SCIS 2012 The 29th Symposium on
Cryptography and Information Security
Kanazawa, Japan, Jan. 30 - Feb. 2, 2012
The Institute of Electronics,
Information and Communication Engineers

The Distributed Decryption Schemes for Somewhat Homomorphic Encryption

Ryo Hiromasa * Yoshifumi Manabe † Tatsuaki Okamoto ‡

Abstract— We propose distributed decryption schemes for somewhat homomorphic encryption (SHE). The proposed schemes are constructed based on the encryption scheme by Brakerski and Vaikuntanathan. In SHE, homomorphic multiplication increases the number of elements in a ciphertext. The proposed scheme can decrypt a ciphertext containing more than two elements with k out of N parties. The distributed decryption scheme introduced by Damgard, Pastro, Smart and Zakarias needs a trusted third party for decrypting a ciphertext that has more than two elements with N out of N parties. We present a distributed decryption scheme in which the parties can decrypt ciphertexts of any size without a trusted third party. In addition, we evaluate the errors of the proposed schemes.

Keywords: Somewhat Homomorphic Encryption, Distributed Decryption, Ring Learning with Errors

1 Introduction

Fully Homomorphic Encryption (FHE) allows us to compute arbitrary operations homomorphically. In the last two years, several FHE schemes have been proposed and improved [2–4, 7, 8, 13, 14], but they are not yet practical. These FHE schemes are constructed from somewhat homomorphic encryption (SHE) schemes by installing error management techniques (e.g., bootstrapping and squashing). SHE schemes can also add and multiple ciphertexts homomorphically. In SHE schemes, the number of the homomorphic multiplications is limited, because it increases more error than homomorphic additions. Therefore, SHE schemes can evaluate only a limited number of circuits, but they are more efficient than FHE schemes. Since practical applications need not to compute arbitrary operations, SHE scheme is useful in practical applications.

Many FHE schemes are based on the learning with errors (LWE) assumption. In several earlier works that relate to the distributed decryption based on the LWE assumption, schemes based on the standard LWE [1] and the ring LWE [5] were proposed. In [1], which is based on the standard LWE, a scheme was constructed from a variant of the cryptosystem described in [11]. In this scheme, if equal to or more than k out of N parties cooperate to decrypt, the decryption succeeds. However, if the amount of the cooperating parties is equal to or less than $k-1$, the decryption fails. In [5], a scheme is constructed by extending SHE [3] based on the ring LWE. This scheme allows the parties to decrypt ciphertexts by cooperating with N out of N parties. The ciphertexts often have more than two elements in the homomorphic encryption, because homomorphic multiplication increases the number of elements in the ciphertexts. In the encryption scheme [3], decrypting ciphertexts requires the powers of the secret key. When every party chooses a share of the secret key and construct a secret key from the share, it is not easy to construct the powers of the secret key from the parties share. Therefore, the scheme in [5] requires a trusted third party to decrypt the ciphertext that contains more than two elements.

1.1 Our Results

We construct a distributed decryption scheme for somewhat homomorphic encryption described in [3]. In this scheme, any party group whose size is equal to or more than k can decrypt ciphertexts. Our (k,N) scheme requires a trusted third party to decrypt the ciphertext containing more than two elements, so we also construct improved schemes. Using the proposed improvements, parties can decrypt a ciphertext of any size without a trusted third party. In addition, we evaluate the size of the errors generated when the parties decrypt using the proposed schemes.

In this paper, we introduce the SHE scheme based on the ring LWE that is used in our schemes. Next, we propose distributed decryption schemes for SHE that allow the parties to decrypt the ciphertext of any size without a trusted third party. Finally, we describe the evaluation of the size of the errors generated in the proposed schemes.

2 Somewhat Homomorphic Encryption

In this section, we show the encryption scheme used in the proposed schemes and a mathematical assump-

* Department of Social Informatics, Graduate School of Informatics, Kyoto University.
† NTT Communication Science Laboratories
‡ NTT Information Sharing Platform Laboratories

tion used in the encryption scheme as a security basis. To construct distributed decryption schemes, we use SHE based on the ring LWE assumption, which is described by Brakerski and Vaikuntanathan [3].

2.1 Ring LWE

Ring LWE(RLWE) is an assumption parameterized by a degree n integer polynomial $f(x) \in \mathbb{Z}[X]$ and a prime integer $q \in \mathbb{Z}$.

