ThAigis-2P: A Two-Party Threshold Signature Scheme for Aigis-sig

Li Wang[1,*] Tihong Qin[2]

[1] School of Computer and Information Technology, Anhui University of Applied Technology, Hefei 230000, China

[2] Beijing Infosec Technologies Co., Ltd., Beijing 100000, China

* Corresponding author: andy@uta.edu.cn

Abstract

Aigis-sig uses Fiat–Shamir with aborts. Its rejection step depends on the full response, which complicates threshold signing. We present ThAigis-2P, a two-party threshold signature scheme whose outputs are accepted by the unmodified Aigis-sig verifier. The parties sample their secret shares independently from the Aigis-sig key distribution, subject to a spectral screening condition. Each party samples masks from a scaled hyperball and performs rejection sampling locally. The protocol runs K candidates per attempt. The spectral screening condition bounds the secret shift for every challenge and contributes approximately one bit of conditioning loss to the security reduction. The protocol has a three-round interactive mode and a preprocessing mode that exchanges commitments offline and requires one online round trip per attempt using committed salts and single-use records. In the random oracle model, the security analysis relates unforgeability to AMLWE under the Aigis-sig key distribution, MLWE under hyperball distributions, and the unforgeability of native Aigis-sig. The preprocessing mode also assumes a rollback-resistant state service. Experiments with a single-process C reference implementation based on PQMagic yield success probabilities of 0.22–0.25 per candidate and above 0.92 per attempt with K = 10. All 39,000 generated threshold signatures pass Aigis-sig verification. Canonical lossless encoding reduces partial response sizes by 21%–25% relative to transmitting raw coefficients. The evaluation covers all three Aigis-sig parameter sets and measures computation in a single-process prototype.

Keywords: Aigis-sig, threshold signatures, rejection sampling, hyperball distributions, spectral checks, lossless coding

1 Introduction

1.1 Research Background

Threshold signatures distribute the signing key among multiple participants, and only when a threshold number of participants cooperate can a signature be generated; the verifier still sees a single

public key and an ordinary signature (Shoup 2000). In this way, the complete private key is no longer stored centrally on a single device, making it suitable for scenarios such as digital asset custody, certificate issuance, and collaborative identity authentication. The two-party case is very common: the user terminal and the server each hold one key share, and compromising either party alone is insufficient to forge a signature. Two-party and multiparty protocols based on ECDSA and Schnorr signatures are already quite mature (Lindell 2017; Canetti et al. 2020; Komlo & Goldberg 2021), and the National Institute of Standards and Technology (NIST) has also launched a call for multiparty threshold schemes (Brandão & Peralta 2026).

Progress in quantum computing motivates the migration of signature schemes to post-quantum algorithms. Fiat–Shamir with aborts (Lyubashevsky 2009, 2012) is a widely used framework for signatures from lattices; ML-DSA (NIST 2024), the NIST standard derived from Dilithium (Ducas et al. 2018), follows this framework. Aigis-sig (Zhang et al. 2020) has a structure like to Dilithium and uses asymmetric variants of the module-LWE and module-SIS problems, denoted by AMLWE and AMSIS. Its two secret components have different coefficient ranges, allowing shorter public keys and signatures for the same security target. The open-source cryptographic library PQMagic (PQCrypto-CN 2026) implements Aigis-sig. We call an output a native signature if it is accepted by the unmodified Aigis-sig verification algorithm. This compatibility preserves the verification algorithm and public key format.

Extending Fiat–Shamir with aborts to threshold signing is difficult primarily because of rejection sampling. The signer samples a mask y and computes the commitment $w = Ay$. It computes the response $z = y + cs_1$ from the challenge c and uses z to decide whether to output a signature or resample. In signing by a single party, rejected attempts are never revealed. A direct extension to multiple parties requires the parties to exchange commitments before deriving a common challenge, whereas the rejection decision depends on the aggregate response. Revealing commitments and partial responses from rejected attempts may therefore leak information about the secret. A second difficulty is the hint. The Aigis-sig public key contains only the high bits of $t = As_1 + s_2$, and the verifier uses the hint h to recover the high bits of the commitment. The threshold protocol must generate a valid hint without reconstructing the private key and must ensure that the aggregate response satisfies the Aigis-sig norm bounds.

1.2 Prior Work and Analysis

Prior work includes generic secure multiparty computation, constructions based on noise flooding, and constructions that use local rejection sampling.

Generic secure multiparty computation and earlier two-party Aigis-sig schemes. Cozzo & Smart (2019) identify costly conversions between MPC representations when applying generic techniques to lattice-based signatures. Trilithium (Dufka et al. 2025) supports two-party ML-DSA signing with the assistance of a correlated-randomness provider. Quorus (Bienstock et al. 2026) gives scalable threshold ML-DSA using MPC, with an honest-majority instantiation and offline preprocessing. For Aigis-sig, Fu, and Zhao (Fu & Zhao 2022) use homomorphic encryption to protect intermediate values in a two-party signing protocol. Wen et al. (2023) propose Aitps, an Aigis-sig-based two-party signature scheme that uses additive homomorphic commitments.

Threshold signatures using noise flooding. Threshold Raccoon (del Pino et al. 2024) uses wide

noise to mask secret-dependent terms and reports support for up to 1,024 signers with signatures of approximately 13 KB (1 KB = 1,000 bytes). It uses its own signature format and verification procedure. Related lattice-based threshold signatures address issues such as reducing the number of rounds, distributed key generation, robustness, and adaptive security (Boschini et al. 2025; Espitau et al. 2024; Katsumata et al. 2024).

Threshold signatures using local rejection sampling. Early lattice-based multisignatures and n-out-of-n signatures (Damgård et al. 2021; Boschini et al. 2022) use tools such as trapdoor commitments to hide information revealed during aborted attempts. Finally! (del Pino & Niot 2025) uses short, replicated secret sharing so that each party can perform rejection sampling locally, and its security analysis treats the simulation of rejected transcripts. Building on this approach, Celi et al. (2026) give a threshold signature scheme compatible with ML-DSA. The parties use imbalanced rejection sampling over hyperballs, check the response norm and hint during combination, and run several candidates in parallel. The published parameter sets cover up to six parties. Rambaud et al. (2026) study threshold signatures as a service and describe an ML-DSA construction that moves commitment exchange offline, leaving one online round under their service model. ThAigis-2P adapts this preprocessing approach to two signers using commitments to salts and single-use records, as specified in Section 4.6.

