Research on Two-Party Cooperative Aigis-sig Digital Signature Protocol

Fu Yu and Zhao Xiufeng(✉)

Information Engineering University, Zhengzhou 450001, China

Abstract. The digital signature scheme is essential for electronic commerce and e-government security authentication. With the rapid advancement of mobile internet technologies, safe key storage in mobile terminals has become a new challenge. To solve the leak of the signature private key, a method for generating a two-party cooperative signature has been proposed. That is, each participant generates the signature secret key and shares it with the other after which the signature is generated interactively. The method is robust because one party cannot recover the secret key, which not only guarantees the correctness of the signature but can resist the security implications caused by the corruption of a single mobile terminal. Considering the threat from quantum computing technology to conventional public-key cryptographic algorithms, in this paper, the lattice-based post-quantum digital signature algorithm, Aigis-sig, published in the international conference on public-key cryptography is discussed. Furthermore, the two-party Aigis-sig signature protocol is proposed. The protocol contains two sub-protocols: distributed secret key generation and collaborative signing protocol. In addition, the homomorphic encryption scheme is introduced in the protocol to ensure that the intermediate of the protocol does not reveal the private key information. The evaluation demonstrates that the protocol has correctness and feasibility. In the case that both parties are honest, the cooperative signature is existential unforgeability against the chosen-message attack.

Keywords: Post-quantum algorithm · Digital signature · Homomorphic encryption · Aigis-sig · Cooperative signature

1 Introduction

The digital signature is applied in identity authentication, data integrity, nonrepudiation, and anonymity. The signer uses the private key to generates the message's signature, and the verifier uses the public key to verify the signature's validity. However, some mobile terminals' security in the mobile internet application is poor, and the storage security of the private key is difficult to guarantee. Therefore, this paper introduces a two-party cooperative digital signature generation method. Two-party collaborate to generate a digital signature, namely, the generation of the private key and signature is distributed. The signing procedure is realized by the interaction of the two-party, and any participant cannot recover the integrated private key, which ensures both the

W. Shi et al. (Eds.): SPNCE 2021, LNICST 423, pp. 53–63, 2022.
https://doi.org/10.1007/978-3-030-96791-8_4

correctness of the signature and the security of the signature private key. The verification process can be performed by a user using the authentication public key. The two-party cooperative generation of the digital signature can achieve better robustness and resist the weakness brought by one-party corruption. Even if one party's share of the signature key of one party is leaked, the private key of the signature will remain secure. Currently, some two-party and multiparty schemes based on the elliptic curve signature algorithm (ECDSA) have been proposed in [1, 2], and [3]. However, for post-quantum digital signature schemes, such as the CRYSTALS-Dilithium [4] and its improved algorithm Aigis-sig [5], no researchers have proposed two-party or multiparty signing schemes.

In this paper, we consider a two-party digital signature protocol based on Aigis-sig. We describe how one can build a distributed key generation and collaborative signing protocol for Aigis-sig. Our protocol achieves coordinative generating Aigis-sig signature resorts to homomorphic encryption [6] by two-party. In Sect. 2, the specific interaction process of the protocol is given and the correctness of the signature protocol is proved. In Sect. 3, the efficiency of the protocol is analyzed and the unforgeability of the cooperative signature is proved.

2 Preliminaries

2.1 Aigis-sig

Aigis-sig is a lattice-based signature scheme [5], constructed using the Fiat-Shamir heuristic [7, 8], whose security is based on the hardness of the asymmetric module learning with errors (AMLWE) problem and the asymmetric module small integer solutions (AMSIS) problem [5]. Compared with the Dilithium scheme [4], the Aigis-sig scheme can achieve better comprehensive efficiency without changing the security, and the lengths of the public key, private key, and signature are smaller. Considering both security and efficiency, the template version of the scheme without compressing the public verify key has been chosen in our protocol. The scheme includes three algorithms: Key generation, signing procedure, and verification, described as follows:

Key Generation Algorithm. $Aigis.sig - KeyGen(1^{\kappa})$: At first, the κ is defined as the secure parameter. The key generation algorithm randomly chooses a seed $\rho \xleftarrow{\$} \{0,1\}^{256}$ and generates a matrix $\boldsymbol{A} = Expand(\rho) \in R_q^{k \times l}$, each of whose entries is a polynomial belong to the ring $R_q = \mathbb{Z}_q[X]/(X^N + 1)$. Afterward, the algorithm samples random secret key vectors $(\boldsymbol{s}_1, \boldsymbol{s}_2) \xleftarrow{\$} S_{\eta_1}^k \times S_{\eta_2}^l$. S_η denotes a polynomial set in $R = \mathbb{Z}[X]/(X^N + 1)$, each polynomial's coefficient belongs to a finite set $\{-\eta, \ldots, \eta\}$ in S_η. Finally, the public key is computed as $\boldsymbol{t} = \boldsymbol{A}\boldsymbol{s}_1 + \boldsymbol{s}_2 \in R_q^k$, and the digest of the public key $tr = CRH(pk) \in \{0,1\}^{384}$ is produced through a collection-resistant hash function $CRH: \{0,1\}^* \to \{0,1\}^{384}$. The key generation algorithm returns $pk = (\rho, \boldsymbol{t})$ and $sk = (\rho, tr, \boldsymbol{s}_1, \boldsymbol{s}_2)$. All algebraic operations in this scheme are assumed to be over the polynomial ring R_q.

Signing Algorithm. $Aigis.sig - Sign(sk, M)$: Given the secret key $sk = (\rho, tr, \boldsymbol{s}_1, \boldsymbol{s}_2)$ and message M, the signing algorithm computes the digest $\mu = CRH(tr||M)$ of the message, and performs the following steps:

- Sample random vector $\boldsymbol{y} \xleftarrow{\$} S^l_{\gamma_1-1}$;
- Compute $\boldsymbol{w} = \boldsymbol{A}\boldsymbol{y} \in R^k_q$ and $\boldsymbol{w}_1 = HighBits_q(\boldsymbol{w}, 2\gamma_2)$;
- Compute the challenge $c = H(\mu, \boldsymbol{w}_1)$, $\boldsymbol{z} = \boldsymbol{y} + c\boldsymbol{s}_1$, and $\boldsymbol{u} = \boldsymbol{w} - c\boldsymbol{s}_2$. Denote B_τ as the set element of R that have τ coefficients are either -1 or 1 and the rest are 0, and the hash function is defined as $H : \{0, 1\}^* \to B_\tau$.
- Compute $(\boldsymbol{r}_1, \boldsymbol{r}_0) = Decompose_q(\boldsymbol{u}, 2\gamma_2)$;
- The potential signature is then computed as $\sigma = (\boldsymbol{z}, c)$, if the rejection sampling condition $\|\boldsymbol{z}\|_\infty < \gamma_1 - \beta_1$, $\|\boldsymbol{r}_0\|_\infty < \gamma_2 - \beta_2$, and $\boldsymbol{r}_1 = \boldsymbol{w}_1$ is satisfied. If not, restart the procedure from the sampling random vector $\boldsymbol{y} \xleftarrow{\$} S^l_{\gamma_1-1}$.

The signing algorithm returns the signature $\sigma = (\boldsymbol{z}, c)$.

Verification. *Aigis.sig* $-$ *Verify*(pk, M, σ). Given the public key $(\rho, \boldsymbol{t})$, message M, and signature $\sigma = (\boldsymbol{z}, c)$, the verifying algorithm computes $\boldsymbol{A} = Expand(\rho)$, $\mu = CRH(CRH(pk)||M)$, $\boldsymbol{w'}_1 = HighBits_q(\boldsymbol{A}\boldsymbol{z} - c\boldsymbol{t}, 2\gamma_2)$. Later, the hash value $c' = H(\mu, \boldsymbol{w'}_1)$ is computed. Then algorithm returns 1 if $\|\boldsymbol{z}\|_\infty < \gamma_1 - \beta_1$ and $c' = c$, otherwise returns 0 (Table 1).

Table 1. Decompose algorithm in Aigis-sig scheme

$Decompose_q(r, \alpha)$:
- $r = r \bmod^+ q$
- $r_0 = r \bmod^\pm \alpha$
- if $r - r_0 = q - 1$
- then $r_1 = 0$; $r_0 = r_0 - 1$
- else $r_1 = (r - r_0)/\alpha$
- return (r_1, r_0)

According to $(r_1, r_0) = Decompose_q(r, \alpha)$, the $r_1 = HighBits_q(r, \alpha)$ and $r_0 = LowBits_q(r, \alpha)$ have been defined to extract the "higher-order" bits and "lower-order" bits from the element r in $\mathbb{Z}_q$.

