A Secure and Verifiable Access Control Scheme for Big Data Storage in Clouds

Chunqiang Hu, *Member, IEEE*, Wei Li, *Member, IEEE*, Xiuzhen Cheng, *Fellow, IEEE*
Jiguo Yu, *Member, IEEE*, Shengling Wang, *Member, IEEE*, and Rongfang Bie, *Member, IEEE*

Abstract—Due to the complexity and volume, outsourcing ciphertexts to a cloud is deemed to be one of the most effective approaches for big data storage and access. Nevertheless, verifying the access legitimacy of a user and securely updating a ciphertext in the cloud based on a new access policy designated by the data owner are two critical challenges to make cloud-based big data storage practical and effective. Traditional approaches either completely ignore the issue of access policy update or delegate the update to a third party authority; but in practice, access policy update is important for enhancing security and dealing with the dynamism caused by user join and leave activities. In this paper, we propose a secure and verifiable access control scheme based on the NTRU cryptosystem for big data storage in clouds. We first propose a new NTRU decryption algorithm to overcome the decryption failures of the original NTRU, and then detail our scheme and analyze its correctness, security strengths, and computational efficiency. Our scheme allows the cloud server to efficiently update the ciphertext when a new access policy is specified by the data owner, who is also able to validate the update to counter against cheating behaviors of the cloud. It also enables (i) the data owner and eligible users to effectively verify the legitimacy of a user for accessing the data, and (ii) a user to validate the information provided by other users for correct plaintext recovery. Rigorous analysis indicates that our scheme can prevent eligible users from cheating and resist various attacks such as the collusion attack.

Index Terms—Big Data Storage; Access Control; the NTRU Cryptosystem; Secret Sharing; Access Policy Update; Cloud Computing.

1 Introduction

BIG data is a high volume, and/or high velocity, high variety information asset, which requires new forms of processing to enable enhanced decision making, insight discovery, and process optimization [1]. Due to its complexity and large volume, managing big data using on hand database management tools is difficult. An effective solution is to outsource the data to a cloud server that has the capabilities of storing big data and processing users' access requests in an efficient manner. For example in e-health applications, the genome information should be securely stored in an e-health cloud as a single sequenced human genome is around 140 gigabytes in size [2], [3]. However, when a data owner outsources its data to a cloud, sensitive information may be disclosed because the cloud server is not trusted; therefore typically the ciphertext of the data is stored in the could. But how to update the ciphertext stored in a cloud when a new access policy is designated by the data owner and how to verify the legitimacy of a user who intends to access the data are still of great concerns.

Most existing approaches for securing the outsourced big data in clouds are based on either *attributed-based encryption (ABE)* or *secret sharing*. ABE based approaches [4]–[11] provide the flexibility for a data owner to predefine the set of users who are eligible for accessing the data but they suffer from the high complexity of efficiently updating the access policy and ciphertext. Secret sharing [11]–[17] mechanisms allow a secret to be shared and reconstructed by certain number of cooperative users but they typically employ asymmetric public key cryptograph such as RSA for users' legitimacy verification, which incur high computational overhead. Moreover, it is also a challenging issue to dynamically and efficiently update the access policies according to the new requirements of the data owners in secret sharing approaches.

As a data owner typically does not backup its data locally after outsourcing the data to a cloud, it cannot easily manage the data stored in the cloud. Besides, as more and more companies and organizations are using clouds to store their data, it becomes more challenging and critical to deal with the issue of access policy update for enhancing security and dealing with the dynamism caused by the users' join and leave activities. To the best of our knowledge, policy update for outsourced big data storage in clouds has never been considered by the existing research [13], [17]–[20].

Another challenging issue is how to verify the legitimacy of the users accessing the outsourced data in clouds. Existing schemes proposed in [7], [8], [10], [21] do not support user eligibility verification. On the other hand, verifiable secret sharing based schemes rely on RSA [13]–[15] for access legitimacy verification. As multiple users need to mutually verify each other using multiple RSA operations, such a procedure has a high computational overhead. Furthermore, the classic asymmetric crypto solutions such as RSA could be broken by quantum computing in the near future. This is not a science fiction as in 2015 IBM brought quantum computing closer to reality [22], making it urgent to exploit new techniques for quantum computing attack

Manuscript received 1 Feb., 2016; revised 25 June, 2016, accepted 15 Oct., 2016.

†*Chunqiang Hu and Xiuzhen Cheng are with the Department of Computer Science, The George Washington University, Washington DC 20052,USA. E-mail: {chu, cheng}@gwu.edu.*
Wei Li is with the Department of Computer Science, Georgia State University, E-mail: wli28@gsu.edu.
†*Jiguo Yu is with the School of Information Science & Engineering, Qufu Normal University, Rizhao, 276826, P. R. China. E-mail: jiguoyu@sina.com.*
Shengling Wang and Rongfang Bie are with the College of Information Science & Technology, Beijing Normal University, Beijing, China. Email: {wangshengling,rfbie}@bnu.edu.cn.
†*corresponding author.*

resistance. The NTRU cryptosystem is a type of lattice-based cryptography [23]–[25], and its security is based on the shortest vector problem (SVP) in a lattice [26]. The major advantages of NTRU are quantum computing attack resistance and lighting fast computation capability. However, NTRU suffers from the problem of decryption failures [27]–[30].

The considerations mentioned above motivate us to develop a verifiable access control scheme for securing the big data stored in clouds, tackling the challenges of the following security services:

- *Security*. The proposed scheme should be able to defend against various attacks such as the collusion attack. Meanwhile, access policy update should not break the security of the data storage, disclose sensitive information about the data owner, and cause any new security problem.
- *Verification*. When a user needs to decrypt a stored ciphertext, its access legitimacy should be verified by other participating users and the secret shares obtained from other users must be validated for correct recovery.
- *Authorization*. To reduce the risk of information leakage, a user should obtain authorization from the data owner for accessing the encrypted data.

In this paper, we first propose an improved NTRU cryptosystem to overcome the decryption failures of the original NTRU. Then we design a secure and verifiable scheme based on the improved NTRU and secret sharing for big data storage. The cloud server can directly update the stored ciphertext without decryption based on the new access policy specified by the data owner, who is able to validate the update at the cloud. The proposed scheme can verify the shared secret information to prevent users from cheating and can counter various attacks such as the collusion attack. It is also deemed to be secure with respect to quantum computing attacks due to NTRU. The multi-fold contributions of the paper can be summarized as follows:

- We propose a new NTRU decryption procedure to overcome the decryption failures of the original NTRU without reducing the security strength of NTRU.
- We propose a secure and verifiable access control scheme to protect the big data stored in a cloud. The scheme can verify a user's access legitimacy and validate the information provided by other users for correct plaintext recovery.
- We devise an efficient and verifiable method to update the ciphertext stored in clouds without increasing any risk when the access policy is dynamically changed by the data owner for various reasons.
- We prove the correctness of the proposed scheme and investigate its efficiency and security strength. Particularly, we demonstrate that our scheme can resist various attacks such as the collusion attack via a rigorous analysis.

The rest of the paper is organized as follows. Section 2 introduces the motivation and the related work. Section 3 presents our system model, security model, and the preliminary knowledge employed by our scheme. Section 4 proposes an improved NTRU cryptosystem to overcome the decryption failures of the original NTRU cryptosystm. Section 5 details our secret sharing based scheme for big data storage in clouds. The correctness, security properties, and performance of our proposed scheme are elaborated in Section 6. Conclusions and future work are discussed in Section 7.

2 Motivation and Related Work

2.1 Motivation

In this information era, companies and organizations are facing a challenging problem of effectively managing their complex data. As the development of cloud storage, outsourcing the data to a cloud is an appropriate approach. Generally speaking, clouds can be classified into two major categories: i) public clouds with each being a multi-tenant environment shared with a number of other tenants, and ii) private clouds with each being a single-tenant environment dedicated to a single tenant. For example, the IBM cloud was proposed as a public one for the data management of banking [31]. When a bank stores its data in the could server as shown in Fig. 1, only its legal staff members have the rights to access the stored data. Typically the bank system contains many sensitive and private consumer information. In order to reduce the risk of information leakage, the access right of an employee should be properly restricted, and a single employee should not be allowed to reveal the data by itself without obtaining the authorization from other users; that is, recovering the data requires to get the authorization of multiple employees. Moreover, the bank should be able to update the access policies for the stored data in a dynamic and efficient manner. Similarly, military applications can utilize a private cloud to store their complex data as shown in Fig. 2. Since the data is confidential, a military member, who needs to access the data, must pass the verification of its legitimacy and receive the authorization from multiple relevant departments. Besides, the military should be able to dynamically and efficiently update its access polices based on the changing requirements.

Such applications usually require the data to be stored in a cloud in ciphertext format, and the access of the data by a user requires multiple other users to verify the access legitimacy of the user. Therefore in this paper we propose a secure and verifiable access control scheme for big data storage to tackle the following challenges: i) how to securely store the data in a cloud server and distribute the shares of the access right to all legitimate users of the data? ii) how to verify the legitimacy of a user for accessing the data? iii) how to recover the plaintext data when the access right needs to be jointly granted by multiple users? and iv) how to dynamically and efficiently update the ciphertext in the cloud when the access policy of the data is changed by the data owner? To overcome these challenges, we make use of the following techniques in the design of our secure and verifiable access control scheme for big data storage. First, a plaintext data is bound to a secret that is shared by all legitimate users of the data based on (t, n)-threshold secret sharing, and a message certificate is computed for the data based on the NTRU encryption; the ciphertext is produced from both the shared secret and the message certificate. Second, the legitimacy of a user for accessing the data is verified by both the data owner and at least $t-1$ other legal users of the data, and the information provided by other users for the plaintext recovery needs to be validated by the user to prevent against cheating behaviors. Third, the plaintext data can be obtained when at least $t-1$ other users participate in the recovery process and provide correct information for the data recovery, based on (t, n)-threshold secret sharing. Last, the access policy of the data and the secret shares bound to the data can be dynamically changed by the data owner, and the update of the ciphertext is conducted by the cloud server without the need of downloading the previous ciphertext from the cloud to the data

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2016.2621106, IEEE Transactions on Big Data

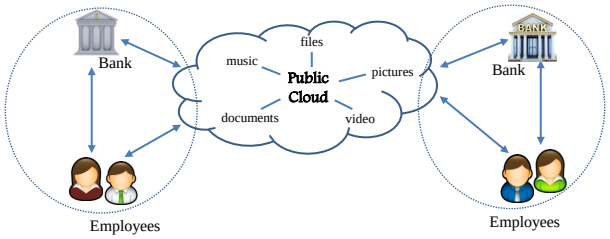


Fig. 1. An example application of big data storage in banking.