Applying local rejection sampling to Aigis-sig with two parties requires addressing four issues. First, γ_1 equals 2^{17} in all three Aigis-sig parameter sets, whereas γ_2 is much larger than the corresponding value in ML-DSA-44. The two secret components also have different coefficient ranges. The relative scaling of the two hyperball components must therefore be chosen to satisfy the Aigis-sig verification conditions. Second, rejection sampling over hyperballs requires the secret shift $\| c\mathbf{s} \|_2$ to be bounded by B. Celi et al. (2026) choose B heuristically as 1.3 times the typical value. Using instead a bound based on the ℓ_1 norm that holds for every valid secret reduces the local acceptance probability below 0.6% and the success probability of one candidate to the order of 10^{-5}, as shown in Section 6.3. Third, the client and relayer roles in the service model must be assigned to the two signers, requiring a corresponding preprocessing protocol and security proof. Fourth, the coefficients of partial responses typically have magnitudes far below $q/2$, so representing each coefficient using a fixed width of $\lceil \log_2 q \rceil$ bits wastes bandwidth.

1.3 Contributions

We propose ThAigis-2P, a two-party threshold signature scheme. The parties sample their shares independently, and the joint public key has the same format as an Aigis-sig public key. During signing, each party performs rejection sampling over hyperballs locally and sends only the first response component of each accepted candidate. The party combining the responses, referred to as the combiner, generates the native hint from public values and outputs the signature. The main contributions are as follows.

(1) A two-party threshold signature scheme. The key shares can be generated by a trusted dealer or jointly by the two parties. The signing protocol provides an interactive mode with three rounds and a preprocessing mode. In the preprocessing mode, both parties commit to their random salts in advance, keeping the challenge unpredictable until the request is authorized. The offline phase requires one round of interaction, and the online phase requires one round trip.

(2) A shift bound valid for every challenge. During share generation, we compute a spectral statistic of the secret and resample if it exceeds a threshold. HAETAE (Cheon et al. 2024) uses a related form of key rejection. We use a maximum of scaled per-frequency energies to screen signing shares. By Parseval's identity, the resulting bound B holds for every element of the challenge set and is approximately 1.47–1.55 times the typical shift. The spectral screening condition contributes a conditioning loss of $\log_2(1/p_E)$ bits in the security reduction, with $p_E \approx 1/2$ in our setting. The reduction still relies on the AMLWE assumption under the Aigis-sig key distribution.

(3) Security analysis. We derive concrete unforgeability bounds in the random oracle model for the interactive mode and for the preprocessing mode under the specified message authorization and state assumptions. The treatment of commitments from rejected candidates follows the simulation approach of (del Pino & Niot 2025).

(4) Parameters, encoding, and implementation. We select the hyperball parameters (ν, r, r') and the number of candidates K for each of the three Aigis-sig parameter sets, and estimate their success probabilities, communication and storage costs, and security levels. We specify a canonical lossless encoding with N, R, and C modes and implement the protocol algorithms using PQMagic. Experiments yield a success probability of 0.22–0.25 per candidate, while the encoding reduces partial response sizes by 21%–25% relative to transmitting raw coefficients.

1.4 Paper Structure

Section 2 outlines the design rationale of the scheme. Section 3 introduces the notation, Aigis-sig, hard problems, Rényi divergence, hyperball rejection sampling, and the security model for two-party threshold signatures. Section 4 presents the key generation, the two signing modes, the combining algorithm, state management, and the encoding format of ThAigis-2P. Section 5 proves correctness and unforgeability. Section 6 analyzes the parameters for each set. Section 7 reports the reference implementation and experimental results. Section 8 concludes the paper. Appendix A specifies the message formats.

2 Scheme Overview

This section outlines the design rationale of ThAigis-2P. A rigorous description is provided in Section 4, and the proofs are given in Section 5.

Design Goals. Two parties, P_0 *and* P_1, jointly hold the private key corresponding to a single Aigis-sig public key. The adversary statically corrupts at most one party before key generation and fully controls the network: it can delay, reorder, or drop messages, and can choose its messages after seeing those of the honest party. The protocol allows aborts and does not guarantee signature generation under malicious behavior. However, the adversary cannot obtain a valid signature on a message not authorized by the honest party, and all outputs must be native signatures.

Shares and public key. The share $u_i = \left(s_{1,\,i}, s_{2,i}\right)$ of party P_i is sampled independently from the uniform distribution used in Aigis-sig key generation, referred to here as the native distribution, and is then subjected to the spectral check described below. The joint secret is $(s_1, s_2) = u_0 + u_1$, and the joint public key is formed from the high bits of $t = As_1 + s_2$. This is the special case of

T-out-of-N replicated secret sharing (del Pino & Niot 2025; Celi et al. 2026) with $T = N = 2$. Given the corrupted party's share, the honest party's public key component $t^{(h)} = [A \mid I]u_h$ can be replaced by a uniform value in the proof under the AMLWE assumption. The coefficient range of the joint secret doubles, affecting the signing success probability while leaving the verification conditions unchanged. The estimates in Section 6.4 indicate a higher cost of recovering the private key from the joint public key.

Local rejection and hint. Each party samples its mask $y_i = \lfloor L_\nu \xi_i \rceil$ from a scaled hyperball, where ξ_i is uniformly distributed in a D-dimensional Euclidean ball and $L_\nu = diag(\nu I_{nl}, I_{nk})$. Unlike in Aigis-sig, the commitment $w_i = Ay_i^{(1)} + y_i^{(2)}$ includes the second component. After obtaining the common challenge c, party P_i checks whether $\xi_i + L_\nu^{-1}(cu_i)$ lies in a ball of radius r. If the candidate is accepted, the party sends $z_i^{(1)} = y_i^{(1)} + cs_{1,\,i}$. Each party makes this decision locally. The combiner computes $z = z_0^{(1)} + z_1^{(1)}$ and obtains $\delta = w - (Az - 2^d c\, t_1) = z^{(2)} - c\, t_0$ from public values, then invokes the native MakeHint algorithm to generate the hint. Combining requires neither $\mathbf{t}_0$ nor disclosure of the second response component. The parameter ν controls the relative sizes of the two components: the first is constrained by $\| z \|_\infty < \gamma_1 - \beta_1$, while the second affects the hint weight through δ. Parameter set I uses $\nu > 1$, whereas sets II and III use $\nu < 1$, as explained in Section 6.2.

