



Article

# A Lattice-Based Group Authentication Scheme

Jheng-Jia Huang <sup>1</sup> , Yi-Fan Tseng <sup>1</sup>, Qi-Liang Yang <sup>1</sup> and Chun-I Fan <sup>1,2,\*</sup> 

<sup>1</sup> Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung 804, Taiwan; jhengjia.huang@gmail.com (J.-J.H.); yftseng1989@gmail.com (Y.-F.T.); s11115263@stu.edu.tw (Q.-L.Y.)

<sup>2</sup> Intelligent Electronic Commerce Research Center, National Sun Yat-sen University, Kaohsiung 804, Taiwan

\* Correspondence: cifan@mail.cse.nsysu.edu.tw; Tel.: +886-7-525-4346

Received: 19 May 2018; Accepted: 11 June 2018; Published: 15 June 2018



**Abstract:** Authentication has been adopted in many areas, but most of these authentication schemes are built using traditional cryptographic primitives. It is widely believed that such primitives are not resistant to quantum algorithms. To deal with those quantum attacks, lattice-based cryptography was introduced by Ajtai in 1996. To the best of our knowledge, the existing lattice-based authentication schemes are based on a lattice-based public key encryption called NTRU: a ring-based public key cryptosystem, proposed by Hoffstein, Pipher, and Silverman in 1998. However, these schemes only support the case of a single user. In view of the aforementioned issue, we propose the first lattice-based group authentication scheme. The proposed scheme is secure against replay attacks and man-in-the-middle attacks. Moreover, compared with the existing lattice-based authentication schemes, ours provides the most efficient method to agree upon a session key among a group of users after mutual authentication.

**Keywords:** group authentication; lattice-based cryptography; quantum attacks; information security

## 1. Introduction

Nowadays, authentication has been adopted in many areas, such as radio frequency identification (RFID), cloud computing, wireless sensor networks, internet of things (IoT), etc. Authentication schemes can be separated into two types: one is individual-oriented authentication, and another is group-oriented authentication. The scenario of individual-oriented authentication is one-to-one communication. There are only two users needing to authenticate each other. In group-oriented authentication, there are more than two users in a group. They build a private network and share messages among the group. Such a cryptographic primitive can be applied heavily to many-to-many network environments, e.g., Internet of Things or RFID.

However, most of these authentication schemes are built on traditional cryptographic primitives, e.g., RSA and ElGamal, where the cryptosystems are constructed based on discrete logarithm or factorization. It is widely believed that such primitives are not resistant to quantum algorithms. For instance, the algorithm proposed by Shor [1] in 1994 is a quantum algorithm that solves discrete logarithm problems and factorization problems in subexponential time complexity. To deal with those quantum attacks, lattice-based cryptography was introduced by Ajtai [2] in 1996. A lattice can be represented as a matrix which has a periodic structure in dimensional space. There are two central hard problems in lattice-based cryptography: the shortest vector problem (SVP) and the closest vector problem (CVP). The shortest vector problem (SVP) is, given a base of a lattice, to find the smallest possible nonzero vector of the lattice. The closest vector problem (CVP) is, given a base of a lattice and a vector which does not belong to the lattice, to find a vector belonging to the lattice that is the closest vector to the given vector. Both SVP and CVP are believed to be invulnerable to quantum attacks such as Shor's algorithm. As a powerful and promising quantum-resistant primitive, it has been adopted in lots of applications, such as public key cryptosystems and sieve algorithms.

To the best of our knowledge, the existing lattice-based authentication schemes [3,4] are based on a lattice-based public key encryption called NTRU encryption [5], proposed by Hoffstein, Pipher, and Silverman in 1998. However, these two schemes support only the case of a single user. Another intuitive way to achieve group authentication is by using a lattice-based group signature. Nevertheless, a signature scheme is an asymmetric cryptographic primitive, which is usually more costly than a symmetric one. Besides this, group signatures support additional but unnecessary properties, such as the anonymity of signers, which may not be a desired property in RFID or IoT. Therefore, in view of the aforementioned issues, we will propose a new construction for lattice-based authentication. The proposed scheme also supports group authentication.

*Contributions*

In this manuscript, we propose a lattice-based authentication scheme supporting group authentication. Compared with the existing lattice-based authentication schemes, our scheme provides the most efficient authentication protocol in terms of the total cost to generate a session key among a group of users after mutual authentication.

**2. Preliminaries**

In this section, we review some preliminaries, including the definition of lattices, two standard worst-case approximation problems on lattices, the SampleD algorithm, and the BasisDel algorithm.

*2.1. Notation*

We denote the set of integers modulo  $q$  by  $\mathbb{Z}_q$ . Column vectors are represented by lower-case bold letters and matrices by upper-case bold letters. For a matrix  $S \in \mathbb{R}^{m_1 \times m_2}$ , we say the norm of  $S$  is  $\|S\| = \max_{1 \leq i \leq m_2} \|s_i\|$ , where  $\|s_i\|$  denotes the  $\ell_2$ -norm (Euclidean norm) of the column vector  $s_i$ . We let  $\tilde{S} \in \mathbb{R}^{m_1 \times m_2}$  denote the matrix whose columns  $\tilde{s}_1, \dots, \tilde{s}_{m_2}$  represent the Gram–Schmidt orthogonalization of the vectors  $s_1, \dots, s_{m_2}$  taken in the same order. Let  $\|\tilde{S}\|$  denote the Gram–Schmidt norm of  $S$ . Let  $f(n)$  and  $g(n)$  denote two positive real-valued functions. We say that  $f = O(g)$  if there exist two constants  $c_1, c_2$  such that  $f(n) < c_1 \cdot g(n)$  for all  $n \geq c_2$ ;  $f = \Omega(g)$  if  $g = O(f)$ ; and  $f = \Theta(g)$  if  $f = O(g)$  and  $g = O(f)$ . We say  $f = \tilde{O}(g)$  if  $f = O(g \cdot \text{poly}(\log g))$  and  $f = \tilde{\Theta}(g)$  if  $f = \Theta(g \cdot \text{poly}(\log g))$ .

*2.2. Lattices*

Based on [6], the definitions are shown as following:

**Definition 1.** Let  $\mathbb{R}^m$  be the  $m$ -dimensional Euclidean space. A lattice  $\Lambda \subseteq \mathbb{R}^m$  is a set

$$\Lambda = \left\{ \sum_{i=1}^k c_i \mathbf{b}_i \mid c_i \in \mathbb{Z} \text{ and } \mathbf{b}_1, \dots, \mathbf{b}_k \in \mathbb{R}^m \right\}$$

of all integral combinations of  $k$  linearly independent vectors  $\mathbf{b}_1, \dots, \mathbf{b}_k$  in  $\mathbb{R}^m$  ( $m > k$ ). The integers  $k$  and  $m$  are called the rank and dimension of the lattice, respectively. The vector set  $\{\mathbf{b}_1, \dots, \mathbf{b}_k\}$  is called a basis of the lattice.