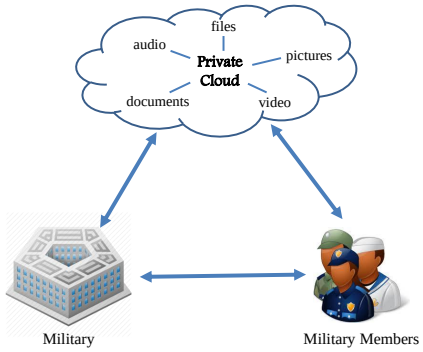


Fig. 2. An example application of big data storage in military.

owner. Meanwhile, the data owner is able to verify whether the ciphertext stored in the cloud is correctly updated.

2.2 Related Work

Because big data frequently contains a huge amount of personal identifiable information, how to securely store the data and how to provide access control over the stored data are two biggest challenges [32]. In this subsection, we mainly summarize the state-of-the-art of securing the big data stored in clouds.

Outsourcing to clouds is one of the most popular approaches to securing the big data storage [33]–[35], in which the data owners encrypt their data based on cryptographic primitives and store the encrypted data to the clouds. In outsourcing, a secure mechanism should be established between a data owner and a cloud. In order for the cloud to perform operations over the encrypted data, "Fully Homomorphic Encryption" (FHE) [36] was usually adopted, which allows direct addition and multiplication operations over the ciphertexts while preserving decryptability. Homomorphic encryption was also applied to guarantee the security of data storage [37], [38]. Nevertheless, it is an immature cryptosystem, and is extremely inefficient in practice, which renders it hardly applicable in real world applications. Securely outsourcing big data computations to the clouds was also extensively studied [39]–[41] but this topic is out of the scope of the paper.

Adequate access control is key to protect the stored data. Access control has traditionally been provided by operating systems or applications restricting access to the information, which typically exposes all the information if the system or application is hacked [42]. A better approach is to protect the information using encryption that only allows decryption by authorized entities. Attribute-Based Encryption (ABE) [4], [7], [8], [10], [11] is one of the most powerful techniques for access control in cloud storage systems. In the past years, quite a few attribute-based access control schemes [19], [20], [43], [44] were proposed, in which the data owners define the access policies based on the attributes required by the data and encrypt the data based on the access policies. By this way the data owners are able to ensure that only the users meeting the access policies can decrypt the ciphertexts. However, it is difficult to update the policies when these ABE based schemes are applied because the data owners do not store the data in their local systems once they outsource the data into the cloud servers. It is also difficult to verify the legitimacy of the downloaded data as the clouds housing the data are not trustworthy. Besides, the operations of encryption and decryption in ABE have a high computational overhead and incur a large energy consumption.

Secret sharing [45] is another powerful technique to protect the big data in cloud storage. The most related work to our proposed scheme are [15] and [14], whose verification procedure can resist potential attacks such as collusion and cheating. In [15], two schemes were proposed, namely Scheme-I and Scheme-II, based on the homogeneous linear recursion and the RSA cryptosystem, in which the homogeneous linear recursion is used to construct the secret share and reconstruct the secret, and RSA is used to verify the users' access legitimacy. The difference between these two schemes lies in that the users in Scheme-I mutually verify each other's legitimacy without seeking help from public values while in Scheme-II the users need the help of public values. In [14], the authors presented a verifiable multi-secret sharing scheme based on cellular automata, which is used to construct the secret share and reconstruct the secret with a linear computational complexity, and the RSA cyptosystem, which is used for verification. In these schemes, as multiple users mutually verify each other using multiple RSA operations, a very high computational overhead occurs. In addition, the classic asymmetric crypto solutions would be broken by quantum computing; that is, these traditional verification methods cannot satisfy the verification requirements with respect to quantum computing, which is made closer to reality by IBM in 2015 [22]. Thus we need to seek new verification methods to meet the future requirements. For this purpose, we utilize the NTRU cryptosystem to counter the quantum computing attacks in the design of our proposed scheme.

Delegation is a popular approach for policy update. In [7], a user generates a new private key using its previous private key, and then delegates the new private key to a local authority for access policy update. In [46], a procedure called "ciphertext delegation" was designed for the third party to 're-encrypt' the ciphertext to a more restrictive policy using only public information. These two approaches cannot satisfy our security requirements because they delegate the private key/ciphertext for a new access policy that is more restrictive than the old one - in our perspective, the access policy to a ciphertext might need to be relaxed as time goes for many real world applications. Therefore in our design, we allow the new access policy designated by the data owner to be independent than the previous one, and the ciphertext is updated by the cloud based on the new access policy.

3 Models and Preliminaries

3.1 System Model

We consider a cloud storage system that is applicable for both public and private clouds as shown in Fig. 3. It consists of the following three types of entities: cloud server, data owner (owners), and data user (users).

Cloud server. A cloud server provides spaces for data owners to store their outsourced ciphertext data that can be retrieved by

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2016.2621106, IEEE Transactions on Big Data

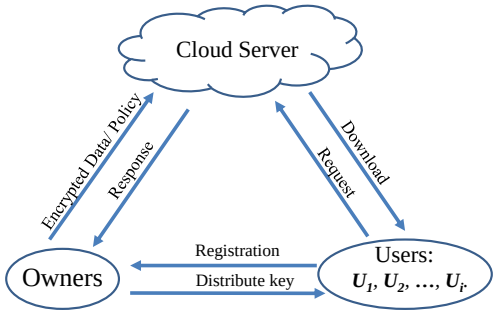


Fig. 3. System model

the users. It is also responsible for updating the ciphertexts when the data owner changes its access policy.

Owners. A data owner designates the access policy for its data, encrypts the data based on the access policy before outsourcing the data to the cloud server, and requests the cloud server to update the encrypted data when a new access policy is adopted. It can also check whether the ciphertext at the cloud server is correctly updated.

Users. Each user is assigned with a sub-key for an encrypted data the user is eligible to access. In order to decrypt the ciphertext, the user's eligibility must be verified by at least $t-1$ other users that are also eligible to access the data. The information provided by the $t-1$ verifiers must be validated by the user for correct message decryption based on the (t,n)-threshold secret sharing.

For a piece of data to be stored in a cloud, the data owner generates a public key and privacy key pair, defines an access policy, and computes a sub-key for each potential user based on the policy. Then, the data owner produces a message certificate for the data, and stores the encrypted data with the access policy in the cloud. When a user needs to use the data, it solicits help from other users to recover the data. The cloud server can update the encrypted data with a new policy is designated by the data owner.

The access policy is defined by a $t-1$ degree polynomial $b(x) = b_0 + \sum_{j=1}^{t-1} b_j x^j$ in this paper. Specifically, the data owner splits the access right of an encrypted plaintext into n pieces for n users, with one piece for each legitimate user of the data. A user can recover the plaintext data if and only if it obtains the message certificate from the data owner and holds at least t pieces of the access rights (with the help of at least $t-1$ other legal users), where t is a threshold designated by the data owner for access control based on (t,n)-threshold secret sharing.

3.2 Adversarial Model and Security Objectives

We assume that the cloud server honestly follows the protocol, i.e. it sends the ciphertext to a user when receiving a request from the user, and updates the ciphertext in its storage when requested by the owner of the data. The data owners do not have incentives to cheat the end users as their goals are to secure the data stored in the cloud server, but one data owner may intend to recover the data belonging to a different data owner. The users do not have to be honest, i.e., they may have different cheating behaviors. For example, a user may provide incorrect information for the decryption of a ciphertext by other users, causing the decryption to fail; multiple users may collude to reveal the message; a user may forge a message certificate; just to name a few.

We intend to achieve the following security objectives when designing our access control scheme for big data storage in clouds:

- Data confidentiality. The outsourced data must be protected from eavesdropping attacks during upload, download, update, and retrieval of the data.
- Verification. The legitimacy of any user to access the data must be verifiable; the information provided by any user for plaintext recovery must be validated.
- Attack resistance. The proposed scheme must be able to counter potential attacks (cheating, collusion attacks, forging attacks, etc.) launched by misbehaving users and adversaries.

3.3 Preliminaries

In this section, we introduce the following two cryptographic primitives: NTRU and secret sharing. For better elaboration, we present the notations and their semantic meanings in Table 1.

TABLE 1
The Notations and Their Semantic Meanings

Notations	means
D	The data owner
U_i	The ith user, $1 \le i \le n$
id	The id of a user
$\{H, H_1\}$	Two one-way hash functions
(h, f)	A pair of keys, with h being the public key and f being the private key.
S	The set of M messages $S = \{S_1, S_2, \ldots, S_M\}$
R	R is a ring, i.e., $R = Z[X]/(X^N - 1)$
p, q	Two integers in R
f	A polynomial in R
g	A polynomial in $Z[X]/(q, X^N - 1)$
f_p	A polynomial in $Z[X]/(p, X^N - 1)$
f_q	A polynomial in $Z[X]/(q, X^N - 1)$
L_f	The set of polynomials in R
L_g	The set of polynomials in R
AES	Advanced Encryption Standard

3.3.1 The NTRU Cryptosystem

The NTRU cryptosystem is based on the shortest vector problem (SVP) in a lattice that makes it lightning fast and resistant to quantum computing attacks. It has been proved to be faster than RSA [28], [29]. In this subsection, we briefly introduce NTRU, and refer the interested readers to [27] and [26] for more details.

NTRU consists of three integer parameters (N, p, q) and four sets L_f, L_g, L_ϕ and L_m of polynomials of degree $N-1$ with integer coefficients. Note that p and q are not required to be prime, but they should simultaneously meet the following two criterions: $gcd(p, q) = 1$ and $q > p$. NTRU works in the ring $R = Z[X]/(X^N - 1)$, and L_f, L_g, L_ϕ and L_m are defined in R whose selection criteria are detailed in [26]. Any element $f \in L_f$ can be expressed as a polynomial or a vector as follows:

$$f = \sum_{i=0}^{N-1} f_i x^i = [f_0, f_1, \ldots, f_{N-1}]. \quad (1)$$

We use $*$ to denote the convolution multiplication of two polynomials f and g in R, which can be computed via (2).

$$f * g = h \quad \text{with} \quad h_k = \sum_{i=0}^{k} f_i g_{k-i} + \sum_{i=k+1}^{N-1} f_i g_{N+k-i}, \quad (2)$$

where $k \in [0, N-1]$. Note that when we do a multiplication of two polynomials modulo q, we mean to reduce the coefficients of the product modulo q.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2016.2621106, IEEE Transactions on Big Data

NTRU implements the following three basic functions. For better presentation, we use Alice and Bob to represent the two entities for encrypting and decrypting a message.