Shift Bound. The statistical properties of hyperball rejection sampling require $\| L_\nu^{-1}(cu_i) \|_2 \leq B$. By Parseval's identity,

$$\| L_\nu^{-1}(cu) \|_2^2 = \frac{1}{n}\sum_{j=0}^{n-1} |c(\zeta_j)|^2\, S_j(u) \leq \max_j S_j(u) \cdot \frac{1}{n}\sum_{j=0}^{n-1} |c(\zeta_j)|^2 = \tau \max_j S_j(u),$$

where $S_j(u)$ is the weighted sum of squared magnitudes of the component polynomials evaluated at the root of unity ζ_j. During share generation, the share is resampled if $\max_j S_j(u)$ exceeds the threshold S_{thr}. Hence, $\| L_\nu^{-1}(cu) \|_2 \leq B = \sqrt{\tau S_{thr}}$ holds for every challenge. The threshold is the median of this statistic under the native distribution. On average, about two samples are needed to obtain a share, taking less than 1 ms in the reference implementation. Conditioning changes the share distribution. In the proof, the honest party's response is first replaced by a simulated value independent of the secret. The game then depends on the share only through the public key component. Probability preservation for Rényi divergence allows the proof to return to the native distribution, with a loss factor of $1/p_E \approx 2$.

Parallel candidates and two signing modes. The probability that both parties accept a candidate locally is approximately $1/M^2 \approx 1/4$, after which the combined response must pass the norm and hint checks. As in (Celi et al. 2026), each signing attempt runs K candidates in parallel and selects the first successful one. The interactive mode follows the three rounds of (Celi et al. 2026): exchange commitment digests, reveal commitments, and send responses. In the preprocessing mode, commitments are exchanged offline, and each record includes both parties' commitments to their random salts. Online, the initiator sends the message and its salt. The responder atomically marks the record as used before revealing its salt and returning its partial responses. The XOR of the salts forms the signature context string and enters the message representative μ. If either party is honest, the challenge remains unpredictable until authorization.

Response encoding. The coefficients of partial responses are approximately normally distributed,

with standard deviations near 2×10^4, well below $q/2$. The encoder selects the shortest of three representations: narrow fixed-width encoding (N), raw coefficients (R), and blockwise truncated Rice encoding (C), based on Golomb–Rice codes (Golomb 1966; Rice & Plaunt 1971). The decoder re-encodes each decoded vector and requires an exact byte-for-byte match with the input. The encoding is a deterministic function of the partial responses already disclosed by the protocol.

3 Preliminaries

3.1 Notation

Let $n = 256$, $R = \mathbb{Z}[X]/(X^n + 1)$, and $R_q = R/qR$. Bold lowercase letters denote vectors over R or R_q, and bold uppercase letters denote matrices. The coefficient embedding identifies elements of R^m with vectors in $\mathbb{Z}^{nm}$. For norms, coefficients in R_q are taken as centered representatives in $[-(q-1)/2, (q-1)/2]$. We write $\| \cdot \|_2$ and $\| \cdot \|_\infty$ for the Euclidean and infinity norms, respectively, and $wt(h)$ for the number of ones in a binary vector. Let S_η be the set of polynomials whose coefficients have absolute values at most η, and let $U(S)$ be the uniform distribution over S. The notation $x := f(\cdot)$ denotes a deterministic assignment, and $x \leftarrow \mathcal{D}$ denotes sampling from $\mathcal{D}$. Let $[K] = \{0, 1, \ldots, K-1\}$, and write $\lfloor \cdot \rceil$ for coordinatewise rounding to the nearest integer. We write Π for a proposition and $[\Pi]$ for its indicator, which equals 1 when Π holds and 0 otherwise. The message is denoted by m. The challenge set C_τ consists of polynomials with exactly τ nonzero coefficients in $\{\pm 1\}$. Aigis-sig uses $\tau = 60$. For any $c \in C_\tau$, we have $\| c \|_1 = \tau$ and $\| c \|_2 = \sqrt{\tau}$.

Let $\zeta_j = exp(i\pi(2j+1)/n)$ for $j \in [n]$; these are all the complex roots of $X^n + 1$. For $f \in R$, $f(\zeta_j)$ denotes the value of f at ζ_j; since the coefficients of f are real, we have $f(\zeta_{n-1-j}) = \overline{f(\zeta_j)}$. Negacyclic convolution satisfies $(f\ g)\ (\zeta_j) = f(\zeta_j) g(\zeta_j)$, and the Parseval identity holds:

$\| f \|_2^2 = \frac{1}{n} \sum_{j \in [n]} |f\left(\zeta_j\right)|^2$.

For dimension D and radius $r > 0$, let $B_D(r) = \{x \in \mathbb{R}^D : \|x\|_2 \le r\}$ and $B_D(r,\ v)\ = v + B_D(r)$. Let $I_x(a, b)$ denote the regularized incomplete beta function. The statistical distance between two distributions is denoted by $\Delta\ (\cdot,\ \cdot)$.