**Definition 2.** Let  $A$  ( $A \in \mathbb{Z}^{m \times n}$ ) be a basis. The integers  $m$  and  $n$  are called the dimension and rank;  $\mathbf{y}$  and  $\mathbf{t}$  are column vectors; and  $q$  is a prime number.

$$\Lambda(A) = \{ \mathbf{y} \in \mathbb{Z}^m \mid \mathbf{y} = A\mathbf{t} \text{ mod } q, \mathbf{t} \in \mathbb{Z}^n \}$$

$$\Lambda^\perp(A) = \{ \mathbf{y} \in \mathbb{Z}^m \mid A^\top \mathbf{y} = 0 \text{ mod } q \}$$

**Definition 3** (TrapGen( $1^\lambda$ ) [7,8]). There is a probabilistic polynomial-time (PPT) algorithm that, on input of a security parameter  $1^\lambda$ , an odd prime  $q = \text{poly}(\lambda)$ , and two integers  $n = \Theta(\lambda)$  and  $m \geq 6n \log q$ , outputs a matrix  $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$  statistically close to uniform, and a basis  $\mathbf{T}_A$  for  $\Lambda^\perp(\mathbf{A})$  with overwhelming probability such that  $\|\tilde{\mathbf{T}}_A\| \leq \tilde{\Theta}(\sqrt{m})$ .

**Definition 4** (The Shortest Vector Problem (SVP) [3]). Given a base  $\mathbf{B}$  (a set of vectors linearly independent) of a lattice  $L$ , find the smallest possible nonzero vector of  $L$ .

**Definition 5** (The Closest Vector Problem (CVP) [3]). Given a base  $\mathbf{B}$  of a lattice  $L$  and a vector  $\mathbf{v} \notin L$ , find a vector  $v \in L$  that is the closest to  $\mathbf{v}$ .

### 2.3. The Gaussian Sampling Algorithm: SampleD( $\mathbf{B}, s, c, \mathbf{t}$ )

Based on [9], the definitions are shown as following:

**Definition 6.** There is a PPT algorithm that, given a basis  $\mathbf{B}$  of an  $m$ -dimensional lattice  $\Lambda$ , a parameter  $s \geq \|\tilde{\mathbf{B}}\| \cdot \omega(\sqrt{\log m})$ , and a center  $\mathbf{c} \in \mathbb{R}^m$ , outputs a sample from a distribution that is statistically close to  $D_{\Lambda, s, \mathbf{c}}$ , where  $D_{\Lambda, s, \mathbf{c}}$  is a discrete Gaussian distribution with Gaussian parameter  $s$  and center  $\mathbf{c}$ .

**Definition 7.** The Gaussian Sampling Algorithm: SampleD( $\mathbf{B}, s, c, \mathbf{t}$ ) ( $\mathbf{B}$  is a trapdoor basis of the lattice.  $\mathbf{A}$  is also a basis of the lattice but not a trapdoor).

- **Input:**

1. A basis  $\mathbf{B}$  of a lattice  $\Lambda \in \mathbb{R}^m$ ;
2. A positive real parameter  $s \geq \|\tilde{\mathbf{B}}\| \cdot \omega(\sqrt{\log m})$ ;
3. A center vector  $\mathbf{c} \in \mathbb{R}^n$ ;
4. A vector  $\mathbf{t} \in \mathbb{Z}_q^n$ .

- **Output:**

A fresh random lattice vector  $x \in \Lambda$  drawn from a distribution statistically close to  $D_{\Lambda, s, c}$ , such that  $\mathbf{A}x = \mathbf{t} \pmod q$ .

**Definition 8** ([10]). SampleD is said to be one-way if, given  $(s, c, \mathbf{t})$ , there is no polynomial-time adversary that outputs  $x$  such that  $x \in D_{\Lambda, s, c}$  and  $\mathbf{A}x = \mathbf{t} \pmod q$  (where  $\mathbf{A}$  is a basis).

### 2.4. The Basis Delegation Algorithm: BasisDel( $\mathbf{T}_A, \mathbf{A}, \bar{\mathbf{A}}$ )

**Definition 9** ([11]). The Basis Delegation Algorithm: BasisDel( $\mathbf{T}_A, \mathbf{A}, \bar{\mathbf{A}}$ )

- **Input:**

1. An arbitrary  $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$  such that  $\mathbf{A}$  is primitive;
2. An arbitrary basis  $\mathbf{T}_A$  of  $\Lambda^\perp(\mathbf{A})$ ;
3. An arbitrary  $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$ .

- **Output:**

A basis  $\mathbf{T}_{A'}$  of  $\Lambda^\perp(\mathbf{A}' = \mathbf{A} \parallel \bar{\mathbf{A}}) \in \mathbb{Z}^{m+\bar{m}}$  such that  $\|\tilde{\mathbf{T}}_{A'}\| = \|\tilde{\mathbf{T}}_A\|$ .

### 3. Our Construction

In this section, we present a lattice-based authentication scheme. The proposed scheme consists of four phases: Setup, Registration, Group Joining, and Authentication. The notation used in the proposed scheme is defined in Table 1.

Table 1. The notation.

Notations	Meaning
$\ \tilde{B}\ $	The Gram–Schmidt norm
$1^\lambda$	Security parameter
$n$	An integer
$q$	Prime $q = poly(n)$
$m$	Dimension $m \geq 6n \log q$
$m'$	$m' = poly(\lambda) \in \mathbb{N}$
$L$	$O(\sqrt{m})$
$s$	Gaussian parameter $s \geq L \cdot \omega(\sqrt{\log(m + m')})$
$M$	A message set, $M = \{B_1, \dots, B_Q \in \mathbb{Z}^{n \times m'}\}$
$Q$	$Q = poly(\lambda)$
$Q'$	$Q' \in [Q]$
$A$	Public key
$T_A$	Secret key
$B_i$	Exclusive matrix of $user_i$
$H(\cdot)$	Convert a vector into a key of symmetrical encryption

#### 3.1. The Proposed Scheme

##### 3.1.1. Setup

Key generation center (KGC) performs the following operations [2]:

1. Choose a security parameter  $1^\lambda$  ( $\lambda \in \mathbb{N}$ ).
2. Choose integers  $n$  and  $q$  (a prime),  $q = poly(n)$ .
3. Choose dimension  $m \geq 6n \log q$  and a bound  $L = O(\sqrt{m})$ .
4. Choose a Gaussian parameter  $s \geq L \cdot \omega(\sqrt{\log(m + m')})$ , where  $m' = poly(\lambda) \in \mathbb{N}$ .
5. Choose a set  $M = \{B_1, \dots, B_Q \in \mathbb{Z}^{n \times m'}\}$ , where  $Q = poly(\lambda)$  and  $B_i$  is independently chosen with uniform distribution. Note that  $B_i$  is the public parameter for user  $i$ .
6. Let  $H(\cdot)$  denote the function which converts a vector into a key of symmetric encryption.
7. Let  $E_k$  denote the symmetric encryption.
8. Publish system parameters chosen as above.