We consider the ring $R = \mathbb{Z}[X]/\langle f(x)\rangle$ and $R_q = R/qR = \mathbb{Z}_q[X]/\langle f(x)\rangle$ ($\langle f(x)\rangle$ is the group generated by $f(x)$). In this SHE scheme, we set $f(x) = x^n + 1$ where n is a power of 2. Each element over the ring is a degree $n-1$ polynomial and can be viewed as a vector of degree n, namely, when we let $a \in R$, a is denoted as $(a_0, a_1, ..., a_{n-1})$, where a_i is an element of $\mathbb{Z}$ and a coefficient of the polynomial. Addition over the ring is done by adding two such vectors component-wise modulo q. Multiplication is polynomial multiplication modulo $f(x)$. For element $a = (a_0, a_1, ..., a_{n-1}) \in R$, we let $|a| = ||a||_\infty = \max |a_i|$ be the size of the element of R. Here, let $a \xleftarrow{U} R$ denote that a is selected from R at uniformly random. In addition, when we let χ be the discrete gaussian distribution over R, let $a \xleftarrow{R} \chi$ denote that we choose a along the distribution.

The RLWE assumption is defined as follows.

Definition. 2.1 (Ring LWE Assumption)
Let χ be a discrete gaussian distribution over R. For $a_i, s \xleftarrow{U} R_q$ and $e_i \xleftarrow{R} \chi$, given any polynomial number of samples $(a_i, b_i = a_i s + e_i)$, b_i are computationally indistinguishable from the elements chosen from R_q at uniformly random.

This assumption is equal to the variant where $s \xleftarrow{R} \chi$. In addition, Theorem 2.2 holds by setting modulus polynomial $f(x)$ to be cyclotomic polynomial $x^n + 1$. We set the error distribution to be the discrete gaussian distribution $D_{\sigma,\mathbb{Z}}$ over R where σ is the standard deviation. According to this setting, Theorem 2.1 holds.

We evaluate the error of the proposed schemes from the following two theorems.

Theorem. 2.1 (see [10])
Let $n \in \mathbb{N}$. For any real number $\sigma > \omega(\sqrt{\log n})$, it holds that

$$Pr_{x \xleftarrow{R} \chi}[||x||_\infty > \sigma\sqrt{n}] \leq 2^{-n+1}.$$

Theorem. 2.2 (see [7])
Let $n \in \mathbb{N}$, $f(x) = x^n + 1$, and $R = \mathbb{Z}[X]/\langle f(x)\rangle$. For any $s, t \in R$, it holds that

$$||st||_\infty < n \cdot ||s||_\infty ||t||_\infty.$$

2.2 Somewhat Homomorphic Encryption

Here, we describe the SHE scheme used in the proposed schemes. The SHE scheme is based on RLWE and uses the following parameters.

n: n is the degree of the modulus polynomial. n is power of 2.

q: q is a prime integer, where $q \equiv 1 \mod 2n$.

$f(x)$: We set the modulus polynomial such that $f(x) = x^n + 1$.

R, R_q: Let $R = \mathbb{Z}[X]/\langle f(x)\rangle$ and let $R_q = \mathbb{Z}_q[X]/\langle f(x)\rangle$.

t: Let $t(< q)$ be a prime integer. It defines the message space.

R_t: Let $R_t = \mathbb{Z}_t[X]/\langle f(x)\rangle$ be the message space.

χ: Let χ be the discrete gaussian distribution over R.

The SHE scheme consists of five algorithms SHE.{KeyGen, Enc, Dec, Add, Mult}.

SHE.KeyGen(1^k): This algorithm takes security parameter 1^k as an input and outputs public key pk and secret key sk. Choose $a_1 \xleftarrow{U} R_q$ and $s, e \xleftarrow{R} \chi$, then $sk = s$ and compute $pk = (a_0 = -(a_1 s + te), a_1)$.

SHE.Enc(pk, m): Choose $u, f, g \xleftarrow{R} \chi$. Let $m \in R_t$ be a message. We compute a ciphertext $\mathbf{c}$ as follows.

$$\mathbf{c} = (c_0, c_1) = (a_0 u + tf + m, a_1 u + tg).$$

SHE.Dec($sk, \mathbf{c}$): Let $\mathbf{c} = (c_0, c_1, ..., c_\delta)$ be a ciphertext. We compute

$$d = \sum_{i=0}^{\delta} c_i s^i \mod q.$$

We can obtain plaintext m by computing $m = d \mod t$.