3.1 Aigis-sig

The public parameters of Aigis-sig (Zhang et al. 2020) include the modulus q, matrix dimensions (k, l), the number d of bits dropped from the public key, secret ranges (η_1, η_2), mask range γ_1, high/low bit decomposition parameter γ_2, verification bounds (β_1, β_2), upper bound on the hint weight ω, and challenge weight τ. All three parameter sets satisfy $q - 1 = 6 \bullet 2\gamma_2$, so the high bits take values in $\{0, \ldots, 5\}$. *ExpandA* uses SHAKE-128 to expand a seed into a matrix $A \in R_q^{k \times l}$ in the NTT domain, CRH uses SHAKE-256 to output a 48-byte digest, and H_c maps the 48-byte message representative and the packed high bits to a challenge in C_τ. By default, PQMagic (PQCrypto-CN 2026) instantiates these hash functions using the SM3 extension function, but SHAKE can also be selected at compile time; the signer and verifier must use the same instantiation. The PQMagic implementation supports a context string ctx of up to 255 bytes. The message representative is defined as

$$\mu = CRH(tr \parallel m'), m' = 0x00 \parallel |ctx| \parallel ctx \parallel m, tr = CRH(pk). \quad (1)$$

When ctx is empty, this is equivalent to signing a message with the fixed prefix $0x00 \parallel 0x00$. Algorithm 1 presents the core steps of key generation, signing, and verification, where the rounding functions are detailed in Section 3.3. The challenge in the signature is encoded as the polynomial itself (a 32-byte position mask followed by 8 bytes of signs), and verification compares the recomputed challenge with the one in the signature.

Algorithm 1 Key generation, signing, and verification of Aigis-sig.

KeyGen ():

1: $\rho \leftarrow \{0,1\}^{256}, A := ExpandA(\rho)$; $s_1 \leftarrow U(S_{\eta_1}^l)$, $s_2 \leftarrow U(S_{\eta_2}^k)$

2: $t := As_1 + s_2$, $(t_1, t_0) := Power2Round_q(t, d)$

3: $pk := (\rho, t_1)$, $tr := CRH(pk)$, $sk := (\rho, tr, s_1, s_2, t_0)$

Sign (sk, m, ctx):

4: Compute μ according to Eq. (1); repeat

5: $y \leftarrow S_{\gamma_1 - 1}^l$, $w := Ay$, $w_1 := HighBits_q(w, 2\gamma_2)$, $c := H_c(\mu, w_1)$, $z := y + cs_1$

6: if $\| z \|_\infty \geq \gamma_1 - \beta_1$, $\| LowBits_q(w - cs_2, 2\gamma_2) \|_\infty \geq \gamma_2 - \beta_2$ or $HighBits_q(w - cs_2, 2\gamma_2) \neq w_1$, then continue

7: if $\| ct_0 \|_\infty \geq \gamma_2$ then continue; $h := MakeHint_q(w - cs_2 + ct_0, \ -ct_0, \ 2\gamma_2)$; if $wt(h) \leq \omega$ then return $\sigma := (z, h, c)$

Verify (pk, m, σ, ctx):

8: Canonically decode $\sigma = (z, h, c)$; require $\| z \|_\infty < \gamma_1 - \beta_1$ and that h contains at most ω ones.

9: Compute $\mu \ accordingtoEq. (1)$; $w_1' := UseHint_q(h, Az - 2^d c \ t_1, 2\gamma_2)$; accept it and only if $c = H_c(\mu, w_1')$

Table 1 summarizes the three parameter sets and the corresponding key and signature sizes.

Table 1 Parameters and sizes of Aigis-sig

Parameter	**I**	**II**	**III**
Modulus q	2 021 377	3 870 721	3 870 721
Dropped bits d	13	14	14
Matrix dimensions (k, l)	(4, 3)	(5, 4)	(6, 5)
Secret ranges (η_1, η_2)	(2, 3)	(2, 5)	(1, 5)
Verification bounds (β_1, β_2)	(120, 175)	(120, 275)	(60, 275)
γ_2	168 448	322 560	322 560
Hint weight upper bound ω	80	96	120
Public key / Secret key / Signature sizes (bytes)	1056 / 2448 / 1852	1312 / 3376 / 2445	1568 / 3888 / 3046

Note: All three parameter sets use $n = 256$, $\gamma_1 = 2^{17}$, and $\tau = 60$.

3.3 Rounding and Hints

For $r \in Z_q$, $Power2Round_q(r,\ d)$ outputs (r_1, r_0), where r_0 is the centered remainder of r modulo 2^d, and $r_1 = (r - r_0)/2^d$. Let $\alpha = 2\gamma_2$. $Decompose_q(r, \alpha)$ first sets $r_0 = rmod^{\pm}\alpha \in (-\alpha/2, \alpha/2]$; if $r - r_0 = q - 1$, then set $r_1 = 0$ and $r_0 \leftarrow r_0 - 1$; otherwise, $r_1 = (r - r_0)/\alpha$. $HighBits_q$ and $LowBits_q$ return r_1 and r_0, respectively. The hint function is defined as

$$MakeHint_q(r, z, \alpha) = [HighBits_q(r, \alpha) \neq HighBits_q(r + z, \alpha)].$$

$UseHint_q(h, r, \alpha)$ returns r_1 when $h = 0$. When $h = 1$, if $r_0 > 0$, increment r_1 cyclically over $\{0, \ldots, 5\}$; otherwise, decrement it cyclically. These functions are applied coefficientwise to vectors.

Lemma 1 (Correctness of the Hint; Ducas et al. 2018; Zhang et al. 2020). Let α be even, suppose α divides $q - 1$, and assume $q > 2\alpha$. For any $r, z \in Z_q$, if $\| z \|_\infty \leq \alpha/2$, then $UseHint_q(MakeHint_q(r, z, \alpha), r, \alpha) = HighBits_q(r + z, \alpha)$.

3.2 Hard Problems and Signature Security

Definition 1 (MLWE and AMLWE; Zhang et al. 2020). Let χ be a distribution over $R^l \times R^k$. The distinguishing advantage of algorithm A for $MLWE_{q,k,l,\chi}$ is defined as

$$Adv_{\chi}^{MLWE}(A) = |\Pr[A(A, [A \mid I]x) = 1] - \Pr[A(A, u) = 1]|,$$

where $A \leftarrow U(R_q^{k \times l})$, $x \leftarrow \chi$, and $u \leftarrow U(R_q^k)$. The distribution $\chi_s = U(S_{\eta_1}^l) \times U(S_{\eta_2}^k)$ used for Aigis-sig key generation is called the native distribution, and the problem when $\chi = \chi_s$ is known as AMLWE.