For an even positive integer α, the Aigis-sig scheme defines $r' = r \bmod^\pm \alpha$ as the unique element in the range $\left(-\frac{\alpha}{2}, \frac{\alpha}{2}\right]$ such that $r' = r \bmod \alpha$. For an odd positive integer α, the scheme defines $r' = r \bmod^\pm \alpha$ as the unique element in the range $\left[-\frac{\alpha-1}{2}, \frac{\alpha-1}{2}\right]$ such that $r' = r \bmod \alpha$. For any positive integer α, the scheme defines $r' = r \bmod^+ \alpha$ as the unique element in the range $[0, \alpha)$ such that $r' = r \bmod \alpha$. When the above algorithm is applied to a polynomial, the implication is that the corresponding operation is applied independently to each coefficient of the polynomial. The following lemma claims a crucial property of the above supporting algorithm, which is necessary for the correctness and security of the Aigis-sig scheme.

Lemma 1 [4] **.** if $\|\boldsymbol{s}\|_\infty \leq \beta$ and $\left\|LowBits_q(\boldsymbol{r}, \gamma)\right\|_\infty \leq \frac{\gamma}{2} - \beta$, we have:

$$HighBits_q(\boldsymbol{r}, \gamma) = HighBits_q(\boldsymbol{r} + \boldsymbol{s}, \gamma)$$

2.2 Homomorphic Encryption Scheme

Homomorphic encryption is a cryptographic primitive that allows users to perform arithmetic operations on encrypted data without decrypting it. if it allows arbitrary boolean or arithmetic circuits to be homomorphically evaluated, it is called fully homomorphic encryption. The first fully homomorphic encryption scheme was invented by Gentry in 2009 [9], and since then many efficient results have been introduced, such as Bra12 [10], BGV [11], GSW [12], FHEW [13], and CKKS [6].

In general, a homomorphic encryption scheme is a group of probabilistic polynomial-time (PPT) algorithms as follows (is the security parameter):

Key Generation. $KeyGen(\kappa, L)$: The inputs are the security parameter κ and the level parameter L, and it outputs an public encrypted key pk, a public evaluation key evk, and a secret decrypted key sk.

Encryption Algorithm. $Enc_{pk}(m)$: The input is public key pk and plaintext m, it outputs the ciphertext ct.

Decryption Algorithm. $Dec_{sk}(ct)$: The input is the secret key sk and ciphertext ct, it outputs the plaintext m.

Ciphertext Homomorphic Addition Evaluation. $Add_{evk}(ct_1, ct_2)$: The input is the ciphertext ct_1 and ct_2 of the plaintext m_1 and m_2. It outputs the ciphertext ct_{add} of the $m_1 + m_2$.

Ciphertext Homomorphic Multiply Evaluation. $Mult_{evk}(ct_1, ct_2)$: The input is the ciphertext ct_1 and ct_2 of the plaintext m_1 and m_2. It outputs the ciphertext $m_1 \cdot m_2$ of the ct_{mult}.

Homomorphic Evaluation. $HomEval_{HE}(evk_{HE}, f, ct_1, \ldots, ct_g)$: Using the evaluation key evk_{HE} applies a function f to ciphertexts $ct_i (i = 1, \ldots, g)$ and outputs a ciphertext ct_f, where f is an arithmetic circuit with addition and multiplication. We denote $ct_1 \oplus ct_2$ as the homomorphic addition of the ciphertexts ct_1 and ct_2, and the homomorphic multiplication is denoted as $ct_1 \otimes ct_2$.

2.3 Definition of Security

Assume that a standard digital signature scheme is denoted as π = (KeyGen, Sign, Verify). The security of standard digital signatures is defined as follows [5]:

EXPERIMENT 1.3.1. ($Expt - Sign_{\mathcal{S},\pi}(1^{\kappa})$)

1. $(pk, sk) \leftarrow KeyGen(1^{\kappa})$
2. $(M^*, \sigma^*) \leftarrow \mathcal{S}^{Sign_{sk}(\cdot)}(pk)$
3. Let Q be the set of all (M, σ) queried by $\mathcal{S}$ to its oracle. Then, the output of the experiment equals 1if and only if $(M^*, \sigma^*) \notin Q$ and $Verify_{pk}(M^*, \sigma^*) = 1$

Definition 1.3.1. If for every PPT adversary $\mathcal{S}$, there exists a negligible function λ such that for every κ,

$$Pr[Expt - Sign_{\mathcal{S},\pi}(1^{\kappa}) = 1] \leq \lambda(\kappa)$$

We announce that the digital signature scheme π satisfies the strongly existential unforgeability against adaptive chosen-message attack (SUF-CMA).