SHE.Add($\mathbf{c}, \mathbf{c}'$): Let $\mathbf{c} = (c_0, c_1, ..., c_\lambda)$ and $\mathbf{c}' = (c'_0, c'_1, ..., c'_\delta)$be two ciphertexts. Homomorphic addition is performed by adding these ciphertexts component wise. If $\lambda \neq \delta$, then the shorter ciphertext is padded with zeros. The algorithm SHE.Add outputs $(c_0 + c'_0, c_1 + c'_1, ..., c_{max(\lambda,\delta)} + c'_{max(\lambda,\delta)})$ as a result.

SHE.Mult($\mathbf{c}, \mathbf{c}'$): Let $\mathbf{c} = (c_0, c_1, ..., c_\lambda)$ and $\mathbf{c}' = (c'_0, c'_1, ..., c'_\delta)$ be two ciphertexts. Here, we do not pad the shorter ciphertext. Let v^i be a symbolic variable, and we consider

$$\sum_{i=0}^{\lambda} c_i v^i \sum_{i=0}^{\delta} c'_i v^i = \sum_{i=0}^{\lambda+\delta} c''_i v^i$$

If we replace v^i with s^i, we find that this implies that the results of the decryption are multiplied. The algorithm SHE.Mult outputs $(c''_0, c''_1, ..., c''_{\lambda+\delta})$ as a result of homomorphic multiplication.

3 Distributed Decryption Schemes

In this section, we describe the $(N,\ N)$ distributed decryption scheme and $(k,\ N)$ distributed decryption scheme. The proposed distributed decryption schemes use the same parameters as the SHE scheme.

In the distributed decryption schemes, the Setup algorithm and Encryption algorithm are changed from the ones in the SHE scheme. Instead of the decryption algorithm in the SHE scheme, the parties cooperate to decrypt the ciphertext in a Distributed Decryption phase. Homomorphic addition and multiplication are performed using the same algorithm as in the SHE scheme. The Setup phase is the phase used to generate a public key and the shares of a secret key. In the Encryption phase, we encrypt a message by using the public key generated in the Setup phase. In the Distributed Decryption phase, the parties cooperate to decrypt a ciphertext.

3.1 $(N,\ N)$ Distributed Decryption Scheme

In $(N,\ N)$ schemes, the parties can decrypt a ciphertext only if all the N parties cooperate. The scheme by Damgard, et al. [5] shown below does not allow the parties to decrypt ciphertexts containing more than two elements.

Here, we describe the $(N,\ N)$ distributed decryption scheme. This scheme was proposed by Damgard, et al. [5].

Setup : Choose $a_1 \xleftarrow{U} R_q$. Each party $P_i (i = 1, ..., N)$ chooses $s_i, e_i \xleftarrow{R} \chi$ and computes distributed public key $pk_i = (a_{0,i} = -(a_1 s_i + te_i), a_1)$ where s_i is a share of the secret key. The parties compute public key pk as follows from pk_i revealed by each P_i.

$$\begin{aligned} pk &= (a_0, a_1) \\ &= (\sum_{i=1}^{N} a_{0,1}, a_1) \\ &= (-(a_1 \sum_{i=1}^{N} s_i + t \sum_{i=1}^{N} e_i)) \\ &= (-(a_1 s + te), a_1), \end{aligned}$$

where $s = \sum_{i=1}^{N} s_i, e = \sum_{i=1}^{N} e_i$.