- *Key Generation*: To create his public and private keys, Bob randomly chooses two polynomials $g \in L_g$ and $f \in L_f$, in which f is his private key and is required to have inverse modulo q and inverse modulo p. Let f_q and f_p respectively denote the inverse modulo q and the inverse modulo p of f, which are formally defined as follows:

$$f_q * f = 1 \pmod{q} \quad \text{and} \quad f_p * f = 1 \pmod{p}. \quad (3)$$

  For any properly selected f, it is easy to compute f_q and f_p via the Euclidean algorithm. Then Bob computes his public key h according to (4), and publishes his public parameters $\{p, q, h\}$.

$$h = f_q * g \pmod{q}. \quad (4)$$

- *Encryption*: Assume that Alice wants to send a message m to Bob, she needs to first convert m into a polynomial in L_m. For simplicity, we use m to denote the polynomial representation of m in L_m too. Then she randomly chooses a polynomial $\phi \in L_\phi$, encrypts m using Bob's public key h according to (5), and sends the encrypted message e to Bob.

$$e = p\phi * h + m \pmod{q}. \quad (5)$$

- *Decryption*: Bob receives the encrypted message e from Alice, and decrypts it using his private key f and the inverse of f module p, i.e., f_p, via (6) and (7).

$$a = e * f \pmod{q}. \quad (6)$$

$$m = a * f_p \pmod{p}. \quad (7)$$

Remarks: In NTRU, if the system parameters are properly selected, the probability of correctly recovering the plaintext message is extremely high; otherwise, the probability of decryption failures is extremely high. There are two types of decryption failures: the *Wrap failure* and the *Gap failure*, which will be discussed in Section 4.

3.3.2 Secret Sharing Scheme

Secret sharing schemes have been extensively studied [13], [17], ever since its development by Shamir [45] and Blakley [47] in 1979. Generally speaking, secret sharing can be briefly described as follows. In the context of a *dealer* sharing a secret with a number of *users* $U_1, \ldots, U_n$, a user learns the secret if and only if it can cooperate with at least $t-1$ other users (on sharing what they learn from the dealer), where $t \leq n$ is a pre-determined parameter. The secret to be shared by the dealer and the users is $s \in GF(p_1)$, where $p_1 > N$. Each user U_i holds a secret key $k_i \in GF(p_1)$, which is only known by U_i and the dealer.

The dealer follows a two-step process. First, it constructs a polynomial function $F(x)$ of degree $t-1$ shown in (8):

$$F(x) = s + \sum_{j=1}^{t-1} \mu_j x^j, \quad (8)$$

by randomly choosing each μ_j i.i.d. with a uniform distribution from $GF(p_1)$. Note that all (addition and multiplication) operations used in (8) is modular arithmetic (defined over $GF(p_1)$) as opposed to real arithmetic. Also note that s is the constant component of $F(x)$ - i.e., $s = F(0)$. Then, in the second step, the dealer transmits to each U_i a shared secret $s_i = F(x_i)$, where x_i is a random number selected by U_i for sharing the secret s and is sent to the dealer via the secure channel protected by the shared key k_i.

We now show how t or more users can cooperate to recover s by sharing the secrets received from the dealer. Without loss of generality, let $U_1, \ldots, U_t$ be the cooperating users. These t users can reconstruct the secret $s = F(0)$ from $s_1 = F(x_1), \ldots, s_t = F(x_t)$ by computing

$$s = F(0) = \sum_{j=1}^{t} \left(s_j \prod_{i \in [1,n], i \neq j} \frac{0 - x_i}{x_j - x_i} \right). \quad (9)$$

Note that the cumulative product in (9) is essentially the Lagrange coefficient. The correctness of (9) can be easily verified based on the definition of $F(x)$.

In secret sharing, a user may cheat when recovering the secret s. For example, a user U_i may provide an incorrect s_i, making the recovery of s fail. In order to overcome this problem, verifiable secret sharing schemes [13]–[15] have been proposed, which are mainly based on the RSA cryptosystem that incurs high computational overhead. In this paper, we propose a new validation method to prevent a user from submitting fake information for the correct message recovery based on the NTRU cryptosystem.

4 An Improved NTRU Cryptosystem

As reported in Section 3.3.1, there exist two types of decryption failures in NTRU: the Wrap failure and the Gap failure. In other words, the decryption process of NTRU cannot always output a correct decrypted message. To overcome this problem, we propose an improved NTRU crytosystem in this section. To proceed, we first analyze the reasons of the decryption failures in the original NTRU. When a ciphertext $e = p\phi * h + m \pmod{q}$ is received, it is decrypted according to the following two steps:

1) Compute $a = e * f \pmod{q}$;
2) Calculate $m' = a * f_p \pmod{p}$.

As analyzed in [48], the NTRU decryption could correctly recover the plaintext m if $a = (a_0, a_1, a_2, \cdots, a_{N-1})$ remains unchanged after the module q operation in step 1. This indicates that as long as $a_i \in (-\frac{q}{2}, \frac{q}{2})$, $i = 0, 1, \cdots, N-1$, before the $\bmod\ q$ operation, m can be correctly recovered. If there is at least one a_i with $|a_i| \geq \frac{q}{2}$, the Wrap failure could happen, resulting in $m' \neq m$ with a high probability; if $\max_{0 \leq i \leq N-1}\{a_i\} - \min_{0 \leq i \leq N-1}\{a_i\} > q$, the Gap failure could happen, leading to $m' \neq m$ with a high probability. From the above analysis, one can see that the Gap failure leads to the Wrap failure and that overcoming the Wrap failure can also getting rid of the Gap failure.

In order to overcome the Wrap failure, we propose the following improved NTRU decryption algorithm. For $a = e * f = p\phi * h * f + m * f$, we first figure out $\Gamma = \max\{|\max_{0 \leq i \leq N-1}\{a_i\}|, |\min_{0 \leq i \leq N-1}\{a_i\}|\}$ in a, and then compute $\tau = \lfloor \frac{\Gamma}{q/2} \rfloor$. If $\tau = 0$, which implies that each $a_i \in (-\frac{q}{2}, \frac{q}{2})$ in a, we can decrypt e to get m via the traditional approach; otherwise, we adjust a to obtain $a' = a - c^{(1)} - c^{(2)} - \cdots - c^{(\tau)}$, where $c^{(1)}, c^{(2)}, \cdots, c^{(\tau)}$ are the adjusting vectors and $c^{(j)} = (c_0^{(j)}, c_1^{(j)}, \cdots, c_{N-1}^{(j)})$ for $j = 1, 2, \cdots, \tau$.

Next, we present how to select the adjusting vectors $\{c^{(1)}, c^{(2)}, \cdots, c^{(\tau)}\}$. Consider any $a_i \in a$. Let $\gamma = \lfloor \frac{|a_i|}{q/2} \rfloor$. There are two cases:

- **Case** 1: If $a_i \geq 0$, it can be denoted by $a_i = \frac{q-1}{2}\gamma + c_i^{(\gamma+1)}$, where $0 \leq c_i^{(\gamma+1)} < \frac{q}{2}$. There are two sub-cases:
  - Case 1-1: If $\gamma = 0$, we have $a'_i = a_i$; then set $c_i^{(1)} = c_i^{(2)} = c_i^{(3)} = \cdots = c_i^{(\tau)} = 0$.
  - Case 1-2: If $\gamma \geq 1$, set $c_i^{(1)} = c_i^{(2)} = \cdots = c_i^{(\gamma)} = \frac{q-1}{2}$ and $c_i^{(\gamma+2)} = \cdots = c_i^{(\tau)} = 0$; then $a'_i = a_i - c_i^{(1)} - c_i^{(2)} - \cdots - c_i^{(\gamma)} = c_i^{(\gamma+1)}$.
- **Case** 2: If $a_i < 0$, it can be denoted as $a_i = -\frac{q-1}{2}\gamma + c_i^{(\gamma+1)}$, where $-\frac{q}{2} < c_i^{(\gamma+1)} < 0$. There also exist two sub-cases:
  - Case 2-1: If $\gamma = 0$, we have $a'_i = a_i$; set $c_i^{(1)} = c_i^{(2)} = c_i^{(3)} = \cdots = c_i^{(\tau)} = 0$.
  - Case 2-2: If $\gamma \geq 1$, set $c_i^{(1)} = c_i^{(2)} = \cdots = c_i^{(\gamma)} = -\frac{q-1}{2}$ and $c_i^{(\gamma+2)} = \cdots = c_i^{(\tau)} = 0$; then we have $a'_i = a_i - c_i^{(1)} - c_i^{(2)} - \cdots - c_i^{(\gamma)} = c_i^{(\gamma+1)}$.

After applying the above process to all a_i's, we obtain a' whose elements are guaranteed to be in $(-\frac{q}{2}, \frac{q}{2})$. Then the decryption procedure can be formalized by the following four steps:

1) Compute $a = e * f$;
2) Calculate the adjusting vectors $c^{(1)}, c^{(2)}, \cdots, c^{(\tau)}$ according to the procedure described above;
3) Compute $a' = a - c^{(1)} - c^{(2)} - \cdots - c^{(\tau)}$;
4) Compute $m' = a' * f_p + \sum_{j=1}^{\tau} c^{(j)} * f_p \pmod p$.

Note that the adjusting vectors $c^{(1)}, c^{(2)}, \cdots, c^{(\tau)}$ can guarantee that any value in a' is in $(-\frac{q}{2}, \frac{q}{2})$; therefore the modular q operation in the first step of the original NTRU decryption is not needed. The plaintext message m can be correctly recovered in the last step because $a' * f_p = p\phi * h * f * f_p + m * f * f_p - c^{(1)} * f_p - c^{(2)} * f_p - \cdots - c^{(\tau)} * f_p = m * f * f_p - c^{(1)} * f_p - c^{(2)} * f_p - \cdots - c^{(\tau)} * f_p \pmod p$. The pseudo-code of the improved NTRU decryption is presented in **Algorithm** 1. Combining with the original NTRU encryption we obtain an improved NTRU cryptosystem that can successfully get rid of the two types of decryption failures.

Theorem 1. The improved NTRU cryptosystem correctly overcomes the Wrap failure and the Gap failure in the decryption procedure of the original NTRU.

Proof According to [48] and our own analysis mentioned above, one can see that the root cause of the Gap failure and the Wrap failure in the original NTRU decryption is that $|a_i| \geq q/2$ exists for some a_i in $a = e * f$. Our improved NTRU decryption procedure first computes a' from a to ensure that $-\frac{q}{2} < a'_i < \frac{q}{2}$ for $\forall a'_i \in a'$. Then the original message m is recovered by computing $a' * f_p + c^{(1)} * f_p + \cdots + c^{(\tau)} * f_p \pmod p$.

Considering the four steps of the improved NTRU decryption, we notice that the first three steps can guarantee that no Gap or Wrap failure occurs during the computation of a'. The last step recovers m from a' and the adjusting vectors $\{c^{(1)}, c^{(2)}, \cdots, c^{(\tau)}\}$, which does not introduce the Gap failure and the Wrap failure. This completes the proof.