Next, we define the security of a distributed signing protocol.

EXPERIMENT 1.3.2. ($Expt - DistSign_{\mathcal{A},\Pi}(1^{\kappa})$)

$\pi = (\text{KeyGen}, \text{Sign}, \text{Verify})$ is a standard digital signature scheme.

1. $(M^*, \sigma^*) \leftarrow \mathcal{A}^{\Pi(\cdot,\cdot)}(pk)$
2. Let Q be the set of all M queried by $\mathcal{A}$ to its oracle. Then, the output of the experiment equals 1if and only if $M^* \notin Q$ and $Verify_{pk}(M^*, \sigma^*) = 1$ 。

If the adversary $\mathcal{A}$ conducts adaptive chosen-message cooperative signature queries with polynomial times and can forge the cooperative signature of the message that has not been inquired, the experiment returns 1, that is, the adversary wins the experiment and corrupts the scheme.

Definition 1.3.2. If for every PPT adversary $\mathcal{A}$, there exists a negligible function λ' such that for every κ,

$$Pr[Expt - DistSign_{\mathcal{A},\Pi}(1^{\kappa}) = 1] \leq \lambda'(\kappa)$$

Then, we state that the two-party signing protocol Π that based on the signature scheme π is security.

3 Two-Party Aigis-sig Signing Protocol

In this section, based on the Aigis-sig scheme [5], we present our two-party Aigis-sig signing protocol resort to homomorphic encryption [6]. The protocol includes two sub-protocols: the distributed key generation protocol and the collaborative signing protocol. Without loss of generality, we mark the two participants as P_1 and P_2, respectively. Before proceeding with the two protocols, P_1 and P_2 conduct the homomorphic key generation algorithm and distributed the key. That is, P_1 performs the key generation algorithm of a homomorphic encryption scheme to obtain the key (pk_{HE}, evk_{HE}, sk_{HE}), and sends (pk_{HE}, evk_{HE}) it to P_2.

3.1 The Distributed Key Generation Protocol

P_1 samples random seed $\rho \xleftarrow{\$} \{0,1\}^{256}$, vectors $(\boldsymbol{s}_{11}, \boldsymbol{s}_{12}) \xleftarrow{\$} S_{\eta_1}^k \times S_{\eta_2}^l$, computes $\boldsymbol{A} = Expand(\rho) \in R_q^{k \times l}$, $\boldsymbol{t}_1 = \boldsymbol{A}\boldsymbol{s}_{11} + \boldsymbol{s}_{12}$, and sends $(\boldsymbol{t}_1, \rho)$ to P_2.

P_2 samples random vectors $(\boldsymbol{s}_{21}, \boldsymbol{s}_{22}) \xleftarrow{\$} S_{\eta_1}^k \times S_{\eta_2}^l$, compute $\boldsymbol{A} = Expand(\rho)$, $\boldsymbol{t}_2 = \boldsymbol{A}\boldsymbol{s}_{21} + \boldsymbol{s}_{22}$, and send $\boldsymbol{t}_2$ to P_1.

P_1 and P_2 computes $\boldsymbol{t}_{pk} = \boldsymbol{t}_1 + \boldsymbol{t}_2$, respectively. Finally, P_1 stores $\left((\boldsymbol{s}_{11}, \boldsymbol{s}_{12}), \boldsymbol{t}_{pk}, sk_{HE}\right)$, and P_2 stores $\left((\boldsymbol{s}_{21}, \boldsymbol{s}_{22}), \boldsymbol{t}_{pk}, pk_{HE}, evk_{HE}\right)$ (Fig. 1).