This paper also employs two ball distributions. Given a scaling factor $\nu > 0$, a sampling radius r', and a target radius r ($r' > r$; see Section 6 for values), let $L_\nu = diag(\nu I_{nl}, I_{nk})$ and $D = n(k + l)$, and define

$$\chi_r = \{\lfloor L_\nu \xi \rceil : \xi \leftarrow U(B_D(r'))\},\ \chi_z = \{\lfloor L_\nu \xi \rceil : \xi \leftarrow U(B_D(r))\} \quad (2)$$

Under the parameters of this paper, the standard deviation of the coordinates of these two distributions is approximately 1.6×10^4 to 3.0×10^4. The ratio of the standard deviation to the modulus is far greater than that in common LWE instances; the analysis of the corresponding MLWE problem is presented in Section 6.4.

The analysis in this paper is conducted in the random oracle model (Bellare & Rogaway 1993). Aigis-sig satisfies strong unforgeability in this model, and its security is based on AMLWE and SelfTargetAMSIS (Zhang et al. 2020; Kiltz et al. 2018). The reduction relies only on the following weaker property.

Definition 2 (Unforgeability under Key-Only Attacks). An adversary is given the Aigis-sig public key pk, has access to a random oracle, and outputs (m^*, ctx^*, σ^*). The adversary wins if $Verify(pk, m^*, \sigma^*, ctx^*) = 1$. Its winning probability is denoted by Adv_{Aigis}^{UF}, which does not exceed the forgery advantage under a chosen-message attack.

3.3 Rényi Divergence

Definition 3 (Rényi Divergence; Bai et al. 2015; Devevey et al. 2022). Let distribution D_1 be continuous with respect to D_2. At order infinity, the Rényi divergence of D_1 relative to D_2 is defined as $R_\infty(D_1 \parallel D_2) = \sup_x D_1(x)/D_2(x)$, where the supremum is the essential supremum. For any two distributions and $\varepsilon \in [0, 1)$, the ε-smooth Rényi divergence of infinite order is defined as

$$R^\varepsilon_\infty(D_1 \| D_2) = \inf\{M > 0 : \Pr_{x \leftarrow D_1}[D_1(x) \le M\, D_2(x)] \ge 1 - \varepsilon\}$$

Lemma 2 (Properties of Rényi Divergence; Bai et al. 2015). (1) (Data Processing Inequality) For any function $f, R_\infty(f(D_1) \parallel f(D_2)) \le R_\infty(D_1 \parallel D_2)$. (2) (Multiplicativity) Let D_1 and D_2 be two joint distributions of the sequence of random variables $(x_i)_i$. If, for any prefix $v_{<i}$, the conditional distributions of the i-th component conditioned on $x_{<i} = v_{<i}$ satisfy $R_\infty \le r_i$, then $R_\infty(\mathcal{D}_1 \parallel \mathcal{D}_2) \le \prod_i r_i$. (3) (Probability Preservation) For any event E, $D_2(E) \ge D_1(E)/R_\infty(D_1 \parallel D_2)$.

3.6 Rejection Sampling over Hyperballs

Given an integer shift v, a source distribution χ_r, a target distribution χ_z, and a constant $M > 1$, the rejection sampling algorithm $Rej(v, \chi_z, \chi_r, M)$ samples $y \leftarrow \chi_r$, sets $z = y + v$, and outputs z with probability $\min\{\chi_z(z)/(M\chi_r(y)), 1\}$; otherwise, it outputs $\perp$. The ideal algorithm $Ideal(\chi_z, M)$ outputs a fresh sample $z \leftarrow \chi_z$ with probability $1/M$; otherwise, it outputs $\perp$. Its output is independent of the secret. Let $(z \mid acc)_v$ and $(y \mid rej)_v$ denote the distributions of z conditioned on acceptance and y conditioned on rejection, respectively.

Lemma 3 (Rejection Sampling; del Pino & Niot 2025; Devevey et al. 2022). If $R^\varepsilon_\infty(\chi_z \parallel v + \chi_r) \le M$, then $R_\infty(Rej \parallel Ideal) \le 1 + \varepsilon/(M-1)$, the acceptance probability lies between $(1-\varepsilon)/M$ and $1/M$, and $\Delta((z \mid acc)_v, \chi_z) \le \varepsilon$.

The first two conclusions can be found in (del Pino & Niot 2025; Devevey et al. 2022). For the third conclusion, let $f(z) = \min\{\chi_z(z), M\chi_r(z-v)\}$ and $a = \sum_z f(z)$; the condition can be written as $a \ge 1 - \varepsilon$. When acceptance occurs, the distribution of z is f/a, so $\sum |f/a - \chi_z| \le \sum |f/a - f| + \sum |f - \chi_z| = 2(1-a)$, meaning the statistical distance does not exceed $1 - a \le \varepsilon$.

When χ_r and χ_z are uniform distributions over balls of radii r' and r, respectively, the acceptance condition reduces to $\| z \|_2 \le r$, so the density ratio need not be computed. This is known as hyperball rejection sampling (Celi et al. 2026; Devevey et al. 2022). The following lemma relates the coverage condition to smooth Rényi divergence.

Lemma 4 (Hyperball Coverage; Celi et al. 2026; Devevey et al. 2022). Let $\varphi \ge 1$ and $\varepsilon \in [0, 1/2)$ satisfy $2\varepsilon = I_{1-1/\varphi^2}(\frac{D+1}{2}, \frac{1}{2})$. If $r'^2 \ge r^2 + \| v \|_2^2 + 2r \| v \|_2 / \varphi$, then

$$R^\varepsilon_\infty(U(B_D(r)) \parallel U(B_D(r', v))) \le M := (r'/r)^D.$$

By Definition 3, the conclusion of Lemma 4 is equivalent to at least a fraction $1 - \varepsilon$ of the volume of $B_D(r)$ lying inside $B_D(r', v)$. Increasing φ lowers the required bound on r' for a fixed target radius r and shift, which can increase the acceptance probability. It also increases ε.

del Pino & Niot (2025) prove that commitments of rejected candidates can be simulated. Lemma 5 is a formulation of this conclusion in the context of $[A \mid I]$.

