existing IoT key management protocols. In Proceedings of the 2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA), Agadir, Morocco, 29 November–2 December 2016; pp. 1–7.
29. Lee, E.J.; Lee, S.E.; Yoo, K.Y. A certificateless authenticated group key agreement protocol providing forward secrecy. In Proceedings of the 2008 International Symposium on Ubiquitous Multimedia Computing, Hobart, Australia, 13–15 October 2008; pp. 124–129.
30. Micciancio, D.; Peikert, C. Trapdoors for lattices: Simpler, tighter, faster, smaller. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, 15–19 April 2012; pp. 700–718.
31. Ajtai, M. Generating hard instances of the short basis problem. In *International Colloquium on Automata, Languages, and Programming*; Springer: Berlin/Heidelberg, Germany, 1999; pp. 1–9.
32. Ajtai, M. Representing hard lattices with O (n log n) bits. In Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, 22–24 May 2005; pp. 94–103.
33. Peikert, C.; Vaikuntanathan, V.; Waters, B. A framework for efficient and composable oblivious transfer. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 17–21 August 2008; pp. 554–571.
34. Tian, Z.; Qiao, S. A complexity analysis of a Jacobi method for lattice basis reduction. In Proceedings of the Fifth International C* Conference on Computer Science and Software Engineering, Montreal, QC, Canada, 27–29 June 2012; pp. 53–60.
35. MacKay, D.J.; Neal, R.M. Near Shannon limit performance of low density parity check codes. *Electron. Lett.* **1996**, *32*, 1645–1646. [CrossRef]
36. Balenson, D.; McGrew, D.; Sherman, A. Key management for large dynamic groups: One-way function trees and amortized initialization. *Mar* **1999**, *15*, 1–14.
37. Ghanem, S.M.; Abdel-Wahab, H. A secure group key management framework: Design and rekey issues. In Proceedings of the Eighth IEEE Symposium on Computers and Communications, ISCC 2003, Antalya, Turkey, 30 June–3 July 2003; pp. 797–802.
38. Di Pietro, R.; Mancini, L.V.; Jajodia, S. Providing secrecy in key management protocols for large wireless sensors networks. *Ad Hoc Netw.* **2003**, *1*, 455–468. [CrossRef]
39. Gentry, C.; Peikert, C.; Vaikuntanathan, V. Trapdoors for hard lattices and new cryptographic constructions. In Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing, Victoria, BC, Canada, 17–20 May 2008; pp. 197–206.
40. Peikert, C. Public-key cryptosystems from the worst-case shortest vector problem. In Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, Bethesda, MD, USA, 31 May–2 June 2009; pp. 333–342.
41. Dunkels, A.; Gronvall, B.; Voigt, T. Contiki-a lightweight and flexible operating system for tiny networked sensors. In Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks, Tampa, FL, USA, 16–18 November 2004; pp. 455–462.
42. Nath, O. Review on raspberry pi 3b+ and its scope. *Int. J. Eng. Appl. Sci. Technol.* **2020**, *4*, 157–159. [CrossRef]
43. Peikert, C. Lattice cryptography for the internet. In *International Workshop on Post-Quantum Cryptography*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 197–219.
44. Lyubashevsky, V.; Peikert, C.; Regev, O. A toolkit for ring-LWE cryptography. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, 26–30 May 2013; pp. 35–54.
45. Micciancio, D. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. *Comput. Complex.* **2007**, *16*, 365–411. [CrossRef]