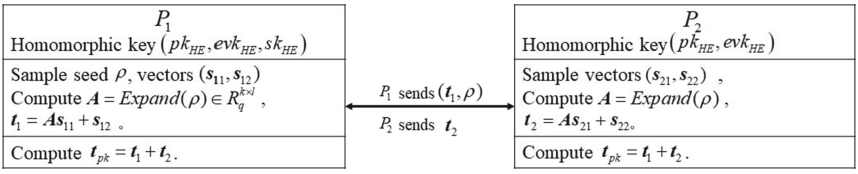

Fig. 1. Distributed secret key generation protocol

3.2 Collaborative Signing Protocol

First, P_1 samples random vector $\boldsymbol{y}_1 \xleftarrow{\$} S_{\frac{\gamma_1}{2}-1}^l$, computes $\boldsymbol{W}_1 = \boldsymbol{A}\boldsymbol{y}_1$, $c_1 = Enc_{pk_{HE}}(\boldsymbol{W}_1)$, and sends c_1 to P_2. P_2 samples random vector $\boldsymbol{y}_2 \xleftarrow{\$} S_{\frac{\gamma_1}{2}-1}^l$, computes $\boldsymbol{W}_2 = \boldsymbol{A}\boldsymbol{y}_2$, $c_2 = c_1 \oplus Enc_{pk_{HE}}(\boldsymbol{W}_2)$, $c_3 = HomEval(evk_{HE}, F_{HighBits_q()}(*, 2\gamma_2), c_2)$, and sends c_3 to P_1.

Second, P_1 decrypts the ciphertext $\boldsymbol{w}_1 = Dec_{sk_{HE}}(c_3) = HighBit_q(\boldsymbol{W}_1 + \boldsymbol{W}_2, 2\gamma_2)$, computes the challenge $c = H(\mu, \boldsymbol{w}_1)$ in which $\mu = CRH(CRH(pk)||M)$, and M is the message to be signed. Then P_1 computes $\boldsymbol{z}_1 = \boldsymbol{y}_1 + c\boldsymbol{s}_{11}$, and verifies whether the following condition is satisfied or not [5]:

$$\|\boldsymbol{z}_1\|_\infty < \frac{\gamma_1}{2} - \beta_1 \tag{1}$$

If the condition is held, P_1 computes $\boldsymbol{u}_1 = \boldsymbol{A}\boldsymbol{y}_1 - c\boldsymbol{s}_{12}$, $c_4 = Enc_{pk_{HE}}(\boldsymbol{u}_1)$, and sends $\boldsymbol{z}_1$, c, c_4 to P_2 If not, P_1 restarts the procedure from sampling random vector $\boldsymbol{y}_1 \xleftarrow{\$} S_{\frac{\gamma_1}{2}-1}^l$.

Similarly, P_2 uses the challenge c to compute $\boldsymbol{z}_2 = \boldsymbol{y}_2 + c\boldsymbol{s}_{21}$, and conducts the verification of the rejection sampling conditions [5]:

$$\|\boldsymbol{z}_2\|_\infty < \frac{\gamma_1}{2} - \beta_1 \tag{2}$$

If the condition is held, P_2 computes $\boldsymbol{u}_2 = \boldsymbol{A}\boldsymbol{y}_2 - c\boldsymbol{s}_{22}$, $c_5 = c_4 \oplus Enc_{pk_{HE}}(\boldsymbol{u}_2)$, $c_6 = HomEval\left(evk_{HE}, F_{\|LowBits_q()\|_\infty}(*, 2\gamma_2), c_5\right)$, and sends $\boldsymbol{z}_2$, c_5 to P_1 If not, P_2 restarts the procedure from sampling random vector $\boldsymbol{y}_2 \xleftarrow{\$} S_{\frac{\gamma_1}{2}-1}^l$.

P_1 utilizes the secret key to decrypt c_6, $\|\boldsymbol{r}_0\|_\infty = \left\|LowBits_q(\boldsymbol{u}_1 + \boldsymbol{u}_2, 2\gamma_2)\right\|_\infty = Dec_{sk_{HE}}(c_6)$. Next, P_1 computes $\boldsymbol{z} = \boldsymbol{z}_1 + \boldsymbol{z}_2 = \boldsymbol{y}_1 + \boldsymbol{y}_2 + c(\boldsymbol{s}_{11} + \boldsymbol{s}_{21})$. The two-party can generate a collaborative signature if and only if the following conditions are held [5] (Fig. 2):

$$\|\boldsymbol{z}\|_\infty < \gamma_1 - \beta_1 \tag{3}$$

$$\|\boldsymbol{r}_0\|_\infty = \left\|LowBits_q(\boldsymbol{A}\boldsymbol{y} - c(\boldsymbol{s}_{12} + \boldsymbol{s}_{22}), 2\gamma_2)\right\|_\infty < \gamma_2 - 2\beta_2 \tag{4}$$