Algorithm 1 The Improved NTRU Decryption

Input: cipher text e, secret key $\{f, f_p\}$.
Output: plaintext m;
The decryptor computes $a = e * f$;
$\Gamma = \max\{|\max_{0\leq i\leq N-1}\{a_i\}|, |\min_{0\leq i\leq N-1}\{a_i\}|\}$;
$\tau = \lfloor \frac{\Gamma}{q/2} \rfloor$;
If $\tau = 0$
  $m = a * f_p \pmod p$.
Else
  For $0 \leq i \leq N-1$,
    Compute $\gamma = \lfloor \frac{|a_i|}{q/2} \rfloor$;
    If $\gamma = 0$
      $a'_i = a_i$ and $c_i^{(1)} = c_i^{(2)} = \cdots = c_i^{(\tau)} = 0$;
    Else If $a_i \geq 0$
      $a'_i = a_i - \frac{q-1}{2}\gamma$;
      $c_i^{(1)} = c_i^{(2)} = \cdots = c_i^{(\gamma)} = \frac{q-1}{2}$;
      $c_i^{(\gamma+1)} = a'_i$;
      $c_i^{(\gamma+2)} = \cdots = c_i^{(\tau)} = 0$;
    Else
      $a'_i = a_i + \frac{q-1}{2}\gamma$;
      $c_i^{(1)} = c_i^{(2)} = \cdots = c_i^{(\gamma)} = -\frac{q-1}{2}$;
      $c_i^{(\gamma+1)} = a'_i$;
      $c_i^{(\gamma+2)} = \cdots = c_i^{(\tau)} = 0$;
    EndIf
  EndFor
  $m' = a' * f_p + c^{(1)} * f_p + \cdots + c^{(\tau)} * f_p \pmod p$;
EndIf
Output plaintext m'.

Theorem 2. The improved NTRU cryptosystem does not reduce the security strength of the original NTRU.

Proof Our improved NTRU cryptosystem consists of the original encryption and the proposed improved decryption algorithm. Thus, proving this theorem is equivalent to showing that our improved decryption cannot reduce the security strength of the original NTRU. This can be analyzed from the following two aspects. First, similar to the original NTRU, our improved NTRU is also based on the shortest vector problem (SVP) in a lattice. Second, the improved decryption gets rid of the Gap failure and the Wrap failure to correctly recover the original message m without revealing any sensitive information as the decryptor computes the adjusting vectors and keeps them to itself, which implies that the improved decryption procedure is as secure as the original NTRU decryption. Therefore we claim that the improved NTRU cryptosystem does not reduce the security strength of the original NTRU.

An instance of decryption failure of the original NTRU cryptosystem is shown in Appendix. We also demonstrate that the improved NTRU cryptosystem can successfully decrypt the ciphertext for the same example.

5 The Proposed Scheme

Let B be the set of users that are eligible to access the outsourced sensitive data in the cloud. Assume that $U_i \in B$ is a user who needs to use the data. According to the access policy, at least $t-1$ other users in B should participate in the processes of verifying U_i's eligibility and obtaining the data for U_i. To achieve

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2016.2621106, IEEE Transactions on Big Data

this objective, we propose a secure and verifiable access control scheme for the big data storage in a cloud server in this section. Our scheme consists of the following four stages: i) Setup: the data owner initializes the system to generate the public and private keys via the improved NTRU cryptosystem; ii) Construction: the data owner generates a sub-key for each user in B, produces a message certificate for each message, and stores the data securely in the cloud server; iii) Reconstruction: the user U_i and $t-1$ other users in B mutually verify each other and work together to help U_i reconstruct the data; iv) Policy update: the cloud server instead of the data owner updates the encrypted data with a new policy if needed.

5.1 Setup

A data owner has a set of messages $S = \{S_1, S_2, ..., S_M\}$ for cloud storage, with M being the total number of messages in S. At the setup stage, the data owner generates its public key h and private key f according to the improved NTRU cryptosystem and then initializes the system according to the process presented in **Algorithm** 2.

Algorithm 2 Initialization

1: Chooses three integer parameters (N, p, q) satisfying $gcd(p, q) = 1$ and $q > p$;
2: Chooses four sets L_f, L_g, L_ϕ, L_m of polynomials of degree $N-1$ in R with integer coefficients;
3: Generates the key pair (f, h) according to the improved NTRU cryptosystem, where f is the private key and h is the corresponding public key;
4: Selects two one-way hash functions H_1 and H;
5: Publishes $\{p, q, h\}$ and the selection criteria of $\{L_\phi, L_m\}$.

5.2 Construction

The data owner constructs a sub-key for each legitimate user in B, generates a certificate for each message in S, and stores the encrypted data into the cloud server.

5.2.1 Sub-Key Construction

The data owner randomly generates t different integers $b_0, b_1, \cdots, b_{t-1}$, where $b_j \in Z[X]/(X^N - 1)$ for $j = 0, 1, \cdots, t-1$, and uses them as coefficients to construct the following $t-1$ degree polynomial $b(x)$:

$$b(x) = b_0 + \sum_{j=1}^{t-1} b_j x^j. \tag{10}$$

Each user U_i chooses a random integer r_i, encrypts r_i using the data owner's public key h to get the ciphertext $v_i = p\phi * h + r_i \pmod q$, and then sends $\{id_i, H(r_i), v_i\}$ to the data owner.

When the data owner receives the ciphertext v_i, it decrypts it using its private key f to get the plaintext r_i'. Then the data owner checks the following two conditions: i) $H(r_i') = H(r_i)$ for the user U_i; and ii) $r_i \neq r_\sigma$ for the user U_i and any user U_σ who has received a sub-key from the data owner. If condition i) cannot hold, the message is falsified during transmission and it could be sent again; if condition ii) cannot hold, the data owner requests the user U_i to choose a different secret number and repeat the procedure to compute a sub-key for U_i.

If the two conditions hold true for U_i, the data owner calculates $H(id_i||r_i)$ and generates the sub-key x_i for U_i using (11). Then the data owner securely broadcasts $\{id_i, x_i, H(id_i||r_i)\}$ to all the users in B.

$$x_i = b(r_i) = \sum_{j=0}^{t-1} b_j \cdot (r_i)^j. \tag{11}$$

5.2.2 Message Certificate Construction

In order to validate the users in B for their legitimacy of accessing the data in S and to prevent against the cheating behaviors of the users, the data owner computes a certificate for each message in S according to the following procedure: the data owner first randomly chooses the parameters $\phi \in L_\phi$ and $e = \{e_1, \cdots, e_j, \cdots, e_M\}$, where $e_j \in Z[X]/(X^N - 1)$; then it generates the certificate (e_j, d_j) for message $S_j \in S$ by (12), where $1 \leq j \leq M$,

$$d_j = p\phi * f + e_j \pmod q; \tag{12}$$

and finally, the data owner publishes $e = \{e_1, e_2, \cdots, e_j, \cdots, e_M\}$ to all users in B.

5.2.3 Data Encryption

Before outsourcing the data to the cloud server, the data owner computes a set of ciphertexts $K = \{k_1, k_2, \cdots, k_j, \cdots, k_M\}$ for the messages in $S = \{S_1, S_2, \cdots, S_j, \cdots, S_M\}$ according to (13), where k_j is the ciphertext of S_j.

$$k_j = S_j \oplus H(b_0 * d_j). \tag{13}$$

Then the data owner keeps $H_1(S_j)$ to itself and stores k_j, $j = 1, 2, \cdots, M$, in the cloud server.

The construction operations in Section 5.2.1, 5.2.2 and 5.2.3 are shown in **Algorithm** 3.

5.3 Message Reconstruction

Assume that a user $U_i \in B$ needs to get the message/data S_j. It downloads k_j from the cloud server and then seeks help from other users in B to decrypt k_j. This procedure consists of the following three stages: i) Exchange certificate computation: U_i gets S_j's message certificate from the data owner and computes an exchange certificate. ii) Certificate verification: other users in B and U_i mutually validate each other. iii) Message reconstruction: U_i reconstructs the message S_j.

5.3.1 Exchange Certificate Computation

U_i sends a request to the data owner to obtain S_j's message certificate d_j. Upon receiving this request, the data owner encrypts d_j using U_i's secret number r_i based on *AES* as shown in (14).

$$C_{d_j} = AES_{r_i}(d_j). \tag{14}$$

Upon receiving the ciphertext C_{d_j} from the data owner, U_i first decrypts C_{d_j} to obtain d_j. Then, it uses its sub-key x_i to compute the exchange certificate W_{ij} via (15) and sends W_{ij} to other users in B.

$$W_{ij} = x_i * d_j. \tag{15}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2016.2621106, IEEE Transactions on Big Data

Algorithm 3 Construction

The data owner generates a polynomial $b(x)$ according to (10) in Section 5.2.1;
For each user U_i in B
  Encrypts its selected number r_i using the data owner's public key h to get $v_i = p\phi * h + r_i \pmod q$;
  Computes $H'' = H(r_i)$ and sends $\{id_i, H'', v_i\}$ to the data owner;
  The data owner decrypts v_i to get r_i using Algorithm 1;
  If $H'' = H(r_i)$
    If $r_i \neq r_\sigma$ for any U_σ that has received a sub-key x_σ
      The data owner generates the sub-key x_i using (11);
      The data owner broadcasts $\{id_i, H(id_i||r_i), x_i\}$ to all users in B;
    Else
      The data owner requests U_i to choose a different random number r_i and then go back to step 3;
    EndIf
  Else
    The message is falsified;
    Retransmits $\{id_i, H'', v_i\}$ to the data owner;
    Go back to step 5;
  EndIf
EndFor
The data owner chooses parameters $\phi \in L_\phi$ and $e = (e_1, e_2, \cdots, e_j, \cdots, e_M)$;
For each message $S_j \in S$
  The data owner generates a message certificate (e_j, d_j) for S_j by (12);
  The data owner encrypts the data S_j by (13) to get k_j and computes $H_1(S_j)$;
  The data owner publishes e_j and stores k_j in the cloud server.
EndFor

5.3.2 Certificate Verification

When a user U_σ in B receives W_{ij}, it uses the public e_j to verify W_{ij} according to (16). If (16) holds, U_i is verified by U_σ, which implies that U_i is a valid user of the message S_j in U_σ's perspective. Then U_σ provides its $\{id_\sigma, r_\sigma\}$ to U_i.

$$W_{ij} = x_i * e_j. \tag{16}$$

When U_i receives U_σ's $\{id_\sigma, r_\sigma\}$, it computes $H' = H(id_\sigma||r_\sigma)$. Since U_i holds the $H(id||r)$ values of all users in B, it can verify whether the r_σ value provided by id_σ is correct. If r_σ does not pass the verification, U_σ's sub-key cannot be used by U_i for the reconstruction of S_j, which implies that U_σ has a cheating behavior in the perspective of U_i.