##### 3.1.2. Registration

KGC performs the algorithm  $\text{TrapGen}(1^\lambda)$  [7,8] to generate  $(A, T_A)$  (where  $A \in \mathbb{Z}^{n \times m}$  and  $T_A$  is a short basis of  $\Lambda^\perp(A)$ ).

##### 3.1.3. Group Joining

Let group  $X = \{B_1, \dots, B_{Q'}\}$  for some  $Q' \in [Q]$  ( $X \subset M$ ).

1. A  $user_i$  sends  $B_i$  via a security channel to a group manager.
2. After receiving  $B_i$ , the manager computes  $Acc_x = [\sum_{B_i \in X} B_i] \in \mathbb{Z}^{n \times m'}$ ,  $F_{B_i} = [A \parallel \sum_{1 \leq j(\neq i) \leq Q'} B_j] \in \mathbb{Z}^{n \times (m+m')}$ , and  $\omega_{B_i} = T_{F_{B_i}} = \text{BasisDel}(T_A, A, \sum_{1 \leq j(\neq i) \leq Q'} B_j)$ .
3. Then, the manager sends  $\omega_{B_i}$  via a secure channel to  $user_i$ .

### 3.1.4. Authentication

1. First, the manager sends  $t_i \in \mathbb{Z}_q^n$  to  $user_i$ .
2. After receiving  $t_i$ ,  $user_i$  computes  $d_{B_i} = \text{SampleD}(\omega_{B_i}, s, 0, t_i)$ , and then chooses a number  $r_i$  randomly, and computes  $k_i = H(\omega_{B_i})$ ,  $a_i = E_{k_i}(d_{B_i}, r_i)$ .
3. The  $user_i$  sends  $a_i$  to the manager.
4. After receiving  $a_i$  from each user, the manager performs the following: for  $i = 1$  to  $Q'$ , the manager computes  $(d_{B_i}, r_i) = D_{H(\omega_{B_i})}(a_i)$  and  $F_{B_i} = [A || (Acc_x - B_i)] \in \mathbb{Z}^{n \times (m+m')}$ , then checks if  $F_{B_i} \cdot d_{B_i} = t_i \pmod q$ . If the check passes, the user is authenticated; otherwise, the manager aborts the session.
5. Next, the manager performs as follows according to  $Q'$ .

#### Case 1 ( $Q' > 1$ ):

1. First, the manager computes  $R_i = r_1 \oplus r_2 \oplus \dots \oplus r_{i-1} \oplus r_{i+1} \dots \oplus r_{Q'}$  for  $i = 1$  to  $Q'$ ,  $sk = r_1 \oplus r_2 \oplus \dots \oplus r_{Q'}$ ,  $k_i = H(\omega_{B_i})$ , and  $b_i = E_{k_i}(sk, R_i)$ .
2. The manager sends  $b_i$  to each  $user_i$ .
3. After receiving  $b_i$ ,  $user_i$  computes  $(sk', R_i') = D_{k_i}(b_i)$ , and then checks if  $sk' = R_i' \oplus r_i$ . If it is true, the manager is authenticated. Then,  $user_i$  sets the session key  $sk = sk'$ .

#### Case 2 ( $Q' = 1$ ):

1. First, the manager chooses a number  $r'$  randomly.
2. Then, the manager computes  $k_i = H(\omega_{B_i})$ ,  $b = E_{k_i}(r', H(r_i))$  and the manager sends  $b$  to  $user_i$ .
3. After receiving  $b$ ,  $user_i$  computes  $(r'', h) = D_{k_i}(b)$ . The  $user_i$  checks if  $h = H(r_i)$ . If it is true, the manager is authenticated. Then,  $user_i$  sets the session key  $sk = r'' \oplus r_i$ .

## 4. Security Analysis

- In this section, we provide the security analyses, which include the analyses on the replay attacks, the man-in-the-middle attacks, and the secure mutual authentication, where the detailed security proofs are shown in Appendix A. **Replay Attacks:**

In the proposed scheme, each user chooses a number  $r_i$  randomly and sends  $r_i$  to the manager. The random number  $r_i$  is different in every authentication round. Hence, there is no chance that an attacker can succeed in replay attacks in the proposed scheme.

- **Man-in-the-Middle Attacks:**

In the proposed scheme, we use a secure symmetric encryption algorithm to encrypt data in every data flow of the authentication phase. If there is an attacker wanting to intercept messages  $a_i$  or  $b$ , there is no chance that the attacker can obtain the plaintext without the symmetric key. In Steps 1–4 of the authentication phase,  $d_{B_i}$  and  $r_i$  are protected by the secret key  $k_i$ , and in Step 5 of the authentication phase,  $sk$ ,  $R_i$ ,  $r'$ , and  $H(r_i)$  are protected by the secret key  $k_i$ , too.

The comparisons of features and security between the proposed scheme and [3,4] are summarized in Table 2.

**Table 2.** Feature and security comparisons.

	Moustaine et al.'s Scheme	Park et al.'s Scheme	Our Scheme
Mutual Authentication	Yes	Yes	Yes
Group Authentication	No	No	Yes
No Replay Attack	Yes	Yes	Yes
No Man-in-the-Middle Attack	Yes	Yes	Yes

- Secure Mutual Authentication:

In the Group Joining phase, a group manager will share a long-term secret  $\omega_{B_i}$  with user  $i$ , which is also a trapdoor to function SampleD. In the Authentication phase, the group manager first sends  $t_i$  to user  $i$  as a challenge, then user  $i$  returns with  $a_i = E_{k_i}(d_{B_i}, r_i)$ , where  $k_i = H(\omega_{B_i})$ . Note that it is necessary to compute  $d_{B_i}$  using SampleD with  $\omega_{B_i}$ . Therefore, if the check in Step 4 passes, user  $i$  is authenticated. On the other hand, if the check in Case 1 (Case 2) passes, then we can assure that the group manager decrypts the ciphertext  $a_i$  with  $k_i = H(\omega_{B_i})$  to obtain the correct  $r_i$  for user  $i$ . By the security of the underlying symmetric encryption scheme, we are sure that the group manager is authenticated. If the two entities are both authenticated by each other, then we know that the session key  $sk$  can be securely established by the security of SampleD and the underlying symmetric encryption scheme.

## 5. Performance Comparisons

In this section, our proposed scheme is compared with some existing lattice-based authentication schemes [3,4]. Tables 3 and 4 summarize the comparisons between the proposed scheme and those schemes in performance.