Lemma 5 (Commitments of Rejected Candidates; del Pino & Niot 2025; Celi et al. 2026) Let v be a fixed integer shift. Let p_{rej} denote the rejection probability of $Rej(v, \chi_z, \chi_r, M)$, and let $\delta = \Delta((z \mid acc)_v, \chi_z)$. Suppose $y \leftarrow (y \mid rej)_v$, and the algorithm A distinguishes $(A, [A \mid I]\, y)$ from a uniform distribution. Then there exist algorithms B_1 and B_2 with comparable running times such that

$$Adv^{MLWE}_{(y|rej)_v}(A) \leq \frac{1}{p_{rej}}(Adv^{MLWE}_{\chi_r}(B_1) + Adv^{MLWE}_{\chi_z}(B_2) + \delta).$$

The key point of the proof is that the source distribution can be written as a mixture $\chi_r = p_{rej}(y \mid rej)_v + (1 - p_{rej})((z \mid acc)_v - v)$, and the commitment corresponding to the latter component differs from $[A \mid I]z$ only by a public shift $[A \mid I]v$. By applying the MLWE assumption once to each component and then using the triangle inequality to add the advantages of the two parts, the result follows.

3.7 Two-Party Threshold Signatures and Their Security

Following (del Pino & Niot 2025; Celi et al. 2026), a two-party threshold signature scheme consists of the following components:

(1) $Setup(1^\kappa)$ outputs public parameters;

(2) $KeyGen$ is executed by a trusted dealer or through a two-party protocol and outputs the public key pk and private key shares sk_0 and sk_1;

(3) The signing protocol comprises several rounds of $Sign_1, ..., Sign_\Lambda$. In round λ, party P_i takes the message received in the preceding round and its own state as input and generates the message for the current round;

(4) $Combine$ takes all protocol messages as input and outputs a signature or $\perp$;

(5) $Verify$ is the Aigis-sig signature verification algorithm.

Definition 4 (Correctness with Aborts; Celi et al. 2026). If, when both parties act honestly, a signing attempt outputs a signature with probability at least p, and the output signature is rejected by $Verify$ with negligible probability, then the scheme satisfies correctness with aborts with termination probability p.

Definition 5 (Two-Party Threshold Unforgeability). The security game $TS-UF$ is played between a challenger and an adversary A.

(1) A selects the corrupted party $P_{\bar{h}}$, and the other party P_h is honest. The challenger performs key generation: with a trusted dealer, $sk_{\bar{h}}$ is given to A; with distributed key generation, A participates in the protocol in place of $P_{\bar{h}}$.

(2) A may make arbitrary concurrent calls to the honest party's signing oracle $O_\lambda(sid,)$. Each call executes round λ using the input specified by A and returns the output of P_h; the network is entirely controlled by A. Whenever the honest party records a message m and a context string ctx to be signed in a session, the challenger adds (m, ctx) to the set Q.

(3)A outputs (m^*, ctx^*, σ^*). If $(m^*, ctx^*) \notin Q$ and $Verify(pk, m^*, \sigma^*, ctx^*) = 1$, then A wins, and its winning probability is denoted as $Adv^{TS-UF}(A)$.

This definition uses static corruption, consistent with (del Pino et al. 2024; Celi et al. 2026). The

security analysis in Section 5 uses this model. Definition 7 in Section 5.5 specifies when messages are added to Q in the preprocessing mode.

4. The ThAigis-2P Scheme

4.1 Public Parameters

In addition to the parameters of Aigis-sig, the public parameters of ThAigis-2P include the scaling factor ν, the target radius r, the sampling radius r', the spectral threshold S_{thr}, and the number of candidates K, whose values are specified in Section 6. Let $D = n(k+l)$, $d_z = nl$, and $B = \lceil \sqrt{\tau S_{thr}} \rceil$. In addition to the *ExpandA*, *CRH*, and H_c provided by Aigis-sig, the protocol uses six additional 256-bit hash functions: the commitment digest H_{cmt}, salt commitment H_{salt}, record digest H_{rec}, matrix seed H_ρ, public key component commitment H_{key}, and share derivation H_{share}. These are implemented using distinct domain separation labels and are treated as mutually independent random oracles in the security analysis. The counterparty for P_i is denoted as P_{1-i}. In this paper, a "round" refers to each party sending one message, and a "round trip" refers to one party sending a request and the other party replying.

4.2 Share Generation and Spectral Check

Definition 6 (Spectral Statistic). For $u = (s_1, s_2) \in R^l \times R^k$ and $j \in [n]$,

Let $S_j(u) = \nu^{-2} \sum_{p=1}^{l} |s_{1,p}(\zeta_j)|^2 + \sum_{p=1}^{k} |s_{2,p}(\zeta_j)|^2, Spec_\nu(u) = \max_{j\in[n]} S_j(u)$. (3)

where $s_{1,p}$ and $s_{2,p}$ are the components with index p of s_1 and s_2, respectively. By conjugate symmetry, it suffices to evaluate the statistic for $j < n/2$.

Lemma 6 (Spectral Upper Bound)

For any $u \in R^l \times R^k$ and any $c \in C_\tau$, $\| L_\nu^{-1}(cu) \|_2^2 \leq \tau \cdot Spec_\nu(u)$.

Proof: By the Parseval identity and $(cs)(\zeta_j) = c(\zeta_j)s(\zeta_j)$,

$$\| L_\nu^{-1}(cu) \|_2^2 = \nu^{-2} \sum_p \| cs_{1,p} \|_2^2 + \sum_p \| cs_{2,p} \|_2^2 = \frac{1}{n} \sum_{j\in[n]} |c(\zeta_j)|^2 S_j(u) \leq Spec_\nu(u) \cdot \frac{1}{n} \sum_{j\in[n]} |c(\zeta_j)|^2.$$

Applying the Parseval identity once more, we have $\frac{1}{n}\sum_j |c(\zeta_j)|^2 = \| c \|_2^2 = \tau$. Q.E.D.

The bound in Lemma 6 holds for every challenge, regardless of how it is selected. For the single share of parameter set I studied in Section 6.3, a local search found a challenge for which $\| L_\nu^{-1}(cu) \|_2$ reached 0.93 times $\sqrt{\tau Spec_\nu(u)}$. HAETAE (Cheon et al. 2024) uses a related key rejection condition in single-party signatures. Here the spectral screen supports local rejection, and its effect on the share distribution is accounted for in the security reduction.