At least $t-1$ users in B should be able to verify the legitimacy of U_i for accessing S_j and these users must be verifiable by U_i in order for U_i to use their provided sub-keys for the reconstruction of S_j.

5.3.3 Message Reconstruction

When the user U_i obtains authorizations from $t-1$ legal users in B, it recovers the message S_j according to (17).

$$S_j = k_j \oplus H((\sum_{U_i \in B} x_i \prod_{U_\sigma \in B, U_\sigma \neq U_i} \frac{r_\sigma}{r_\sigma - r_i}) * d_j). \tag{17}$$

The processes of reconstruction and verification are summarized in **Algorithm** 4.

Algorithm 4 Reconstruction and Verification

The user U_i downloads k_j from the cloud server, and sends a request to the data owner to obtain d_j;
The data owner encrypts d_j with r_i to obtain ciphertext $C_{d_j} = AES_{r_i}(d_j)$, and then sends C_{d_j} to U_i;
U_i decrypts C_{d_j} to get d_j using its secret number r_i;
U_i computes the exchange certificate W_{ij} via (15), and sends W_{ij} to other users in B;
For each user U_σ in B **Do**
  Upon receiving W_{ij}, U_σ verifies W_{ij} by (16);
  If (16) holds
    U_σ sends its $\{id_\sigma, r_\sigma\}$ to U_i;
    Upon receiving $\{id_\sigma, r_\sigma\}$, U_i computes $H(id_\sigma||r_\sigma)$ to verify $\{id_\sigma, r_\sigma\}$;
    If $H(id_\sigma||r_\sigma)$ passes the verification
      U_σ participates in the reconstruction of S_j;
    EndIf
  EndIf
EndFor
If at least $t-1$ other users in B are able to participate in the recovery of S_j
  U_i can reconstruct S_j via (17);
EndIf

5.4 Policy Update

The data owner can dynamically control the access of its data by updating the access policy defined for the data. Intuitively, as the ciphertext of the data is stored in the cloud server, the data owner needs to download the ciphertext, decrypt it to get the plaintext, and then re-encrypt the plaintext according to the new access policy. This is the approach adopted by all existing research. In this subsection, we consider to update the encrypted data at the cloud server based on the new access policy, which differs significantly than the traditional approaches.

The new access policy is defined by $b'(x) = b'_0 + \sum_{j=0}^{t'-1} b'_j x_j$, where $b'_0, b'_1, \cdots, b'_{t'-1}$ are the t' coefficients of the new polynomial $b'(x)$ selected by the data owner. The data owner needs to execute the procedures in Section 5.2.1 and Section 5.2.2 to generate a new sub-key x'_i for each user $U_i \in B$, and compute a new message certificate d'_j for each $S_j \in S$. Then the data owner generates the intermediate value k^j_{iv} based on the values of b_0, d_j, b'_0, and d'_j by (18):

$$k^j_{iv} = H(b_0 * d_j) \oplus H(b'_0 * d'_j). \tag{18}$$

The data owner D sends k^j_{iv} to the cloud server, which will update the encrypted data k_j via (19).

$$k'_j = k_j \oplus k^j_{iv}. \tag{19}$$

After that the cloud server sends k'_j back to the data owner to verify whether the server has successfully update the ciphertext constructed from the new access policy. This procedure can be done by the data owner according to (20) .

$$H_1(S_j) = H_1(k'_j \oplus H(b'_0 * d'_j)). \tag{20}$$

Remarks: The cloud server can update the ciphertext when a new access policy is specified by the data owner, and this update

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2016.2621106, IEEE Transactions on Big Data

can be verified by the data owner; meanwhile, each user in B can easily update its secret number and the corresponding sub-key.

6 Analysis of The Proposed Scheme

In this section, we conduct a rigorous analysis on the correctness, security strengths, and computational complexity of the proposed scheme.

6.1 Correctness

Theorem 3. A user $U_i \in B$ who needs the data S_j and each of the other users in B can verify each other to help U_i recover the message S_j.

Proof After receiving d_j from the data owner, U_i generates its exchange certificate $W_{ij} = x_i * d_j$ and sends W_{ij} to the other users in *B*. Any user $U_\sigma \in B$ can verify the exchange certificate W_{ij} by computing $x_i * e_j$ and checking whether it is equal to W_{ij}. If W_{ij} does equal $x_j * e_j$, U_σ concludes that U_i holds the correct message certificate d_j, which implies that U_i is the legitimate user of S_j. Such a verification process is supported by the following derivation:

$$\begin{aligned} W_{ij} &= x_i * d_j \\ &= x_i * (p\phi * f + e_j) \bmod p \\ &= x_i * e_j \end{aligned}$$

If U_σ verifies U_i to be a legitimate user of S_j, it sends its $\{id_\sigma, r_\sigma\}$ to U_i. After receiving $\{id_\sigma, r_\sigma\}$, U_i calculates $H' = H(id_\sigma||r_\sigma)$ and checks against its local copy of $H(id_\sigma||r_\sigma)$ - recall that the data owner broadcasts the $H(id||r)$ value of each user in B to all other users in B during the sub-key construction stage. If they are equal, U_σ provides the correct r_σ for the recovery of S_j; otherwise, U_σ cheats and it's r_σ value will not be used for the reconstruction of S_j.

Theorem 4. A user U_i in *B* can reconstruct a message S_j via (17) if there exist at least $t-1$ other users in B satisfying the following two conditions: i) each of the $t-1$ users can verify that U_i is a legitimate user of S_j; and ii) each of the $t-1$ users can be verified by U_i to provide the correct sub-key information.

Proof Without loss of generality, we denote by $U_1, U_2, ..., U_{t-1}$ the $t-1$ users with each being able to verify the legitimacy of U_i and each being verified by U_i for providing the correct r values. This implies that U_i holds the correct message certificate d_j and the correct $\{id_\sigma, r_\sigma\}$ of user U_σ, where $\sigma = 1, 2, \cdots, t-1$. Thus U_i can obtain the following result:

$$\begin{aligned} & (\sum_{U_i \in B} x_i \prod_{U_\sigma \in B, U_\sigma \neq U_i} \frac{r_\sigma}{r_\sigma - r_i}) * d_j \\ = \; & (\sum_{U_i \in B} x_i \prod_{U_\sigma \in B, U_\sigma \neq U_i} \frac{r_\sigma}{r_\sigma - r_i}) * d_j \\ = \; & (\sum_{U_i \in B} b(r_i) \prod_{U_\sigma \in B, U_\sigma \neq U_i} \frac{r_\sigma}{r_\sigma - r_i}) * d_j. \end{aligned}$$

Since

$$\begin{aligned} b_0 * d_j &= b(0) * d_j \\ &= (\sum_{U_i \in B} b(r_i) \prod_{U_\sigma \in B, U_\sigma \neq U_i} \frac{r_\sigma}{r_\sigma - r_i}) * d_j, \end{aligned}$$

We have $b_0 * d_j = (\sum_{U_i \in B} x_i \prod_{U_\sigma \in B, U_\sigma \neq U_i} \frac{r_\sigma}{r_\sigma - r_i}) * d_j$. Thus $S_j = k_j \oplus H(b_0 * d_j)$, which implies that the message S_j can be correctly reconstructed by U_i via (17).

Theorem 5. The cloud server instead of the data owner can correctly update the access policy of a message without any decryption process. The data owner can verify the correctness of the update procedure.

Proof According to Section 5.4, the data owner re-generates the t' integers $\{b'_0, b'_1, ..., b'_{t'-1}\}$ that are used as the coefficients of the polynomial $b'(x)$, the new access policy, and computes the intermediate value $k^j_{iv} = H(b_0 * d_j) \oplus H(b'_0 * d'_j)$ based on the values of b_0, d_j, b'_0, and d'_j by (18). Then the data owner sends the intermediate value k^j_{iv} to the cloud server, which updates the encrypted data in the cloud via (19) as follows.

$$\begin{aligned} k'_j &= k_j \oplus k^j_{iv} \\ &= S_j \oplus H(b_0 * d_j) \oplus k^j_{iv} \\ &= S_j \oplus H(b_0 * d_j) \oplus H(b_0 * d_j) \oplus H(b'_0 * d'_j) \\ &= S_j \oplus H(b'_0 * d'_j) \end{aligned}$$

Then, the cloud server sends k'_j back to the data owner who can verify the correctness of the policy update result via (20).

6.2 Security Analysis

The security of our scheme is based on the NTRU public key cryptosystem [26] and the Shamir's secret sharing scheme [45]. In this section, we analyze the security strength of our proposed scheme by examining how it can defend against several major attacks.

6.2.1 Attack Resistance

Attack 1: *A plotter E (not in B) may try to reveal the message S_j from k_j without knowing any information about t users.*

Analysis: If the plotter E intends to recover the message S_j from k_j by computing $k_j \oplus H((\sum_{U_i \in B} x_i \prod_{U_i \in B, U_\sigma \neq U_i} \frac{r_\sigma}{r_\sigma - r_i}) * d_j)$, it has to (i) get d_j from the data owner of S_j and pass the verification by at least t users in B; and (ii) obtain the secret numbers $\{r_1, r_2, \cdots, r_t\}$ from the t users. To get d_j from the data owner, E needs a shared secret with the data owner to prove that it is a legal user of S_j; to get r_σ from U_σ, E needs to pass the verification by U_σ, which again needs d_j for the computation of the exchange certificate. As sharing a secret with the data owner is a proof of legitimacy, it is impossible for E to get d_j; thus the plotter *E* cannot recover S_j.

Attack 2: *The users in B may cheat to fail data recovery.*

Analysis: The cheating behaviors can be prevented during data recovery according to **Theorem** 3. Let U_i be the user who would like to get S_j. (i) If U_i cheats, it provides a wrong exchange certificate W_{ij}, which can be identified by any other honest user in B when verifying the legitimacy of U_i on S_j; (ii) If U_σ provides an incorrect secret number r_σ to a legal user U_i for the recovery of S_j, it can be detected because U_i holds a local copy of the correct $H(id_\sigma||r_\sigma)$, which is broadcast by the data owner during the sub-key construction process. Therefore, our proposed scheme can prevent users from cheating.

Attack 3: *The users in B may collude.*

Analysis: Our scheme can resist the following type of collusion attack: $t-1$ or less number of users in B collude to recover

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2016.2621106, IEEE Transactions on Big Data

!tb

TABLE 2
The comparisons among [15]'s Scheme-I and Scheme-II, [16] scheme and our scheme.