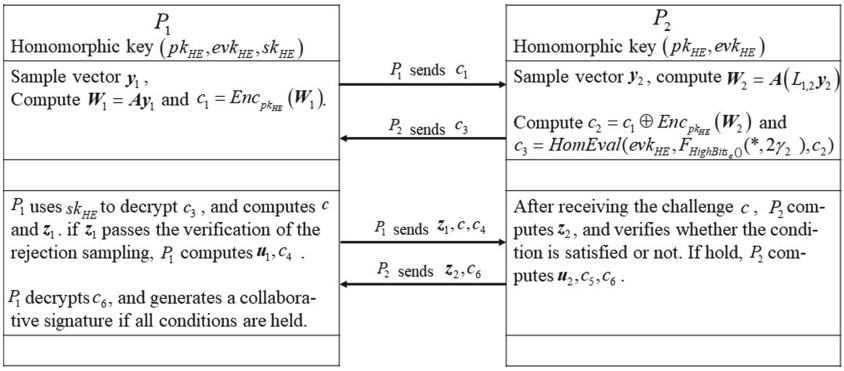

Fig. 2. Collaborative signing protocol

3.3 Correctness

The verifier uses the public key $(\boldsymbol{t}_{pk}, \rho)$ to verify the correctness of the signature $(\boldsymbol{z}, c)$.

There is $\boldsymbol{A}\boldsymbol{z} - c\boldsymbol{t}_{pk} = \boldsymbol{u}_1 + \boldsymbol{u}_2 = \boldsymbol{A}\boldsymbol{y} - c(\boldsymbol{s}_{12} + \boldsymbol{s}_{22})$, based on Lemma 1, from $\|c\boldsymbol{s}_2\|_\infty = \|c(\boldsymbol{s}_{12} + \boldsymbol{s}_{22})\|_\infty \le \|c\boldsymbol{s}_{12}\|_\infty + \|c\boldsymbol{s}_{22}\|_\infty < 2\beta_2$ and $\|\boldsymbol{r}_0\|_\infty < \gamma_2 - 2\beta_2$, we conclude that:

$HighBits_q(\boldsymbol{A}\boldsymbol{z} - c\boldsymbol{t}_{pk}, 2\gamma_2) = HighBits_q(\boldsymbol{W} - c(\boldsymbol{s}_{12} + \boldsymbol{s}_{22}), 2\gamma_2) = HighBits_q(\boldsymbol{W}, 2\gamma_2)$.

The two-party signing protocol and the verifier obtain the same $\boldsymbol{w}_1$, therefore, the verification algorithm always accepts the signature generated by the protocol.

4 Analysis of the Two-Party Aigis-sig Signing Protocol

4.1 Collaborative Signature Generation Probability

To ensure that the output value does not reveal the private key information, the Aigis-sig scheme uses the rejection sampling technology [8]. Therefore, the signing algorithm's efficiency is determined by the number of repetitions caused by steps (1), (2), (3), and (4) of the signing protocol.

Assuming that $\|cs_{11}\|_\infty < \beta_1$ holds, then we always have $\|z_1\|_\infty \leq \frac{\gamma_1}{2} - \beta_1 - 1$ whenever $\|y_1\|_\infty \leq \frac{\gamma_1}{2} - 2\beta_1 - 1$. The size of this range is $\gamma_1 - 2\beta_1 - 1$. Note that each coefficient y_1 is chosen randomly from $\gamma_1 - 1$ possible values. That is, for a fixed cs_{11}, each coefficient of vector $z_1 = y_1 + cs_{11}$ has $\gamma_1 - 1$ possibilities. Thus, the probability that $\|z_1\|_\infty \leq \frac{\gamma_1}{2} - \beta_1 - 1$ is $\left(\frac{\gamma_1 - 2\beta_1 - 1}{\gamma_1 - 1}\right)^{nl} \approx e^{-\frac{2nl\beta_1}{\gamma_1}}$.

Similarly, the probability that $\|z_2\|_\infty \leq \frac{\gamma_1}{2} - \beta_1 - 1$ is $e^{-\frac{2nl\beta_1}{\gamma_1}}$.

If $\|z_1\|_\infty \leq \frac{\gamma_1}{2} - \beta_1 - 1$ and $\|z_2\|_\infty \leq \frac{\gamma_1}{2} - \beta_1 - 1$ hold, that is, $\|z\|_\infty = \|z_1 + z_2\|_\infty \leq \|z_1\|_\infty + \|z_2\|_\infty < \gamma_1 - 2\beta_1 < \gamma_1 - \beta_1$ holds, which implies that if (1) and (2) both hold, then (3) holds.