Encryption : Let $m \in R_t$ be a message, and we compute ciphertext $\mathbf{c}$ as follows.

$$\mathbf{c} = \mathsf{SHE.Enc}(pk, m)$$

Distributed Decryption :

1. Party $P_i (i = 1, ..., N)$ performs the following procedure in turns.
   (i) P_i computes $\mathbf{d}_i = (d_{i,0}, d_{i,1}) = (c_0, c_1 s_i)$.
   (ii) P_i chooses $r_i \xleftarrow{R} \chi$.
   if $i = 1$ **then** $D = d_{i,0} + d_{i,1} + tr_i$
   else $D = D + d_{i,1} + tr_i$
   (iii) **if** $i \neq N$ **then** P_i sends D to P_{i+1}.
2. Parties can obtain the plaintext by computing as follows.

$$(D \mod q) \mod t$$

3.2 $(k,\ N)$ Distributed Decryption Scheme

In the $(k,\ N)$ scheme, the parties recover the secret key by using lagrange interpolation at the Distributed Decryption phase. This secret sharing method is described by Shamir [12]. To recover the secret key, each party P_i generates a degree $k-1$ polynomial $f_i(x)$ such that

$$s_i = f_i(0) = \sum_{j=1}^{N} f_i(j) l_j(0),$$

where

$$l_j(x) = \frac{\prod_{\theta=1}^{j-1}(x-\theta) \cdot \prod_{\theta=j+1}^{N}(x-\theta)}{\prod_{\theta=1}^{j-1}(j-\theta) \cdot \prod_{\theta=j+1}^{N}(j-\theta)}$$

and s_i is a share of the secret key.

We describe the $(k,\ N)$ distributed decryption scheme.

Setup : Choose $a_1 \xleftarrow{U} R_q$. Each party $P_i (i = 1, ..., N)$ chooses $s_i, e_i \xleftarrow{R} \chi$ and computes public key $pk_i = (a_{0,i} = -(a_1 s_i + te_i), a_1)$. In addition, P_i generates a polynomial over R_q of degree $k-1$ that has s_i as the constant term.

$$f_i(x) = s_i + a_1 x + a_2 x^2 + \cdots + a_{k-1} x^{k-1},$$

where $a_1, ..., a_{k-1} \in R_q$ and s_i is a share of the secret key. P_i sends $f_i(l)$ to $P_l (l = 1, ..., N)$. In addition, the parties compute public key pk in the same way in the $(N,\ N)$ Distributed Decryption scheme.

Encryption : We compute a ciphertext in the same way as the Encryption phase of the $(N,\ N)$ Distributed Decryption scheme.

Distributed Decryption : Let $\mathbf{c} = (c_0, c_1)$ be the ciphertext. Equal to or more than k out of N parties cooperate in the decryption. Let $\mathcal{S} = \{\iota_1, ..., \iota_h\}$ $(h \geq k)$ be the set of the indexes of the cooperating parties.

1. Every $P_{\iota_i} (i = 1, ..., h)$ performs the following procedure in turns.
   (i) Let $\rho_i = l_i(0) \sum_{j=1}^{N} f_j(\iota_i)$ and every P_{ι_i} computes $\mathbf{d}_i = (d_{i,0}, d_{i,1}) = (c_0, c_1 \rho_i)$.
   (ii) P_{ι_i} chooses $r_i \xleftarrow{R} \chi$.
   if $i = 1$ **then** $D = d_{i,0} + d_{i,1} + tr_i$
   else $D = D + d_{i,1} + tr_i$
   (iii) **if** $i \neq h$ **then** P_{ι_i} sends D to $P_{\iota_{i+1}}$.
2. The parties can obtain the plaintext by computing as follows.

$$(D \mod q) \mod t$$

3.3 Distributed Decryption for Ciphertext of Any Size

The above schemes can be used for ciphertexts that have less than three elements. However, the number of elements of a ciphertext increases by computing homomorphic multiplication, so we need a scheme in which the parties can decrypt such a ciphertext to allow us to compute homomorphic multiplication.

Let $= (c_0, c_1, ..., c_\delta)$ be the ciphertext. The Distributed Decryption phase for the ciphertext containing more than two elements proceeds as follows.

Distributed Decryption:

1. $\mathbf{d} = (d_0, d_1, ..., d_\delta) = (c_0, c_1, c_2, ..., c_\delta)$
2. $j = 1$, and perform the following procedure to $j = \delta$.
   (i) Every party P_i computes $\mathbf{d}'_i$ as follows.
   if $i = 1$ **then**

   $$\mathbf{d}'_i = (d_0, d_1, ..., d_{j-1}, d_j s_i, d_{j+1} s_i, ..., d_\delta s_i)$$

   else

   $$\mathbf{d}'_i = (\underbrace{0, 0, ..., 0}_{j}, d_j s_i, d_{j+1} s_i, ..., d_\delta s_i)$$