In the authentication phase of the proposed scheme, when there is only one user ( $Q' = 1$ ), the manager uses one symmetric encryption operation, one symmetric decryption operation, and  $n \times (m + m')$  module multiplications. The user performs one SampleD operation, two hashing operations, one symmetric encryption operation, one symmetric decryption operation, and one exclusive-or operation. We adopt AES 256 as the symmetric encryption operation. The cost of AES 256 is 15.875 ms [12]. The cost of SampleD is 0.00045 ms [13]. We take SHA-3 256 bits as our hashing operation. According to [14], the time taken to hash a 100-byte message is 0.001295 ms. We use the Montgomery algorithm with 256 bits as our modular multiplication. The time of a modular multiplication is 0.137 ms [15]. According to [16], we set  $n \times (m + m') = 192 \times 417 = 80,064$ . Hence, the computation time of  $n \times (m + m')$  module multiplications is  $0.137 \text{ ms} \times 80,064 = 10,968.77 \text{ ms}$ . In conclusion, the total computation cost of the manager is  $15.875 \times 2 + 10,968.77 \approx 11,000.52 \text{ ms}$ . The total cost of the user is  $0.00045 + 0.001295 \times 2 + 15.875 \times 2 \approx 31.753 \text{ ms}$ . When there is more than one user ( $Q' > 1$ ), the manager uses one symmetric encryption operation, one symmetric decryption operation, one hashing operation, and  $n \times (m + m')$  module multiplications. Each user uses one SampleD operation, one hashing operation, one symmetric encryption operation, and one symmetric decryption operation. The total computation cost of the manager is  $15.875 \times 2 + 0.001295 + 22,446.08 \approx 22,477.83 \text{ ms}$ . The total computation cost of each user is  $\approx 31.753 \text{ ms}$ .

In the authentication phase of Moustaine et al.'s Scheme [3], the reader/back-end executes one NTRU decryption operation, two rotation operations, two exclusive-or operations, one hashing operation, one subtraction, and  $2N^2$  multiplications. The tag performs four addition operations, three rotation operations, one hashing operation, and two exclusive-or operations. According to [17],  $HMAC(K, m) = H((K' \oplus opad) || H((k' \oplus ipad) || m)) \approx 2 \times \text{SHA-3 hashing operations}$ . Thus, the running time of HMAC is  $0.001295 \times 2 = 0.00259 \text{ ms}$ . According to [3], the cost of an NTRU decryption is  $2N^2$  modular multiplications ( $N = 167$ ). Hence, the cost of an NTRU decryption is  $2 \times 167 \times 167 \times 0.137 \approx 7641.586 \text{ ms}$ . In summary, the cost of the reader/back-end is  $7641.586 \times 2 + 0.00259 \approx 15,283.1746 \text{ ms}$ . The cost of the tag is 0.00259 ms.

In the authentication phase of Park et al.'s Scheme [4], the user performs  $2N^2$  multiplication operations, one hashing operation, two rotation operations, one exclusive-or operation, and two addition operations. The bank executes  $N^2$  multiplication operations, one exclusive-or operation, one rotation operation, and one hashing operation. In conclusion, the cost of the bank is  $167 \times 167 \times 0.137 + 0.001295 \approx 3820.79 \text{ ms}$ . The cost of the user is  $2 \times 167 \times 167 \times 0.137 + 0.001295 \approx 7641.59 \text{ ms}$ .

Note that for a single-user situation, the computation cost of the server side is 28% faster than [3], and the computation cost of the user side is 99% faster than [4]. As for the computation cost of the



group authentication scenario, we can see that the total cost of the server side and the user side is less than those of [3,4].

**Table 3.** The cost comparison of the single-user situation between [3,4] and our scheme.

	Server (Manager/Reader or Back-End/Bank)	User (Tag)
Our Scheme	11,000.52 ms	31.75 ms
Moustaine et al.'s Scheme	15,283.17 ms	0.00259 ms
Park et al.'s Scheme	3820.79 ms	7641.59 ms

**Table 4.** The cost comparison of a  $Q' + 1$  users ( $Q' + 1$ ) situation between [3,4] and our scheme.

	Group Manager	Group Member	Total Cost
Our Scheme	11,000.52 $Q'$ ms	31.75 ms	11,032.27 $Q'$ ms
Moustaine et al.'s Scheme	15,283.17 $Q'$ ms	0.00259 ms	15,283.17 $Q'$ ms
Park et al.'s Scheme	3820.79 $Q'$ ms	7641.59 ms	11,462.38 $Q'$ ms

## 6. Conclusions

Most authentication schemes are built on traditional cryptographic primitives. It is widely believed that traditional cryptographic primitives are not resistant to quantum attacks. In 1994, Shor proposed a quantum algorithm that can solve discrete logarithm problems and factorization problems. To deal with those quantum attacks, Ajtai presented lattice-based cryptography in 1996. To the best of our knowledge, there is no lattice-based group mutual authentication scheme which is based on symmetric encryption. There are two lattice-based authentication schemes which use the NTRU asymmetric encryption algorithm in the literature. Usually, the cost of an asymmetric encryption is higher than that of a symmetric encryption. In this manuscript, we have proposed a lattice-based authentication scheme based on symmetric encryption. Additionally, not only does the proposed scheme provide group authentication, but also it resists replay attacks and man-in-the-middle attacks. In the proposed scheme, after making sure of the total number of users in a group, the manager can choose any of them to create an authentication procedure for any subgroups. Besides this, we give a cost comparison between our scheme and the other two NTRU-based mutual authentication schemes in terms of authentication and establishing a session key. Our scheme provides the lowest cost in generating a session key among a group of users after mutual authentication.

**Author Contributions:** All of the authors worked collaboratively in the design of the scheme, proofs of its security, and analyses of its performance.

**Funding:** This research received no external funding.

**Acknowledgments:** This work was partially supported by the Ministry of Science and Technology of Taiwan under grants MOST 105-2923-E-110-001-MY3, MOST 105-2221-E-110-053-MY2, MOST 106-3114-E-110-001, and financially supported by the Intelligent Electronic Commerce Research Center from The Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education (MOE) in Taiwan.

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

## Appendix A.

### Appendix A.1. Security Proofs

In this section, there is no security model suited for the proposed scheme, so we give the security models at first. Then, we prove that the proposed scheme achieves mutual authentication security and session key security.