Assuming that each coefficient r_0 is uniformly distributed modulo $2\gamma_2$. Therefore, the probability that $\|r_0\|_\infty < \gamma_2 - 2\beta_2$ is $\left(\frac{2(\gamma_2 - \beta_2) - 1}{2\gamma_2}\right)^{nk} \approx e^{-\frac{2nk\beta_2}{\gamma_2}}$.

By Lemma 1, if $\|cs_2\|_\infty = \|c(s_{12} + s_{22})\|_\infty \leq \|cs_{12}\|_\infty + \|cs_{22}\|_\infty < 2\beta_2$, then $\|r_0\|_\infty < \gamma_2 - 2\beta_2$ implies that $r_1 = w_1$. This means that the overall probability that the above steps will not cause a repetition is $e^{-2n\left(\frac{2l\beta_1}{\gamma_1} + \frac{k\beta_2}{\gamma_2}\right)}$.

To reduce the number of homomorphic operations and the computational complexity of the protocol, the upper bound of rejection sampling P_1 and P_2 is modified accordingly because of the high cost of homomorphic encryption. First, P_1 receives c_6, to generate a valid signature, the validation of the rejection sampling condition must be performed. If P_1 and P_2 do not carry out the verification of rejection sampling conditions when generating signature portion respectively, that is, there is no restriction of conditions (1) and (2), although the probability of (3) changes from $e^{-\frac{4nl\beta_1}{\gamma_1}}$ to $e^{-\frac{2nl\beta_1}{\gamma_1}}$, homomorphic operations will be performed in the subsequent process. Second, even if P_1 and P_2 perform the verification of condition (4) respectively, although $u = u_1 + u_2$ holds, $\|LowBits_q(u, 2\gamma_2)\|_\infty = \|LowBits_q(u_1, 2\gamma_2)\|_\infty + \|LowBits_q(u_2, 2\gamma_2)\|_\infty$ maybe not hold.

4.2 Feasibility Analysis

The Cheon, Kim, Kim, and Song (CKKS) homomorphic encryption scheme is introduced into the collaborative signing protocol [6]. First, to ensure the security of transmission share; second, compared with other homomorphic encryption schemes, the CKKS scheme allows approximate encryption calculation for real or complex numbers, and supports homomorphic ciphertext comparison algorithm [14].

This section mainly analyzes whether the homomorphic operations can be realized through several homomorphic addition operations, homomorphic multiplication operations, and homomorphic ciphertext comparison operations.

$c_2 = c_1 \oplus Enc_{pk_1}(W_2)$ can be achieved by a ciphertext homomorphic addition operation. As for $c_3 = HomEval(evk_{HE}, F_{HighBits_q()}(*, 2\gamma_2), c_2)$, when calculating $HighBits_q(c_2, 2\gamma_2)$ in ciphertext state, we need to invoke the $Decompose_q()$ algorithm on each coefficient of the polynomial vector. Specifically, for each coefficient, seven additions, two multiplications, and five comparison operations are required. Correspondingly,

we transform these operations into the corresponding homomorphic ciphertext operations, namely, the corresponding number of ciphertext homomorphic addition, ciphertext homomorphic multiplication, and ciphertext comparison operations.

$c_5 = c_4 \oplus Enc_{pk_{HE}}(\boldsymbol{u}_2)$ can be achieved by a ciphertext homomorphic addition operation as well. In the process of calculating $c_6 = HomEval(evk_{HE}, F_{\|LowBits_q()\|_\infty}(*, 2\gamma_2), c_5)$, we divide it into two steps. We compute $LowBits_q(c_5, 2\gamma_2)$ at first. when calculating it in ciphertext state, we need to invoke the $Decompose_q()$ algorithm on each coefficient of the polynomial vector as well. When it comes to computing $\|LowBits_q(c_5, 2\gamma_2)\|_\infty$ in ciphertext state, we introduce the method of calculating the maximum value of homomorphic ciphertext proposed by Cheon et al. [14]. We define a set $\Omega = \{a_1, a_2, \ldots, a_e\}$, α is precision bits, and d its iterations. The parameter d can be calculated by e and α. The time complexity of the algorithm is $\Theta(ed)$. The experimental results in reference [14] show that the maximum value of 2^{16} 32-bit integers can be solved out within 75 s. It is estimated that in our protocol, it takes about 29 s to obtain the decryption state function $\|\|_\infty$ when the parameter is set as parameter II of the Aigis-sig scheme.