   (ii) Every party P_i performs computations as follows in turns.
   i. P_i chooses $r_{i,1}, ..., r_{i,\delta+1-j} \xleftarrow{R} \chi$, generates a vector

   $$\mathbf{r} = (\underbrace{0, 0, ..., 0}_{j}, r_{i,1}, r_{i,2}, ..., r_{i,\delta+1-j}),$$

   and computes
   if $i = 1$ **then** $\mathbf{d} = \mathbf{d}'_i + t\mathbf{r}$
   else $\mathbf{d} = \mathbf{d} + \mathbf{d}'_i + t\mathbf{r}$.
   ii. **if** $i \neq N$ **then** P_i sends $\mathbf{d}$ to P_{i+1}.
   (iii) $j = j + 1$
3. The parties can obtain the plaintext by computing as follows.

$$(\sum_{i=0}^{\delta} d_i \mod q) \mod t$$

In addition, we show the improvement of the Distributed Decryption phase in the (k,N) scheme. This improved phase is almost the same as that in the (N,N) scheme.

Distributed Decryption: Equal to or more than k out of N parties cooperate in the decryption. Let $\mathcal{S} = \{\iota_1, ..., \iota_h\}$ $(h \geq k)$ be the set of indexes of the cooperating parties.

1. $\mathbf{d} = (d_0, d_1, ..., d_\delta) = (c_0, c_1, c_2, ..., c_\delta)$
2. $j = 1$, and perform the following procedure to $j = \delta$.
   (i) Let $\rho_i = l_{\iota_i}(0) \sum_{\theta=1}^{N} f_\theta(\iota_i)$, every party P_{ι_i} computes
   if $i = 1$ **then**

   $$\mathbf{d}'_i = (d_0, d_1, ..., d_{j-1}, d_j \rho_i, d_{j+1} \rho_i, ..., d_\delta \rho_i)$$

   else

   $$\mathbf{d}'_i = (\underbrace{0, 0, ..., 0}_{j}, d_j \rho_i, d_{j+1} \rho_i, ..., d_\delta \rho_i)$$

   (ii) Every party P_{ι_i} performs computations as follows in turns.
   i. P_{ι_i} chooses $r_{i,1}, ..., r_{i,\delta+1-j} \xleftarrow{R} \chi$, generates a vector

   $$\mathbf{r} = (\underbrace{0, 0, ..., 0}_{j}, r_{i,1}, r_{i,2}, ..., r_{i,\delta+1-j}),$$

   and computes
   if $i = 1$ **then** $\mathbf{d} = \mathbf{d}'_i + t\mathbf{r}$
   else $\mathbf{d} = \mathbf{d} + \mathbf{d}'_i + t\mathbf{r}$.
   ii. **if** $i \neq h$ **then** P_{ι_i} sends $\mathbf{d}$ to $P_{\iota_{i+1}}$.
   (iii) $j = j + 1$
3. The parties can obtain the plaintext by computing as follows.

$$(\sum_{i=0}^{\delta} d_i \mod q) \mod t$$

4 Evaluation of Errors

In this section, we describe the evaluation of the errors of the proposed schemes. The error consists of the sum of the errors of a ciphertext and the errors generated when the parties cooperate to decrypt a ciphertext in the distributed decryption schemes. The parameters of the proposed schemes depend on the size of the error. Thus, we must evaluate the size of the errors in the ciphertext to which some homomorphic computations are done.

First we evaluate the error for decrypting a fresh ciphertext, and then we evaluate the error generated when the parties decrypt the ciphertext computed D multiplications followed by A additions.

4.1 Error for Decrypting Fresh Ciphertext

When the parties decrypt a fresh ciphertext, the error term $te' + t\sum_i r_i$ is generated. This is because $\mathsf{SHE.Dec}(sk, \mathbf{c}) = c_0 + c_1 s = te' + t\sum_i r_i + m$, where $e' = (-eu + fs + g)$ in the above SHE scheme. Here, te' is the error of the ciphertext and $t\sum_i r_i$ is the error generated when the parties decrypt the ciphertext. Term e, u, f, s, and g are drawn from discrete gaussian distribution χ. According to Theorem 2.1, the size of g is at most $\sigma\sqrt{n}$. From Theorem 2.2, eu and fs have size at most $N \cdot n \cdot (\sigma\sqrt{n})^2 = N\sigma^2 n^2$, because eu and fs are polynomial multiplications. According to this estimation, we find that the size of te' is at most $t(2N\sigma^2 n^2 + \sigma\sqrt{n})$.