Properties	Scheme-I [15]	Scheme-II [15]	[16]	Our scheme
Prevent users from cheating by providing fake information	Yes	Yes	Yes	Yes
Prevent the collusion of $t-1$ or fewer users in message recovery	Yes	Yes	Yes	Yes
Methods for establishing a secure channel	RSA	RSA	RSA	AES
Resist quantum computing attacks	No	No	No	Yes
Resist collusion attack	Yes	Yes	Yes	Yes
Dynamically update the access policy	No	No	No	Yes
Protect previously encrypted data	No	No	No	Yes
Dynamically update the ciphertext for user join and leave activities	No	No	No	Yes
Computational complexity of Verification	$O(N^3)$	$O(N^3)$	$O(N^3)$	$O(N \log N)$
Computational complexity of Reconstruction	$O(N)$	$O(N)$	$O(N)$	$O(N \log^2 N)$
NP-Hard problems for security guarantee	DFL	DFL	DFL	SVP

the message S_j. From (13), we know that the users should obtain b_0 and d_j in order to decipher the message. Nevertheless, since the security of our proposed scheme is guaranteed by (t, n)-threshold secret sharing, it is impossible for any $t-1$ or less number of users to obtain b_0.

Attack 4: *An attacker intends to forge W_{ij}.*

Analysis: In order to forge W_{ij}, which is defined to be $x_i * (p\phi * f + e_j)$ according to (12) and (15), the attacker has to obtain f, the private key of the data owner. However, it is impossible for the attacker to get f as f is generated according to the NTRU cryptosystem, which is proved to be secure.

Theorem 6. The data owner can only decrypt its own ciphertexts stored in the cloud server.

Proof Although a data owner can obtain any ciphertext from the cloud server, it cannot decrypt the ciphertext encrypted by other data owners. This claim can be proved as follows.

Without loss of generality, assume the data owner D_i downloads the ciphertext k'_{D_j} generated by the data owner D_j, where $k'_{D_j} = S'_{D_j} \oplus H(b_0^{D_j} * d_j^{D_j})$. To obtain the plaintext S'_{D_j}, D_i needs to compute $H(b_0^{D_j} * d_j^{D_j})$. However, the values of $b_0^{D_j}$ is securely chosen by D_j, which can be recovered only by t or more legal users of S'_{D_j} based on secret-sharing, and the value of $d_j^{D_j}$ is computed by D_j from $e_j^{D_j}$, which can be securely obtained from D_j only when a legal user of S'_{D_j} requests from D_j and proves its legitimacy to D_j via the shared secret between the user and D_j; therefore D_i cannot figure out the correct $b_0^{D_j}$ and $d_j^{D_j}$ as it cannot prove its legitimacy with respect to the data S'_{D_j}. Thus a data owner can only decrypt its own ciphertexts stored in the cloud server.

6.3 Performance Analysis

6.3.1 Computational Complexity

In this section, we analyze the computational complexity of our proposed scheme and compare it with those of the three schemes proposed in [15] and [16]. Note that our analysis will be focused on the processes of certificate verification and message reconstruction as other processes incur very low computational overhead that can be ignored. The two schemes Scheme-I and Scheme-II proposed in [15] and the scheme presented in [16] are briefly introduced in Section 2.2.

Certificate Verification: The verification processes of Scheme-I and Scheme-II in [15] and the scheme in [16] are based on the RSA cryptosystem. Since the computational complexity of the RSA encryption is $O(N^3)$, the computational complexities of these three schemes are also $O(N^3)$. Our proposed scheme is based on the NTRU cryptosystem whose security is based on the SVP hardness in lattice. Its encryption process only requires add and shift operations without involving any multiplication. By employing the Fast Fourier transform (FFT), the encryption and decryption of NTRU can be performed within $O(N \log N)$.

Message Reconstruction: In [15] and [16], recovering a plaintext message involves a linear computational complexity of $O(N)$. In our scheme, the message is recovered by a Lagrange interpolation polynomial with a computational complexity of $O(N \log^2 N)$, which is slightly larger than those of the schemes in [15] and [16].

Remark: The schemes in [15] and [16] and our proposed scheme all require a secure channel to transmit the messages. However, the secure channels of [15] and [16] are established by the asymmetric cryptosystem *RSA*, while our scheme employs the symmetric cryptosystem *AES*, which has a much lower computational overhead.

6.3.2 Dynamic Update

Our proposed scheme allows the dynamic update (refresh, insert, delete) of the information regarding the users and the messages.

- **Insert a New Message.** When a data owner needs to store a new message S_{new} into the cloud server, it generates the certificate (e_{new}, d_{new}) for S_{new} by (12), encrypts S_{new} to obtain the ciphertext k_{new} via (13), computes $H_1(S_{new})$, and then stores k_{new} in the cloud server.
- **Delete an Old Message.** When a data owner needs to delete a message S_{del}, it sends an authenticated request to the cloud server. Upon receiving the request, the cloud server deletes the ciphertext of S_{del}.
- **A New User Joins.** When a new user U_{new} joins the system, the data owner first verifies its legitimacy of accessing its data; then it computes the sub-key x_{new} for the legal user U_{new} according to (11), and publishes x_{new} to all users in B.
- **Remove a User.** When a data owner wants to prevent a user in B from accessing its data, it updates its access policy according to the policy update procedure proposed in Section 5.4 and computes a new sub-key for each of the other users in B.

6.3.3 Summary

In Table 2, we present a comprehensive comparison study over Scheme-I and Scheme-II in [15], the scheme in [16], and our

proposed scheme in terms of attack resistance, security, policy update, and so on. From Table 2, one can see that the significant advantages of our scheme over the other three schemes include: quantum computing attack resistance, dynamic policy update, and lower computational complexity.

6.4 Discussions

In our scheme, a data owner maintains the access right to its encrypted data. The advantages of such a policy are listed as follows:

- The data owner splits the access right of the encrypted data into n pieces, with each legitimate user holding one piece of the access right. This can effectively reduce the risk of information leakage in big data storage.
- If a user U_i wants to decrypt the data owner's encrypted data S_j, its legitimacy must be verified by the data owner (by securely and successfully retrieving the message certificate d_j from the data owner) and at least $t-1$ other legitimate users of the data (via the exchange certificate); the information provided by the $t-1$ other users for the message recovery also must be verified by U_i to prevent the user from providing fake information. This verification process can ensure the legality of U_i to access S_j.
- The cloud sever only stores the outsourced ciphertext data - it cannot obtain the data owner's original plaintext data. In addition, the data owner updates the access policy while the cloud server updates the encrypted data. No data owner can access the plaintext data outsourced by other data owners in the cloud server.

Compared with RSA, at an equivalent cryptographic strength, NTRU performs the costly encryption and decryption operations at a much faster speed, and its implementations in both hardware and software are easier and require a smaller size of memory; therefore NTRU can be employed in applications involving devices with limited computation power and storage such as mobile devices and smart-cards. On the other hand, when our scheme is utilized in real world applications, an adequate threshold t needs to be selected by the data owner before deployment because an unsuitable threshold t would decrease the security of the data if t is too small or have a harmful effect on the flexibility of access control if t is too large. Nevertheless, the selection of a right value for t is not easy. One possible remedy approach is to pick up an initial value of t based on experience and adjust it via access policy update as time goes.

7 Conclusion and Future Works

In this paper, we first propose an improved NTRU cryptosystem to overcome the decryption failures of the original NTRU and then present a secure and verifiable access control scheme based on the improved NTRU to protect the outsourced big data stored in a cloud. Our scheme allows the data owner to dynamically update the data access policy and the cloud server to successfully update the corresponding outsourced ciphertext to enable efficient access control over the big data in the cloud. It also provides a verification process for a user to validate its legitimacy of accessing the data to both the data owner and $t-1$ other legitimate users and the correctness of the information provided by the $t-1$ other users for plaintext recovery. The security of our proposed scheme is guaranteed by those of the NTRU cryptosystem and the (t, n)-threshold secret sharing. We have rigorously analyzed the correctness, security strength, and computational complexity of our proposed scheme.

Designing a secure, privacy preserving, and practical scheme for big data storage in a cloud is an extremely challenging problem. In our future research, we will further improve our scheme by combining the (t, n)-threshold secret sharing with attribute-based access control, which involves an access structure that can place various requirements for a user to decrypt an outsourced ciphertext data in the cloud. Meanwhile, we will investigate the security problems when a data owner outsources its data to multi-cloud servers and consider an attribute-based access structure that can be dynamically updated, which is more applicable for practical scenarios in big data storage.

Acknowledgment

This research was partially supported by the National Science Foundation of the US under grants CNS-1318872, CCF-1442642 and IIS-1343976, and the National Natural Science Foundation of China under grants 61672119, 61373027, 61472044, 61272475, 61571049, 61472403, and 61672321, and the Fundamental Research Funds for the Central Universities (No.2014KJJCB32).

References

[1] M. A. Beyer and D. Laney, "The importance of big data: a definition," *Stamford, CT: Gartner*, 2012.
[2] V. Marx, "Biology: The big challenges of big data," *Nature*, vol. 498, no. 7453, pp. 255–260, 2013.
[3] G. P. Consortium *et al.*, "A map of human genome variation from population-scale sequencing," *Nature*, vol. 467, no. 7319, pp. 1061–1073, 2010.
[4] A. Sahai and B. Waters, "Fuzzy identity-based encryption," *Advances in Cryptology–EUROCRYPT 2005*, pp. 457–473, 2005.
[5] C. Hu, F. Zhang, X. Cheng, X. Liao, and D. Chen, "Securing communications between external users and wireless body area networks," in *Proceedings of the 2nd ACM workshop on Hot topics on wireless network security and privacy*. ACM, 2013, pp. 31–36.
[6] C. Hu, H. Li, Y. Huo, T. Xiang, and X. Liao, "Secure and efficient data communication protocol for wireless body area networks," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 2, no. 2, pp. 94–107, 2016.
[7] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM conference on Computer and communications security*. ACM, 2006, pp. 89–98.
[8] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," *Public Key Cryptography–PKC 2011*, pp. 53–70, 2011.
[9] C. Hu, N. Zhang, H. Li, X. Cheng, and X. Liao, "Body area network security: a fuzzy attribute-based signcryption scheme," *IEEE journal on selected areas in communications*, vol. 31, no. 9, pp. 37–46, 2013.
[10] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," *Advances in Cryptology–EUROCRYPT 2011*, pp. 568–588, 2011.
[11] C. Hu, X. Cheng, Z. Tian, J. Yu, K. Akkaya, and L. Sun, "An attribute-based signcryption scheme to secure attribute-defined multicast communications," in *SecureComm 2015*. Springer, 2015, pp. 418–435.
[12] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Advances in cryptology*. Springer, 1985, pp. 47–53.
[13] M. Dehkordi and S. Mashhadi, "An efficient threshold verifiable multi-secret sharing," *Computer Standards & Interfaces*, vol. 30, no. 3, pp. 187–190, 2008.
[14] Z. Eslami and J. Z. Ahmadabadi, "A verifiable multi-secret sharing scheme based on cellular automata," *Information Sciences*, vol. 180, no. 15, pp. 2889–2894, 2010.
[15] M. H. Dehkordi and S. Mashhadi, "New efficient and practical verifiable multi-secret sharing schemes," *Information Sciences*, vol. 178, no. 9, pp. 2262–2274, 2008.
[16] J. Zhao, J. Zhang, and R. Zhao, "A practical verifiable multi-secret sharing scheme," *Computer Standards & Interfaces*, vol. 29, no. 1, pp. 138–141, 2007.