Appendix A.1.1. Security Models

**Definition A1** (Group Mutual Authentication). *An attacker interacts with a simulator in the following game (Figure A1).*

**Setup.** *The KGC publishes the system parameters to the attacker.*

**Training Phase.** *The attacker is allowed to query the following oracles:*

- *Registration oracle: The outputs of this oracle are the public key  $A$  and the secret key  $T_A$ .*
- *Group-joining oracle: The input of this oracle is  $B_i$  and the output is  $w_{B_i}$ .*
- *Authentication oracle: The inputs of the oracle are a group manager name, the group name, and the user of the group. The outputs of the oracle are  $t_i$ ,  $a_i$ , and  $b_i$ .*

**Challenge.** *There are two parts of this phase, depending on the entity that the attacker impersonates.*

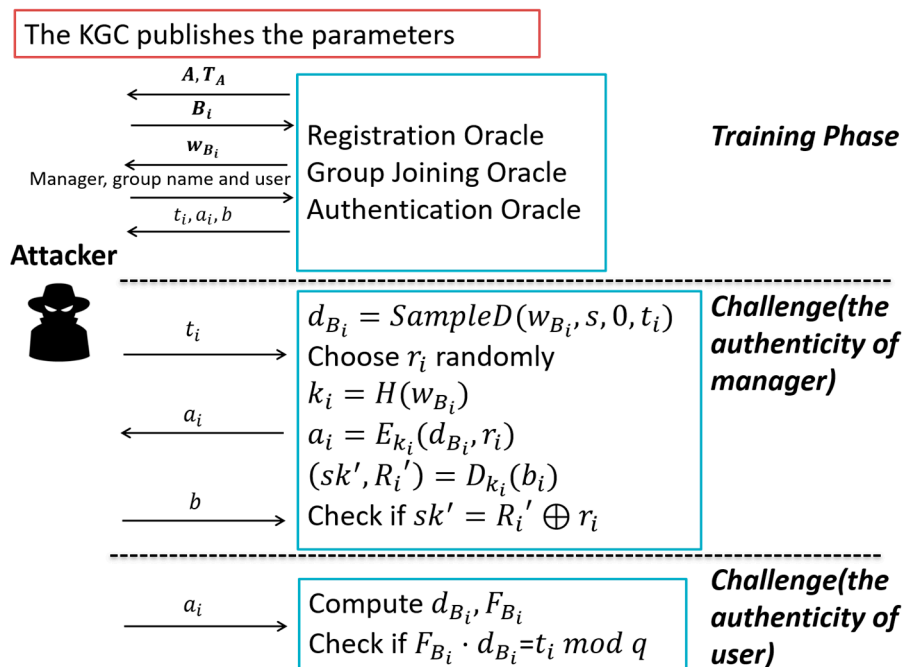
**Part I:** (The Authenticity of the Manager)

- The attacker sends  $t_i$  to the simulator. After receiving  $t_i$ , the simulator computes  $d_{B_i}$ .*
- The simulator chooses a number  $r_i$  randomly, computes  $k_i$ ,  $a_i$ , and sends  $a_i$  to the attacker.*
- The attacker sends  $b$  to the simulator. If the attacker is authenticated, the attacker wins the game.*

**Part II:** (The Authenticity of the User)

- The attacker sends  $a_i$  to the simulator. After receiving  $a_i$ , the simulator computes  $d_{B_i}$  and  $F_{B_i}$ .*
- The simulator checks if  $F_{B_i} \cdot d_{B_i} = t_i \text{ mod } q$ . If it is true, the attacker is authenticated and wins the game.*

*A scheme is said to achieve secure group mutual authentication if there is no polynomial-time adversary winning the above game with non-negligible advantage.*



**Figure A1.** The security model of group mutual authentication.

**Definition A2** (Session Key Security). *An attacker interacts with a simulator in the following game (Figure A2).*

**Setup.** *The KGC publishes the system parameters to the attacker.*

**Training Phase.** *The attacker is allowed to query the following oracles:*



- Registration oracle: The outputs of this oracle are the public key  $A$  and the secret key  $T_A$ .
- Group joining oracle: The input of this oracle is  $B_i$  and the output is  $w_{B_i}$ .
- Authentication oracle: The inputs of the oracle are a group manager name, the group name, and the user of the group. The outputs of the oracle are  $t_i$ ,  $a_i$  and  $b_i$ .

**Challenge.** First, the simulator runs the authentication protocol and generates a valid session key  $sk$ , and chooses another random session key  $sk''$ . Then, the simulator chooses a bit  $\beta$  and sets  $sk_\beta = sk$ ,  $sk_{1-\beta} = sk''$ . Next, the simulator sends  $sk_0$ ,  $sk_1$  to the attacker. After receiving  $sk_0$ ,  $sk_1$ , the attacker outputs a bit  $\beta'$ , and wins the game if  $\beta' = \beta$ .

A scheme is said to achieve a secure session key if there is no polynomial-time adversary winning the above game with non-negligible advantage.

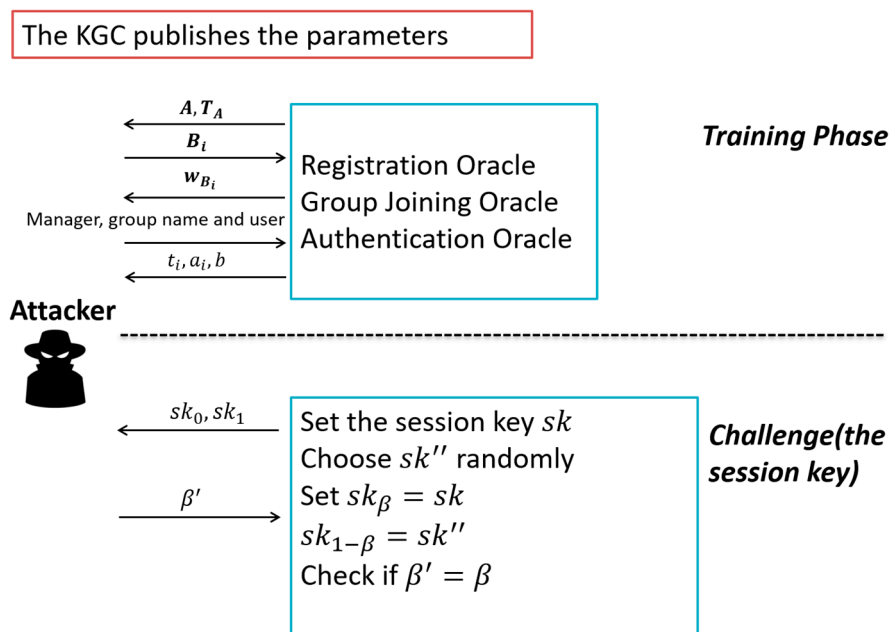


Figure A2. The security model of a secure session key.

### Appendix A.1.2. Security Proof

**Lemma A1** (Group Mutual Authentication Security). *If the underlying symmetric encryption scheme used in our authentication scheme is secure against adaptive chosen ciphertext attacks (IND-CCA2) and the SampleD function used in our scheme achieves one-way-ness, then our authentication scheme can achieve secure mutual authentication between the manager and the users.*

#### Proof.

##### Part I (The Authenticity of the Manager):

Assume that an attacker interacts with a simulator via the game defined in Definition A1. The simulator simulates the **Setup Phase** and **Training Phase** like those in the proposed scheme.