Next, we evaluate the errors generated in the distributed decryption phase in the proposed schemes. In the (N, N) distributed decryption scheme, all the N parties cooperate in decryption. When a party passes to the next party the ciphertext to which its share of the secret key is multiplied, the party adds the error tr_i to the ciphertext, where r_i is chosen from χ. This procedure generates error $t\sum_i r_i$, which is at most $tN\sigma\sqrt{n}$. In the (k, N) distributed decryption scheme, equal to or more than k out of N parties might cooperate in the decryption, so $t\sum_i r_i$ has magnitude at most $tN\sigma\sqrt{n}$. Therefore, the error of the (k, N) scheme is as large as that for the (N, N) scheme. Based on the above estimation, the size of the errors generated when the parties perform decryption in the (N,N) and (k,N) schemes is at most $t(2N\sigma^2n^2 + (N+1)\sigma\sqrt{n})$.

Decryption succeeds if $|te'| < q/2$, so we must choose q such that

$$2t\{2N\sigma^2n^2 + (N+1)\sigma\sqrt{n}\} < q$$

In the proposed improved schemes, if the ciphertext size is $\delta + 1$, the parties multiply their shares and add error to the ciphertext in δ rounds. For a fresh ciphertext, the multiplication and the addition in the distributed decryption phase is performed in one round. The addition in one round generates error that is at most $tN\sigma\sqrt{n}$, which is as large as that in the (N,N) and (k,N) schemes. Therefore, after computing the distributed decryption phase in the improved schemes, the size of the error of the fresh ciphertext is increased to at most $t\{2N\sigma^2n^2 + (N+1)\sigma\sqrt{n}\}$. Consequently, the size of the error in the improved schemes is as large as those in (N,N) schemes and the (k,N) schemes, so in the improved schemes we have only to choose q that has the same size as q of the (N, N) schemes to decrypt the fresh ciphertext.

4.2 Error for Decrypting Ciphertext Computed D Multiplications Followed by A Additions

Let η be the error of a fresh ciphertext. One multiplication increases the error from η to $n \cdot \eta^2$, so the size of the error after computing D multiplications is at most $n^D \cdot \eta^{D+1}$. Then with A additions, the error increases to $A \cdot n^D \cdot \eta^{D+1}$. Let η_{final} be the error of the ciphertext computed D multiplications followed by A additions. In the schemes for ciphertexts that have less than three elements, The size of the error of a fresh ciphertext is at most $t\{2N\sigma^2n^2 + (N+1)\sigma\sqrt{n}\}$. Therefore, the size of η_{final} is

$$\eta_{\text{final}} \leq A \cdot n^D \cdot t^{D+1}(2N\sigma^2n^2 + (N+1)\sigma\sqrt{n})^{D+1}.$$

After computing the distributed decryption phase in the improved schemes, the error that has magnitude at most $tN\sigma\sqrt{n} \cdot |\sum_{i=0}^{n-1} s^i|$ is added to each element c_n in the ciphertext. Overall, the distributed decryption phase in the improved schemes add the error at most $tN\sigma\sqrt{n} \cdot |\sum_{i=1}^{\delta}\sum_{j=0}^{i-1} s^j|$ to error η_{final}. Hence, we have only to evaluate the size of $\sum_{i=1}^{\delta}\sum_{j=0}^{i-1} s^j$. From Theorem 2.2, $|s^j| = n^{j-1}|s|^j$. Thus, the size of $\sum_{i=1}^{\delta}\sum_{j=0}^{i-1} s^j$ is at most

$$\begin{aligned}
\left|\sum_{i=1}^{\delta}\sum_{j=0}^{i-1} s^j\right| &= \sum_{i=1}^{\delta}(1 + \sum_{j=1}^{i-1}|s^j|) \\
&= \delta + \sum_{i=1}^{\delta}\sum_{j=1}^{i-1} n^{j-1}|s|^j \\
&= \delta + \sum_{i=1}^{\delta}\frac{|s| - n^{i-1}|s|^i}{1 - n|s|} \\
&\leq \delta + \sum_{i=1}^{\delta} n^{i-1}|s|^i \\
&= \delta + \frac{|s| - n^{\delta}|s|^{\delta+1}}{1 - n|s|} \\
&\leq \delta + n^{\delta}|s|^{\delta+1}.
\end{aligned}$$