4.3 Security Analysis

In this section, based on the security of the Aigis-sig scheme and CKKS scheme, we prove that Π is a secure two-party signing protocol.

Theorem 3.3.1. Assumeing that the CKKS homomorphic encryption scheme is indistinguishable against chosen-plaintext attacks [6] and Aigis-sig is SUF-CMA [5]. Then, in the random oracle model, the two-party Aigis-sig signing protocol is existentially unforgeable against a chosen-message attack (Fig. 3).

Fig. 3. The security reduction from Π to π

Proof. Because we define that both participants are honest, the adversary $\mathcal{A}$ cannot corrupt either party or participate in the protocol interaction. However, the adversary $\mathcal{A}$ can obtain the public key $(\boldsymbol{t}_{pk}, \rho)$ and signature $(\boldsymbol{z}, c)$ and have a view of the protocol interaction. As a result, if an adversary $\mathcal{A}$ attacks the protocol, and forges a cooperative signature with nonnegligible probability in experiment 3.2 by using public key, signature and the middle value of the protocol, we can construct a simulator $\mathcal{S}$ that forges an Aigis-sig signature with nonnegligible probability in experiment 3.1. Formally, for any PPT adversary $\mathcal{A}$, there exists a PPT adversary $\mathcal{S}$ and a negligible function λ'' such that, for any κ, it holds that:

$$\left|Pr[Expt - Sign_{\mathcal{S},\pi}(1^\kappa) = 1] - Pr[Expt - DistSign_{\mathcal{A},\Pi}(1^\kappa) = 1]\right| \leq \lambda''(\kappa)$$

In combination with the Definition 1.3.1, it holds that $\left|Pr[Expt - Sign_{\mathcal{S},\pi}(1^{\kappa}) = 1]\right| \leq \lambda(\kappa)$, we conclude that:

$$\left|Pr[Expt - DistSign_{\mathcal{A},\Pi}(1^{\kappa}) = 1]\right| \leq \lambda(\kappa) + \lambda''(\kappa),$$

According to Definition 1.3.2, Π is a secure two-party Aigis-sig signing protocol.

5 Conclusion

In this paper, a new postquantum two-party cooperative digital signature protocol based on the Aigis-sig scheme is presented, which is composed of distributed key generation and collaborative signing protocol. CKKS homomorphic encryption is used in the protocol to provide a secure cooperative signature. However, under the condition of the current parameter setting, the probability of passing rejection sampling in the signature generation is relatively small, resulting in a large number of running times. The focus of our future work will be on how to improve the efficiency of the two-party cooperative signature generation scheme.

References

1. Lindell, Y.: Fast secure two-party ECDSA signing. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10402, pp. 613–644. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63715-0_21
2. Wang, J., Wu, L., Luo, M., et al.: Secure and efficient two-party ECDSA signature scheme. J. Commun. **42**(2), 12–25 (2021)
3. Lindell, Y., Ariel, N.: Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 1837–1854. ACM, New York (2018)
4. Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., et al.: CRYSTALS-dilithium: a lattice-based digital signature scheme. IACR Trans. Cryptogr. Hardw. Embed. Syst. **1**, 238–268 (2018)
5. Zhang, J., Yu, Y., Fan, S., Zhang, Z., Yang, K.: Tweaking the asymmetry of asymmetric-key cryptography on lattices: KEMs and signatures of smaller sizes. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020. LNCS, vol. 12111, pp. 37–65. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45388-6_2
6. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10624, pp. 409–437. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_15
7. Lyubashevsky, V.: Fiat-Shamir with aborts: applications to lattice and factoring-based signatures. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 598–616. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10366-7_35
8. Lyubashevsky, V.: Lattice signatures without trapdoors. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 738–755. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_43
9. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the Annual ACM Symposium on Theory of Computing, pp. 169–178. ACM, MD (2009)

10. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical GapSVP. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 868–886. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_50
11. Brakerski, Z., Gentry, C., Vaikuntanathan, V., et al.: (Leveled) fully homomorphic encryption without bootstrapping. ACM Trans. Comput. Theory (TOCT) **6**(3), 1–36 (2014)
12. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 75–92. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_5
13. Ducas, L., Micciancio, D.: FHEW: bootstrapping homomorphic encryption in less than a second. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 617–640. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_24
14. Cheon, J.H., Kim, D., Kim, D., Lee, H.H., Lee, K.: Numerical method for comparison on homomorphically encrypted numbers. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019. LNCS, vol. 11922, pp. 415–445. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34621-8_15