Then, the simulator simulates the **Challenge Phase** as follows. First, the attacker sends  $t_i$  to the simulator. Then, the simulator computes  $d_{B_i} = \text{SampleD}(w_{B_i}, s, 0, t_i)$  and chooses two numbers  $r_i^0, r_i^1$ . Next, the simulator sets  $m_0 = (d_{B_i}, r_i^0)$  and  $m_1 = (d_{B_i}, r_i^1)$ , and sends  $m_0, m_1$  to the IND-CCA2 game for symmetric encryption as the challenge messages. After receiving the challenge ciphertext  $c^*$  from the IND-CCA2 game, the simulator sends  $c^*$  to the attacker. Then, the simulator receives  $b$  from the attacker, and the simulator sends  $b$  to the decryption oracle of the IND-CCA2 game. In the case that

$Q' > 1$  (Figure A3), the simulator parses the returned plaintext as  $(sk', R'_i)$ . Then the simulator outputs 0 if  $sk' \oplus R'_i = r_i^0$ , and outputs 1 if  $sk' \oplus R'_i = r_i^1$ . In the case that  $Q' = 1$  (Figure A4), the simulator parses the returned plaintext as  $(r'', h)$ . Then the simulator outputs 0 if  $h = H(r_i^0)$ , and outputs 1 if  $h = H(r_i^1)$ .

If the attacker is able to pass the authentication with non-negligible advantage, then the plaintext in  $b$  should be correct. Therefore, the simulator is able to make a correct guess based on the decryption result of  $b$ , and win the IND-CCA2 game with the same advantage as the attacker.

**Part II (The Authenticity of the User, Figure A5):**

Assume that an attacker interacts with a simulator via the game defined in Definition A1, and the attacker sends a user set  $U \subseteq \{1, \dots, Q\} (|U| = Q')$  to the simulator. To simulate the **Setup Phase**, the simulator chooses  $B_i$  such that  $A' = \{A \parallel \sum_{i=1}^{Q'} (B_i - B_{i^*})\}$ , where  $i \in U$  and  $B_{i^*}$  is the target user that the attacker wants to impersonate. Other system parameters are set the same as in the proposed scheme. The simulator then simulates the **Training Phase** as in the proposed scheme. In the **Challenge Phase**, the simulator first outputs  $t_i$  to the attackers, and obtains  $a_i$  from the attacker. Then, the simulator performs as in the proposed scheme to obtain  $d_{B_i}$ , and check whether  $F_{B_i}d_{B_i} = t_i \text{ mod } q$ . If the formula holds—which means that the attacker passes the authentication and thus wins the game—then the attacker can be viewed as an inverter for the function SampleD. Therefore, we are able to break the one-way-ness of SampleD with the same advantage as the attacker.  $\square$

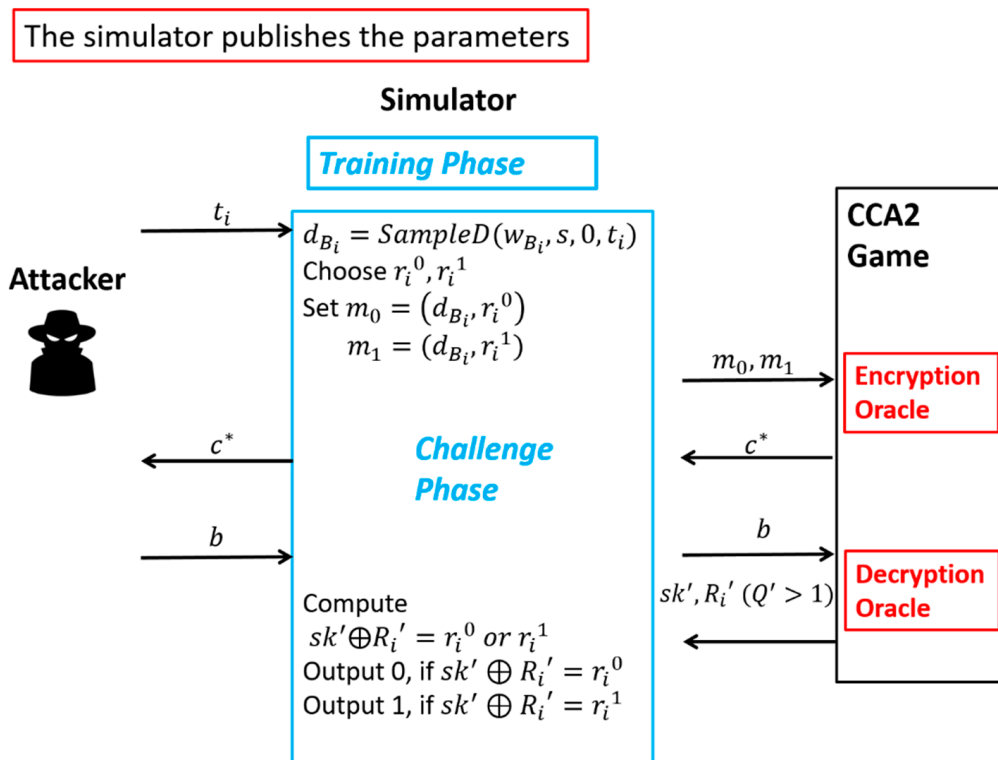


Figure A3. The authenticity of the manager ( $Q' > 1$ ).

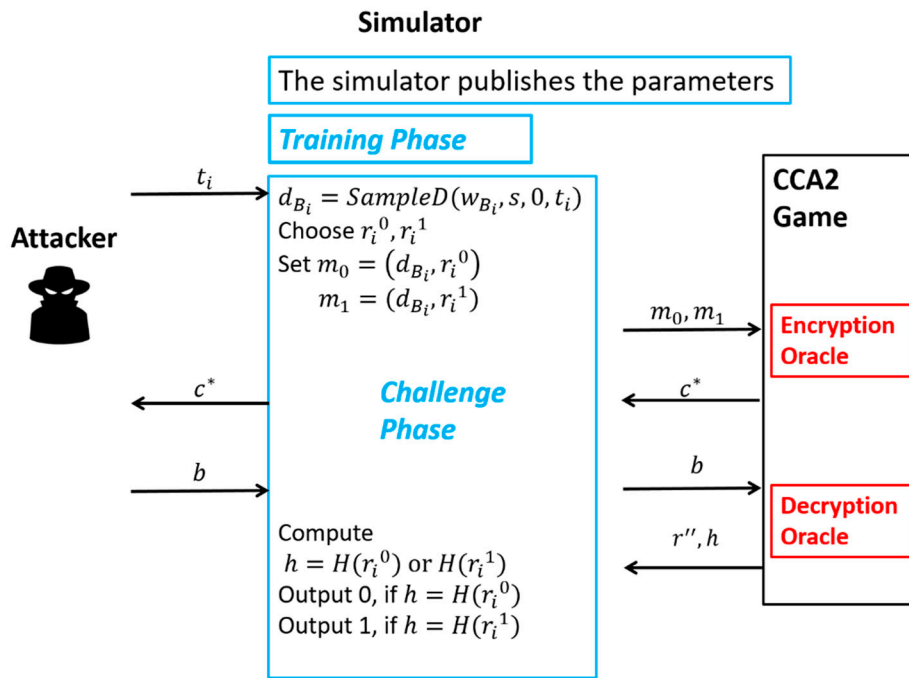


Figure A4. The authenticity of the manager ( $Q' = 1$ ).