Algorithm 2 *ShareGen*(*seed*)

1: $ctr := 0$

2: **repeat**

3: Derive a seed from $H_{share}(seed, ctr)$ and sample $s_1 \leftarrow U(S_{\eta_1}^l)$ and $s_2 \leftarrow U(S_{\eta_2}^k)$ according to the Aigis-sig key generation method; $ctr := ctr + 1$

4: **until** $Spec_\nu(s_1, s_2) \leq S_{thr}$

5: **return** $u := (s_1, s_2)$

Let E be the event $Spec_\nu(u) \le S_{thr}$, let $p_E = \Pr_{u \leftarrow \chi_s}[E]$, and let χ_s^E denote χ_s conditioned on E, i.e., the output distribution of ShareGen. We take S_{thr} to be the median of $Spec_\nu$ under χ_s, so $p_E \approx 1/2$ and ShareGen requires about two sampling attempts on average. By Lemma 6, every output of ShareGen satisfies the following: for any $c \in C_\tau$, $\| L_\nu^{-1}(cu) \|_2 \le B$.

Lemma 7 (Conditional Cost) $R_\infty(\chi_s^E \parallel \chi_s) = 1/p_E$.

Proof: For $u \in E$, $\chi_s^E(u)/\chi_s(u) = 1/p_E$; for$u \notin E$, the ratio is 0. Q.E.D.

4.3 Key Generation

Key generation with a trusted dealer is described in Algorithm 3. The dealer deletes all secrets after Step 4; neither party holds t_0, and it is not required for signing.

Algorithm 3: *KeyGen* (with a Trusted Dealer)

1: $\rho \leftarrow \{0,1\}^{256}$, $A := ExpandA(\rho)$

2: **for** $i \in \{0,1\}$: $seed_i \leftarrow \{0,1\}^{256}$, $u_i = (s_{1,i}, s_{2,i}) := \mathsf{ShareGen}(seed_i)$, $t^{(i)} := [A \mid I]u_i$

3: $t := t^{(0)} + t^{(1)}$, $(t_1, t_0) := Power2Round_q(t, d)$, $pk := (\rho, t_1)$, $tr := CRH(pk)$

4: $sk_i := (tr, u_i)$; **return** (pk, sk_0, sk_1)

The two-party distributed key generation protocol (Algorithm 4) has two phases and four rounds in total. First, the parties generate a matrix seed by committing to their seeds and then revealing them. They then exchange public key components in the same way. In the first phase, neither party can choose its own seed after seeing the other party's seed; at most, it can force a restart by aborting. In the second phase, exchanging commitments first prevents a party from choosing $t^{(i)} = t^\star - t^{(1-i)}$ after seeing $t^{(1-i)}$ and thereby setting the joint public key to $t^\star$, for which it knows the private key. If the corrupt party submits $t^{(\bar{h})}$ that does not correspond to any short share, the honest party cannot sign using it; this only constitutes a denial-of-service attack and does not affect unforgeability.

Algorithm 4: Two-Party Distributed Key Generation (from the perspective of P_i)

1: Round 1:$\rho_i \leftarrow \{0,1\}^{256}$, send $C_i := H_\rho(com, i, \rho_i)$

2: Round 2: After receiving C_{1-i}, send ρ_i

3: Round 3: Verify $C_{1-i} = H_\rho(com, 1-i, \rho_{1-i})$; $\rho := H_\rho(seed, \rho_0, \rho_1), A := ExpandA(\rho)$; $seed_i \leftarrow \{0,1\}^{256}$, $u_i := ShareGen(seed_i)$, $t^{(i)} := [A \mid I]u_i$; send $C_i' := H_{key}(\rho, i, t^{(i)})$

4: Round 4: After receiving C_{1-i}', send $t^{(i)}$

5: Output: Verify $C_{1-i}' = H_{key}(\rho, 1-i, t^{(1-i)})$; abort if any verification fails; $t := t^{(0)} + t^{(1)}$, $(t_1,) := Power2Round_q(t, d)$, $pk := (\rho, t_1)$, $sk_i := (CRH(pk), u_i)$

After distributed key generation is complete, both parties know the full t. Security analyses of both Aigis-sig and Dilithium treat the full t as a public value (Ducas et al. 2018; Zhang et al. 2020); only the high bits are included in the public key to reduce its length, so the fact that both parties know$t$ does not affect security.

4.4 Hyperball Sampling and Local Rejection

Each candidate mask is generated by Algorithm 5: $D + 2$ independent standard normal variables are normalized, and the first D coordinates are selected to obtain a uniformly distributed point in the unit ball (Celi et al. 2026), which is then scaled to a radius of r'.

Algorithm 5 $HSample(A)$

1: Sample $g_1, \ldots, g_{D+2} \leftarrow N(0,1)$ independently, and set

$$\xi := r' \cdot (g_1, \ldots, g_D) / \| (g_1, \ldots, g_{D+2}) \|_2$$

2: $y = (y^{(1)}, y^{(2)}) := \lfloor L_\nu \xi \rceil \in Z^{nl} \times Z^{nk}$, $w := Ay^{(1)} + y^{(2)} \bmod q$

3: return (ξ, y, w)

Algorithm 6 $LocalRej(\xi, y, u_i, c)$

1: Calculate the integer vectors $cs_{1,i}$ and $cs_{2,i}$, and let $v := L_\nu^{-1}(cu_i) = (cs_{1,i}/\nu,\ cs_{2,i})$

2: if $\| \xi + v \|_2 \leq r$ then return $z^{(1)} := y^{(1)} + cs_{1,i}$

3: else return $\perp$

The rejection condition depends only on the continuous sample ξ. Since $L_\nu v = cu_i$ is an integer vector and rounding commutes with integer translation, upon acceptance we have

$$\lfloor L_\nu(\xi + v) \rceil = \lfloor L_\nu \xi \rceil + cu_i = y + cu_i. \quad (4)$$

ξ follows a continuous distribution, with a probability of 0 of falling on half-integer rounding boundaries. Therefore, if $\xi \leftarrow U(B_D(r'))$, then the output of LocalRej (along with the unsent second component $y^{(2)} + cs_{2,i}$) is the result of running hyperball rejection sampling with the continuous shift v and then applying the deterministic mapping $x \mapsto \lfloor L_\nu x \rceil$; the corresponding integer displacement is cu_i. By Lemma 6 and ShareGen, $\| v \|_2 \leq B$ holds for all challenges; when the parameters satisfy $r'^2 \geq r^2 + B^2 + 2rB/\varphi$, the conditions of Lemma 4 hold. Under the parameters of this paper, $\nu r' < (q-1)/2$ and $r' < (q-1)/2$; both the partial responses and the coefficients of $y^{(2)}$ can be unambiguously represented by centered representatives.