[17] C. Hu, X. Liao, and X. Cheng, "Verifiable multi-secret sharing based on LFSR sequences," *Theoretical Computer Science*, vol. 445, 2012.

[18] C. Hu, X. Liao, and D. Xiao, "Secret image sharing based on chaotic map and chinese remainder theorem," *International Journal of Wavelets, Multiresolution and Information Processing*, vol. 10, no. 03, 2012.

[19] K. Yang, X. Jia, K. Ren, B. Zhang, and R. Xie, "Dac-macs: Effective data access control for multiauthority cloud storage systems," *Information Forensics and Security, IEEE Transactions on*, vol. 8, no. 11, pp. 1790–1801, 2013.

[20] K. Yang and X. Jia, "Expressive, efficient, and revocable data access control for multi-authority cloud storage," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 25, no. 7, pp. 1735–1744, 2014.

[21] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Security and Privacy, 2007. SP'07. IEEE Symposium on*. IEEE, 2007, pp. 321–334.

[22] A. Córcoles, E. Magesan, S. J. Srinivasan, A. W. Cross, M. Steffen, J. M. Gambetta, and J. M. Chow, "Demonstration of a quantum error detection code using a square lattice of four superconducting qubits," *Nature communications*, vol. 6, pp. 1–10, 2015.

[23] O. Regev, "New lattice-based cryptographic constructions," *Journal of the ACM (JACM)*, vol. 51, no. 6, pp. 899–942, 2004.

[24] D. Micciancio, "Lattice-based cryptography," in *Post-Quantum Cryptography*. Springer, 2009, pp. 147–191.

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

[26] J. Hoffstein, J. Pipher, and J. Silverman, "NTRU: A ring-based public key cryptosystem," in *Algorithmic number theory: third international symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998: proceedings*, vol. 1423. Springer Verlag, 1998, pp. 267–288.

[27] N. Cryptosystems, "The NTRU public key cryptosystem-a tutorial," 1998.

[28] A. Langlois, D. Stehlé, and R. Steinfeld, "Gghlite: More efficient multilinear maps from ideal lattices," in *Advances in Cryptology–EUROCRYPT 2014*. Springer, 2014, pp. 239–256.

[29] S. Garg, C. Gentry, and S. Halevi, "Candidate multilinear maps from ideal lattices." in *Eurocrypt*, vol. 7881. Springer, 2013, pp. 1–17.

[30] European Telecommunications Standards Institute, "Quantum safe cryptography and security-an introduction, benefits, enablers and challenges," European Telecommunications Standards Institute, White Paper, June 2015. [Online]. Available: https://www.etsi.org/images/files/ETSIWhitePapers/QuantumSafeWhitepaper.pdf

[31] IBM, "IBM storage solutions for banking," http://www-03.ibm.com/systems/storage/solutions/industries/banking.html.

[32] A. Mora, Y. Chen, A. Fuchs, A. Lane, R. Lu, P. Manadhata *et al.*, "Expanded top ten big data security and privacy challenges," *Cloud Security Alliance*, 2013.

[33] L. Wei, H. Zhu, Z. Cao, X. Dong, W. Jia, Y. Chen, and A. V. Vasilakos, "Security and privacy for storage and computation in cloud computing," *Information Sciences*, vol. 258, pp. 371–386, 2014.

[34] H. Cheng, C. Rong, K. Hwang, W. Wang, and Y. Li, "Secure big data storage and sharing scheme for cloud tenants," *China Communications*, vol. 12, no. 6, pp. 106–115, 2015.

[35] J. Baek, Q. H. Vu, J. K. Liu, X. Huang, and Y. Xiang, "A secure cloud computing based framework for big data information management of smart grid," *IEEE transactions on cloud computing*, vol. 3, no. 2, pp. 233–244, 2015.

[36] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford University, 2009.

[37] S. Kamara and K. Lauter, "Cryptographic cloud storage," in *International Conference on Financial Cryptography and Data Security*. Springer, 2010, pp. 136–149.

[38] A. Hamlin, N. Schear, E. Shen, M. Varia, S. Yakoubov, and A. Yerukhimovich, "Cryptography for big data security," in *Big Data: Storage, Sharing, and Security*, F. Hu, Ed. CRC Press, 2016, ch. 10, pp. 241–288.

[39] G. Zhuo, Q. Jia, L. Guo, M. Li, and P. Li, "Privacy-preserving verifiable set operation in big data for cloud-assisted mobile crowdsourcing," *to appear in IEEE Internet of Things Journal*, 2016.

[40] L. Guo, Y. Fang, M. Li, and P. Li, "Verifiable privacy-preserving monitoring for cloud-assisted mhealth systems," in *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 1026–1034.

[41] S. Salinas, X. Chen, J. Ji, and P. Li, "A tutorial on secure outsourcing of large-scale computations for big data," *IEEE Access*, vol. 4, pp. 1406–1416, 2016.

[42] V. C. Hu, D. Ferraiolo, and D. R. Kuhn, *Assessment of access control systems*. US Department of Commerce, National Institute of Standards and Technology, 2006.

[43] K. Yang and X. Jia, "Attributed-based access control for multi-authority systems in cloud storage," in *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*. IEEE, 2012, pp. 536–545.

[44] T. Jung, X.-Y. Li, Z. Wan, and M. Wan, "Privacy preserving cloud data access with multi-authorities," in *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013, pp. 2625–2633.

[45] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[46] A. Sahai, H. Seyalioglu, and B. Waters, "Dynamic credentials and ciphertext delegation for attribute-based encryption," in *Advances in Cryptology–CRYPTO 2012*. Springer, 2012, pp. 199–217.

[47] G. Blakley, "Safeguarding cryptographic keys," *Proc. of the National Computer Conference1979*, vol. 48, pp. 313–317, 1979.

[48] J. H. Silverman and W. Whyte, "Estimating decryption failure probabilities for ntruencrypt," NTRU Cryptostystems, Tech. Rep. Report #018, Version 1, May 2003. [Online]. Available: http://grouper.ieee.org/groups/1363/lattPK/submissions/NTRUTech018.pdf

APPENDIX

In this appendix, we give an instance of the NTRU decryption failure of the original NTRU cryptosystem. Meanwhile, we demonstrate how our improved NTRU can correct the failure. The NTRU system parameters are $N = 107, p = 3, q = 64$.

Secret key:

$$f = \begin{bmatrix} -1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & -1 & -1 \\ -1 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & -1 & 0 & -1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & & & \end{bmatrix}$$

$$g = \begin{bmatrix} 0 & 0 & -1 & 0 & 0 & 1 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & -1 & 0 & & & \end{bmatrix}$$

$$f_p = \begin{bmatrix} 2 & 2 & 1 & 1 & 1 & 0 & 2 & 0 & 2 & 0 \\ 2 & 1 & 0 & 2 & 2 & 0 & 0 & 2 & 2 & 1 \\ 1 & 1 & 1 & 2 & 1 & 1 & 0 & 1 & 2 & 2 \\ 2 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 1 & 2 \\ 1 & 1 & 2 & 2 & 0 & 1 & 2 & 1 & 1 & 0 \\ 0 & 2 & 2 & 2 & 2 & 1 & 0 & 0 & 2 & 2 \\ 2 & 0 & 0 & 1 & 0 & 0 & 2 & 0 & 1 & 1 \\ 0 & 1 & 2 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 2 & 1 & 2 & 1 & 1 & 0 & 2 & 2 & 1 & 0 \\ 1 & 0 & 1 & 2 & 0 & 0 & 2 & 0 & 0 & 1 \\ 1 & 1 & 2 & 2 & 2 & 2 & 1 & & & \end{bmatrix}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2016.2621106, IEEE Transactions on Big Data

$$f_q = \begin{bmatrix} 59 & 5 & 41 & 10 & 34 & 28 & 31 & 13 & 5 & 36 \\ 0 & 21 & 36 & 28 & 18 & 41 & 46 & 13 & 38 & 5 \\ 31 & 7 & 1 & 49 & 26 & 37 & 27 & 12 & 14 & 58 \\ 60 & 9 & 23 & 7 & 17 & 41 & 6 & 29 & 38 & 12 \\ 54 & 39 & 52 & 9 & 36 & 19 & 47 & 29 & 44 & 26 \\ 40 & 30 & 52 & 60 & 25 & 52 & 10 & 61 & 7 & 7 \\ 48 & 21 & 14 & 46 & 55 & 51 & 7 & 16 & 49 & 29 \\ 27 & 3 & 60 & 10 & 12 & 8 & 24 & 61 & 7 & 39 \\ 2 & 54 & 47 & 34 & 4 & 47 & 52 & 17 & 41 & 60 \\ 25 & 44 & 20 & 62 & 55 & 33 & 52 & 58 & 10 & 63 \\ 63 & 7 & 58 & 42 & 52 & 44 & 49 & & & \end{bmatrix}$$

Public Key:

$$h = \begin{bmatrix} 28 & 63 & 13 & 41 & 31 & 6 & 30 & 54 & 45 & 37 \\ 63 & 41 & 33 & 58 & 32 & 62 & 13 & 62 & 18 & 59 \\ 44 & 25 & 12 & 40 & 49 & 7 & 41 & 8 & 2 & 31 \\ 10 & 7 & 10 & 10 & 34 & 47 & 4 & 7 & 27 & 47 \\ 32 & 46 & 58 & 54 & 36 & 48 & 38 & 51 & 7 & 45 \\ 26 & 22 & 35 & 45 & 53 & 9 & 19 & 63 & 50 & 38 \\ 5 & 52 & 14 & 25 & 60 & 61 & 50 & 37 & 20 & 2 \\ 36 & 11 & 20 & 36 & 18 & 11 & 54 & 23 & 36 & 6 \\ 45 & 26 & 11 & 20 & 41 & 45 & 59 & 31 & 21 & 13 \\ 9 & 8 & 48 & 25 & 9 & 57 & 45 & 0 & 29 & 60 \\ 12 & 50 & 14 & 22 & 1 & 55 & 33 & & & \end{bmatrix}$$