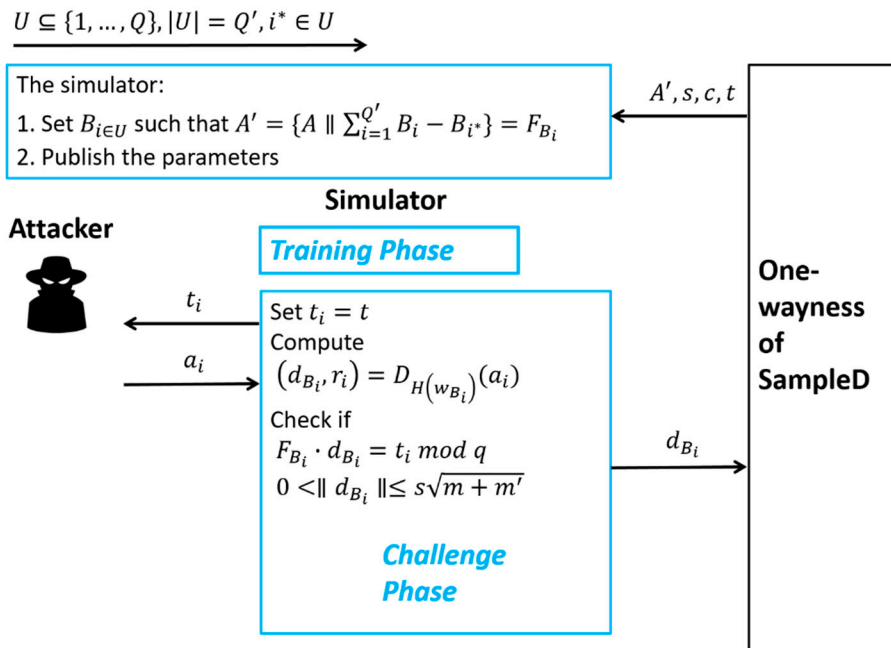


Figure A5. The authenticity of the user.

**Lemma A2** (Session Key Security). *If the symmetric encryption used in our authentication scheme is secure against adaptive chosen ciphertext attacks (IND-CCA2), then our scheme is a secure key exchange scheme.*

**Proof.** Assume that an attacker interacts with a simulator via the game defined in Definition A2. The simulator simulates the **Setup Phase** and **Training Phase** as in the proposed scheme.  $\square$

Then, the simulator simulates the **Challenge Phase** as follows.

**Case I** ( $Q' > 1$ , Figure A6): First, the simulator chooses  $j \in [1, Q']$ . Then, for  $i = 1$  to  $Q'$ , the simulator chooses  $t_i \in \mathbb{Z}_q$  and sends  $t_i$  to the attacker. Next, for  $i \in [1, Q'] / \{j\}$ , the simulator computes  $w_{B_i}$ ,  $d_{B_i}$ , and  $a_i$  like in the proposed scheme. For user  $j$ , the simulator chooses two random numbers  $r_j^0, r_j^1$  and sets  $m_0 = (d_{B_j}, r_j^0)$ ,  $m_1 = (d_{B_j}, r_j^1)$ . Then, the simulator sends  $m_0, m_1$  to the IND-CCA2 game to invoke the challenge phase. After receiving the challenge ciphertext  $c^*$ , the simulator sends  $a_j = c^*$  to the attacker. Then, the simulator sets  $r_j = r_j^0$  and computes  $sk$  and  $b_i$  for  $i \in [1, Q']$  as in the proposed scheme. Next, the simulator sends all  $b_i$  to the attacker. Finally, the simulator outputs  $sk_0 = sk$ ,  $sk_1 = r_1 \oplus \dots \oplus r_j^1 \oplus \dots \oplus r_{Q'}$  to the attacker. After receiving  $sk_0, sk_1$ , the attacker outputs a bit  $\beta'$  to the simulator. Then, the simulator outputs 0 if  $\beta' = 0$  to the IND-CCA2 game. Otherwise, the simulator outputs a random bit  $\beta$ . Let  $\epsilon$  be the advantage that the attacker wins the game. If the IND-CCA2 game chooses  $m_0$ , the simulator makes a correct guess with probability  $\epsilon$ . If the IND-CCA2 game chooses  $m_1$ , then the simulator makes a correct guess with probability  $\frac{1}{2}$ . Therefore, the simulator's advantage in winning the IND-CCA2 game is  $\left| \left( \epsilon + \frac{1}{2} \right) - \frac{1}{2} \right| = \epsilon$ , which is non-negligible.

**Case II** ( $Q' = 1$ , Figure A7): First, the simulator sends  $t_i$  to the attacker. Then, the simulator chooses  $r_i^0, r_i^1$  and sets  $m_0 = (d_{B_j}, r_i^0)$ ,  $m_1 = (d_{B_j}, r_i^1)$ . Next, the simulator sends  $m_0, m_1$  to the IND-CCA2 game to invoke the challenge phase. After receiving the challenge ciphertext  $c^*$ , the simulator sends  $a_j = c^*$  to the attacker. Then, the simulator sends  $(r', H(r_i^0))$  to the encryption oracle of the IND-CCA2 game and receives the ciphertext  $c$ . Next, the simulator sets  $b = c$  and sends  $b$  to the attacker. Finally, the simulator computes  $sk_0 = r' \oplus r_i^0$ ,  $sk_1 = r' \oplus r_i^1$ , and sends  $sk_0, sk_1$  to the attacker. After receiving  $sk_0, sk_1$ , the attacker outputs a bit  $\beta'$  to the simulator. Then, the simulator outputs 0 if  $\beta' = 0$  to the IND-CCA2 game. Otherwise, the simulator outputs a random bit  $\beta$ . Let  $\epsilon$  be the advantage that the attacker wins the game. If the IND-CCA2 game chooses  $m_0$ , the simulator makes a correct guess with probability  $\epsilon$ . If the IND-CCA2 game chooses  $m_1$ , then the simulator makes a correct guess with probability  $\frac{1}{2}$ . Therefore, the simulator's advantage in winning the IND-CCA2 game is  $\left| \left( \epsilon + \frac{1}{2} \right) - \frac{1}{2} \right| = \epsilon$ , which is non-negligible.