Here, s is the sum of s_i, so from Theorem 2.1, the size of s is at most $N\sigma\sqrt{n}$. Consequently, the error added at the distributed decryption phase in the improved schemes has size that is at most

$$\begin{aligned}
&tN\sigma\sqrt{n} \cdot \left|\sum_{i=1}^{\delta}\sum_{j=0}^{i-1} s^j\right| \\
&< tN\sigma\sqrt{n} \cdot \{n^{\delta}(N\sigma\sqrt{n})^{\delta+1} + \delta\}.
\end{aligned}$$

After D multiplications, the number of the elements in the ciphertext is $D+2$, so $\delta = D+1$. Hence, the size of the final error generated when the parties decrypt the ciphertext in the improved schemes is at most

$$\begin{aligned}
&\eta_{\text{final}} + tN\sigma\sqrt{n} \cdot \left|\sum_{i=1}^{\delta}\sum_{j=0}^{i-1} s^j\right| \\
&< A \cdot n^D \cdot t^{D+1}(2N\sigma^2n^2 + (\delta N+1)\sigma\sqrt{n})^{D+1} \\
&\quad + tN\sigma\sqrt{n} \cdot \{n^{D+1}(N\sigma\sqrt{n})^{D+2} + D + 1\}.
\end{aligned}$$

For successful decryption of a ciphertext of any size, we must choose q that is at most

$$\begin{aligned}
q &> 2(\eta_{final} + tN\sigma\sqrt{n} \cdot \left|\sum_{i=1}^{\delta}\sum_{j=0}^{i-1} s^j\right|) \\
&> 2A \cdot n^D \cdot t^{D+1}(2N\sigma^2n^2 + (\delta N+1)\sigma\sqrt{n})^{D+1} \\
&\quad + 2tN\sigma\sqrt{n} \cdot \{n^{D+1}(N\sigma\sqrt{n})^{D+2} + D + 1\}.
\end{aligned}$$

5 Conclusion

We constructed distributed decryption schemes for somewhat homomorphic encryption. Our schemes allow equal to or more than k out of N parties to decrypt ciphertexts of any size without a trusted third party. In addition, we evaluated the errors generated when the parties cooperate to decrypt ciphertexts in the proposed schemes.

References

[1] Rikke Bendlin and Ivan Damgard. Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. *Theory of Cryptography - TCC 2010, LNCS*, 5978:201–218, 2010.

[2] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. *Cryptology ePrint Archive*, 2011. Report 2011/344.

[3] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key depedent message. *Advances in Cryptology - CRYPTO 2011, LNCS*, 6841:505–524, 2011.

[4] Jean-Sebastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Fully homomorphic encryption over the integers with shorter public-keys. *Advances in Cryptology - CRYPTO 2011, LNCS*, 6841:487–504, 2011.

[5] Ivan Damgard, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. *Cryptology ePrint Archive*, 2011. Report 2011/535.

[6] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. *Advances in Cryptology - CRYPTO '89, LNCS*, 435:307–315, 1990.

[7] Craig Gentry. Fully homomorphic encryption using ideal lattices. *STOC*, pages 169–178, 2009.

[8] Craig Gentry. Fully homomorphic encryption without bootstrapping. *Cryptology ePrint Archive*, 2011. Report 2011/277.

[9] Kristin Lauter, Michael Naehrig, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? *Cryptology ePrint Archive*, 2011. Report 2011/405.

[10] Daniele Micciancio and Oded Regev. Worst-case to average-case reduction based on gaussian measures. *SIAM J.Comput*, 37(1):267–302, 2007.

[11] Oded Regev. On lattice, learning with errors, random linear codes, and cryptography. *STOC*, pages 84–93, 2005.

[12] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[13] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. *Public Key Cryptography - PKC 2010, LNCS*, 6056:420–443, 2010.

[14] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integer. *Advances in Cryptology - EUROCRYPT 2010, LNCS*, 6110:24–43, 2010.