Plaintext m:

$$m = \begin{bmatrix} 1 & 2 & 1 & 2 & 0 & 2 & 0 & 0 & 2 & 0 \\ 1 & 2 & 0 & 2 & 2 & 2 & 0 & 2 & 1 & 0 \\ 0 & 2 & 2 & 1 & 2 & 2 & 2 & 2 & 0 & 2 \\ 2 & 2 & 1 & 2 & 2 & 2 & 2 & 0 & 0 & 2 \\ 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 2 & 2 \\ 2 & 1 & 0 & 2 & 0 & 0 & 2 & 1 & 0 & 2 \\ 2 & 2 & 0 & 0 & 0 & 2 & 0 & 2 & 2 & 0 \\ 2 & 2 & 1 & 0 & 2 & 1 & 2 & 0 & 0 & 0 \\ 2 & 0 & 0 & 2 & 2 & 2 & 2 & 1 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 & 0 & 0 & 2 & 2 & 2 \\ 1 & 2 & 2 & 0 & 2 & 0 & 2 & & & \end{bmatrix}$$

Polynomial ϕ:

$$\phi = \begin{bmatrix} 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & & & \end{bmatrix}$$

Ciphertext e:

$$e = \begin{bmatrix} 29 & 45 & 0 & 46 & 9 & 58 & 40 & 23 & 3 & 13 \\ 26 & 11 & 62 & 31 & 23 & 4 & 9 & 18 & 12 & 22 \\ 41 & 46 & 24 & 0 & 21 & 63 & 43 & 39 & 32 & 1 \\ 27 & 29 & 29 & 47 & 39 & 10 & 32 & 60 & 19 & 48 \\ 32 & 22 & 42 & 28 & 19 & 15 & 10 & 18 & 54 & 50 \\ 29 & 42 & 35 & 57 & 57 & 29 & 44 & 26 & 28 & 57 \\ 38 & 58 & 33 & 3 & 47 & 28 & 21 & 29 & 41 & 1 \\ 23 & 4 & 62 & 53 & 17 & 57 & 45 & 47 & 59 & 39 \\ 42 & 22 & 18 & 23 & 61 & 60 & 11 & 54 & 31 & 56 \\ 36 & 32 & 58 & 5 & 59 & 25 & 12 & 47 & 26 & 8 \\ 22 & 46 & 56 & 49 & 20 & 60 & 51 & & & \end{bmatrix}$$

The decryption result of the original NTRU cryptosystem m':

$$m' = \begin{bmatrix} 2 & 0 & 1 & 2 & 1 & 2 & 0 & 0 & 2 & 2 \\ 2 & 1 & 2 & 0 & 0 & 2 & 2 & 0 & 0 & 2 \\ 0 & 2 & 0 & 2 & 0 & 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 1 & 2 & 1 & 2 & 2 & 1 & 0 & 1 \\ 2 & 0 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 0 & 2 & 2 & 2 & 2 & 1 & 1 \\ 2 & 1 & 0 & 2 & 1 & 2 & 0 & 0 & 2 & 0 \\ 1 & 1 & 0 & 1 & 0 & 2 & 0 & 2 & 1 & 1 \\ 1 & 2 & 2 & 2 & 0 & 2 & 0 & 1 & 0 & 1 \\ 2 & 0 & 0 & 2 & 2 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 2 & 2 & 2 & 0 & & & \end{bmatrix}$$

Obviously, $m' \neq m$, which indicates that a decryption failure occurs in the original NTRU decryption. Next we analyze the cause of the decryption failure. The process of decryption based on the original NTRU is as follows:

$$a \overset{e \times f}{=} \begin{bmatrix} 1 & 11 & 1 & -4 & 5 & 2 & -5 & -4 & 6 & 10 \\ 12 & 0 & 4 & -10 & 1 & -3 & -7 & -1 & 3 & -1 \\ 1 & 6 & -5 & -9 & -9 & 1 & 6 & 1 & 3 & 8 \\ 3 & -15 & 2 & 2 & -1 & 14 & 1 & -4 & -15 & -2 \\ -7 & -6 & 3 & -8 & -6 & -8 & 6 & 1 & 12 & 5 \\ -2 & -5 & -4 & 8 & 0 & -8 & 0 & -10 & 8 & -9 \\ 0 & -5 & -5 & -6 & 2 & 7 & -6 & 10 & 2 & -2 \\ 11 & -3 & 12 & 7 & 8 & -8 & 8 & \mathbf{33} & 8 & 7 \\ 15 & 6 & 5 & -2 & -3 & -1 & 4 & -3 & 10 & -6 \\ 1 & 11 & 10 & 1 & 6 & 6 & 5 & 6 & 4 & -1 \\ -2 & 6 & -4 & 5 & 10 & 10 & -4 & & & \end{bmatrix}$$

One can see that one element of a is 33, which is larger than $\frac{q}{2} = 32$ and causes the decryption failure. Now we utilize the improved NTRU cryptosystem to decrypt e. One can obtain a' as follows.

$$a' = \begin{bmatrix} 1 & 11 & 1 & -4 & 5 & 2 & -5 & -4 & 6 & 10 \\ 12 & 0 & 4 & -10 & 1 & -3 & -7 & -1 & 3 & -1 \\ 1 & 6 & -5 & -9 & -9 & 1 & 6 & 1 & 3 & 8 \\ 3 & -15 & 2 & 2 & -1 & 14 & 1 & -4 & -15 & -2 \\ -7 & -6 & 3 & -8 & -6 & -8 & 6 & 1 & 12 & 5 \\ -2 & -5 & -4 & 8 & 0 & -8 & 0 & -10 & 8 & -9 \\ 0 & -5 & -5 & -6 & 2 & 7 & -6 & 10 & 2 & -2 \\ 11 & -3 & 12 & 7 & 8 & -8 & 8 & \mathbf{31} & 8 & 7 \\ 15 & 6 & 5 & -2 & -3 & -1 & 4 & -3 & 10 & -6 \\ 1 & 11 & 10 & 1 & 6 & 6 & 5 & 6 & 4 & -1 \\ -2 & 6 & -4 & 5 & 10 & 10 & -4 & & & \end{bmatrix}$$

The adjusted vector c is shown below:

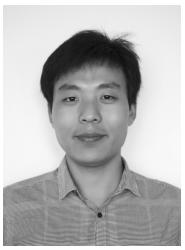
$$c = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & & & \end{bmatrix}$$

And the decrypted result is:

$$m'' = a' \times f_p + c \times f_p = \begin{bmatrix} 1 & 2 & 1 & 2 & 0 & 2 & 0 & 0 & 2 & 0 \\ 1 & 2 & 0 & 2 & 2 & 2 & 0 & 2 & 1 & 0 \\ 0 & 2 & 2 & 1 & 2 & 2 & 2 & 2 & 0 & 2 \\ 2 & 2 & 1 & 2 & 2 & 2 & 2 & 0 & 0 & 2 \\ 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 2 & 2 \\ 2 & 1 & 0 & 2 & 0 & 0 & 2 & 1 & 0 & 2 \\ 2 & 2 & 0 & 0 & 0 & 2 & 0 & 2 & 2 & 0 \\ 2 & 2 & 1 & 0 & 2 & 1 & 2 & 0 & 0 & 0 \\ 2 & 0 & 0 & 2 & 2 & 2 & 2 & 1 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 & 0 & 0 & 2 & 2 & 2 \\ 1 & 2 & 2 & 0 & 2 & 0 & 2 & & & \end{bmatrix}$$

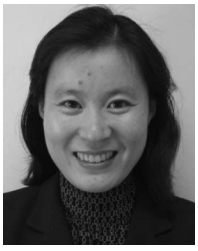
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2016.2621106, IEEE Transactions on Big Data

The decrypted result shows that $m'' = m$, which implies that our improved NTRU cryptosystem can successfully correct the decryption failure while the original NTRU can not.



Chunqiang Hu obtained his M.S degree and first PhD degree in computer Science and Technology from Chongqing University, Chongqing, China, in 2009 and 2013, respectively. He received his B.S. degree in Computer Science and Technology from Southwest University, Chongqing, China, in 2006. He was a visiting scholar at The George Washington University from Jan., 2011 to Dec., 2011, and then got his second PhD degree in Computer Science, The George Washington University, Washington DC in 2016. He won the Best Paper Award in ACM PAMCO 2016. His current research interests include applied cryptography, big data security and privacy, privacy-aware computing, wireless and mobile security, and algorithm design and analysis. He is a member of IEEE and ACM.

Wei Li received her Ph.D. degree in computer science from The George Washington University, Washington DC, in 2016, and her M.S. degree in Computer Science from Beijing University of Posts and Telecommunications, in 2011. She is an assistant professor at the Department of Computer Science, George State University. She won the Best Paper Awards in ACM MobiCom Workshop CRAB 2013 and WASA 2011. Her current research spans the areas of secure and privacy-aware computing, secure and truthful auctions in dynamic spectrum access, resource management in cognitive radio networks and WiFi-based wireless access networks, game theory, and algorithm design and analysis. She is a member of IEEE and a member of ACM.



Xiuzhen Cheng received her M.S. and Ph.D. degrees in computer science from the University of Minnesota – Twin Cities in 2000 and 2002, respectively. She is a professor in the Department of Computer Science, The George Washington University, Washington, DC. Her current research interests include privacy-aware computing, wireless and mobile security, cyber physical systems, mobile computing, and algorithm design and analysis. She has served on the editorial boards of several technical journals and the technical program committees of various professional conferences/workshops. She also has chaired several international conferences. She is a Fellow of the IEEE and a member of ACM.

Jiguo Yu received his Ph.D. degree in School of mathematics from Shandong University in 2004. He became a full professor in the School of Computer Science, Qufu Normal University, Shandong, China in 2007. Currently he is a full professor in the School of Information Science and Engineering, Qufu Normal University. His main research interests include privacy-aware computing, wireless networking, distributed algorithms, peer-to-peer computing, and graph theory. Particularly, he is interested in designing and analyzing algorithms for many computationally hard problems in networks. He is a member of the IEEE and ACM, and a senior member of the CCF (China Computer Federation).

Shengling Wang is currently an associate professor in the College of Information Science and Technology, Beijing Normal University. She received her Ph.D. in 2008 from Xi'an Jiaotong University. After that, she did her postdoctoral research in the Department of Computer Science and Technology at Tsinghua University. Then she worked as an assistant and associate professor, respectively, from 2010 and from 2013 in the Institute of Computing Technology of the Chinese Academy of Sciences. Her Research interest include privacy-aware computing, mobile/wireless networking, game theory, and crowdsourcing.

Rongfang Bie received her Ph.D. degree from Beijing Normal University, China, in 1996. Currently she is a professor at Beijing Normal University. She visited the Computer Laboratory, University of Cambridge, United Kingdom, in 2003. Her current research interests include knowledge representation and acquisition for the Internat of Things, computational intelligence, and model theory.