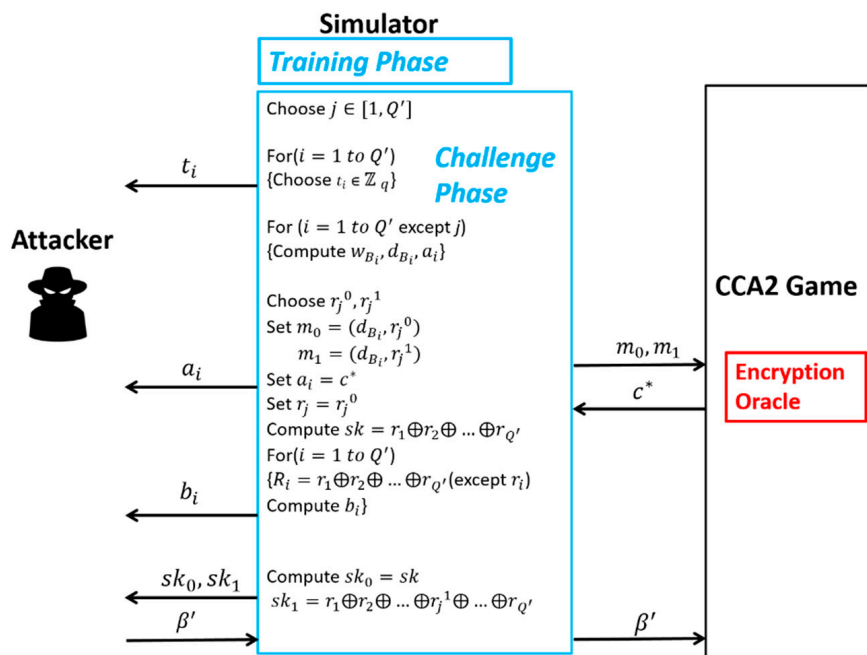


Figure A6. The session key security ( $Q' > 1$ ).

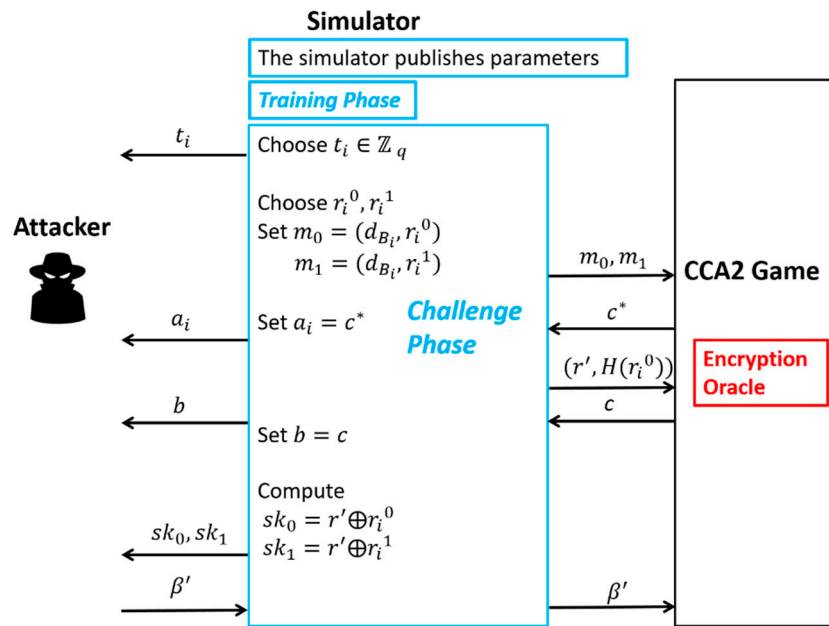


Figure A7. The session key security ( $Q' = 1$ ).

**Theorem A1.** *If the underlying symmetric encryption is secure against adaptive chosen ciphertext attacks (IND-CCA2) and the function Sample D achieves one-way-ness, then, according to Lemma A1, our authentication scheme is a secure group mutual authentication protocol. If the underlying symmetric encryption is secure against adaptive chosen ciphertext attacks (IND-CCA2), then, according to Lemma A2, our authentication scheme is a secure key exchange protocol.*

## References

- 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.
- Ajtai, M. Generating hard instances of lattice problems. In Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, Philadelphia, PA, USA, 22–24 May 1996; pp. 99–108.
- Moustaine, E.E.; Laurent, M. A lattice-based authentication for low-cost RFID. In Proceedings of the 2012 IEEE International Conference on RFID-Technologies and Applications (RFID-TA), Nice, France, 5–7 November 2012; pp. 68–73.
- Park, S.-W.; Lee, I.-Y. Mutual authentication scheme based on lattice for NFC-PCM payment service environment. *Int. J. Distrib. Sens. Netw.* **2016**, *12*. [[CrossRef](#)]
- Hoffstein, J.; Pipher, J.; Silverman, J.H. NTRU: A ring-based public key cryptosystem. In *International Algorithmic Number Theory Symposium*; Springer: Berlin, Germany, 1998; pp. 267–288.
- Mahabir, P.J.; Reihaneh, S.N. Compact accumulator using lattices. In Proceedings of the International Conference on Security, Privacy, and Applied Cryptography Engineering, Jaipur, India, 3–7 October 2015; Springer: Berlin, Germany, 2015; pp. 347–358.
- Alwen, J. Generating shorter bases for hard random lattices. *Theory Comput. Syst.* **2011**, *48*, 535–553. [[CrossRef](#)]
- Micciancio, D. Trapdoors for lattices: Simpler, tighter, faster, smaller. *EuroCrypt* **2012**, 7237, 700–718.
- 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; ACM: New York, NY, USA, 2008; pp. 197–206.
- Bellare, M.; Rogaway, P. Entity authentication and key distribution. *Crypto* **1993**, *93*, 232–249.

11. Cash, D.; Hofheinz, D.; Kiltz, E.; Peikert, C. Bonsai trees, or how to delegate a lattice basis. *Eurocrypt* **2010**, *6110*, 523–552.
12. Sahu, S.K.; Kushwaha, A. Performance analysis of symmetric encryption algorithms for mobile ad hoc network. *Int. J. Emerg. Technol. Adv. Eng.* **2014**, *4*, 619–624.
13. Follath, J. Gaussian sampling in lattice based cryptography. *Tatra Mt. Math. Publ.* **2014**, *60*, 1–23. [[CrossRef](#)]
14. Gaj, K.; Homsirikamol, E.; Rogawski, M. Fair and comprehensive methodology for comparing hardware performance of fourteen round two SHA-3 candidates using FPGAs. In Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, Santa Barbara, CA, USA, 17–20 August 2010; pp. 264–278.
15. Min, H.S.; Yeop, O.S.; Hyunsoo, Y. New modular multiplication algorithms for fast modular exponentiation. *EuroCrypt* **1996**, *1070*, 166–177.
16. Micciancio, D.; Regev, O. Lattice-based cryptography. In *Post-Quantum Cryptography*; Springer: Berlin, Germany, 2009; pp. 147–191.
17. Krawczyk, H.; Bellare, M.; Canetti, R. *HMAC: Keyed-Hashing for Message Authentication*; ACM: New York, NY, USA, 1997. Available online: <https://tools.ietf.org/html/rfc2104> (accessed on 10 July 2017).



© 2018 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 (<http://creativecommons.org/licenses/by/4.0/>).