4.5 Interactive Mode

The signing protocol for the interactive mode is described in Algorithm 7. Each signature attempt uses a new session identifier sid. Round 1 is message-independent and can be executed in advance; the honest party records (sid, m, ctx) at the start of Round 2, and this value remains unchanged thereafter. Since the commitment digests were exchanged in Round 1, the corrupt party fixes its commitment before seeing the honest party's commitment; thus, the high-order bits of the aggregated commitment have sufficient entropy, allowing the proof to program H_c at the end of Round 2.

Session state must be used only once. If state from after Round 1 is reused for two messages m and $m' \neq m$, an honest partial response for a candidate accepted in both executions satisfies $z - z' = (c - c')s_{1,h}$. This relation leaks information about the honest share. A unit challenge difference suffices to determine that component in the ring. A general claim of unique recovery requires a separate argument. Persist the round-2 binding and ensure that the round-3 deletion of ξ and y cannot be undone by rollback. The state service must satisfy the requirements in Section 5.4.

Algorithm 7: Signing in Interactive Mode (from the perspective of P_i, with input $sk_i = (tr, u_i)$)

1: Round 1: For $j \in [K]$, $(\xi_{ij}, y_{ij}, w_{ij}) := HSample(A)$; $W_i := (w_{ij})_{j \in [K]}$; send $cmt_i := H_{cmt}(tr, sid, i, W_i)$

2: Round 2 (input: m and optionalctx): After receiving cmt_{1-i}, persist the record (sid, m, ctx), then send W_i

3: Round 3: Receive W_{1-i}; if $cmt_{1-i} \neq H_{cmt}(tr,\ sid, 1-i, W_{1-i})$, abort

4: Calculate μ according to Equation (1); for $j \in [K]$: $w_j := w_{0j}+w_{1j}, c_j := H_c(\mu, HighBits_q(w_j, 2\gamma_2))$

5: For $j \in [K]$: $z_{ij} := LocalRej(\xi_{ij}, y_{ij}, u_i, c_j)$, $\theta_{ij} := [z_{ij} \neq \perp]$

6: Delete all ξ_{ij} and y_{ij} from this session, then send the bitmap $(\theta_{ij})_{j\in[K]}$ and $(Enc(z_{ij}))_{\theta_{ij}=1}$

Either party or a third party may collect all messages and run Algorithm 8. The combiner checks the candidates accepted by both parties one by one and outputs the first signature that passes the check; if all candidates fail, the attempt ends, and both parties restart with a new session. The combiner uses only public data, and its execution by any party does not affect security.

Algorithm 8 $Combine(pk, (w_j, c_j)_j, (\theta_{0j}, \theta_{1j}, z_{0j}, z_{1j})_j)$

1: **for**$j = 0, 1, ..., K-1$ and $\theta_{0j} = \theta_{1j} = 1$ **do**

2: $z := z_{0j} + z_{1j}$ (integer addition); **if**$\| z \|_\infty \geq \gamma_1 - \beta_1$ **then continue**

3: $\widetilde{w} := Az - 2^d c_j t_1 \bmod q$, $\delta := w_j - \widetilde{w} \bmod q$ (select a centered representative); **if**$\| \delta \|_\infty > \gamma_2$, **then continue**

4: $h := MakeHint_q(\widetilde{w}, \delta, 2\gamma_2)$; **if** $wt(h) > \omega$ **then continue**

5: **return** $\sigma := (z, h, c_j)$

6: **return** $\perp$

The combiner may call Aigis-sig's signature verification one more time before outputting the result to rule out implementation errors; in the experiments conducted for this paper, this check never failed.

4.6 Preprocessing Mode

A preprocessing record consists of a record identifier rid, K candidate commitments from each party, and the parties' commitments to their salts; it is hereinafter referred to as a "record." rid is assigned by the state service upon registration, ensuring uniqueness under the same key and appearing in the input of all relevant hashes. Either party can act as the initiator; the initiator P_a runs Combine and outputs a signature, while the other party P_b ($b = 1 - a$) acts as the responder.

Algorithm 9 Preprocessing (Offline, from the perspective of P_i)

1: Register a batch of records with the state service and obtain new record identifiers; proceed only after successful registration

2: For each rid: Run $HSample(A)$ on $j \in [K]$; $salt_i \leftarrow \{0,1\}^{256}$, $com_i := H_{salt}(tr, rid, i, salt_i)$; send (rid, W_i, com_i)

3: Upon receiving $(rid, W_{1-i}, com_{1-i})$, compute $w_j := w_{0j} + w_{1j}$; if there exist $j \neq j'$ such that $HighBits_q(w_j) = HighBits_q(w_{j'})$, set the record to INVALID

4: Otherwise, save $(\xi_{ij})_j$, $(w_j)_j$, $salt_i$, com_{1-i}, and the digest $d_{rec} := H_{rec}(tr, rid, W_0, W_1, com_0, com_1)$, and set the record to READY

Algorithm 10: Online Signing (Initiator: P_a; Responder: P_b)

1: P_a: Atomically change the record rid from READY to SPENT and bind the request digest; abort if the operation fails

2: P_a: Send $req := (rid, d_{rec}, m, salt_a)$

3: P_b: Verify that d_{rec} matches the local record and that $H_{salt}(tr, rid, a, salt_a) = com_a$; atomically

change the record status from READY to SPENT and bind req; abort on failure

4:P_b: $ctx := salt_0 \oplus salt_1$; calculate μ according to Equation (1); for $j \in [K]$: $c_j := H_c(\mu, HighBits_q(w_j))$, $z_{bj} := LocalRej(\xi_{bj}, y_{bj}, u_b, c_j)$

5: P_b: Delete the private randomness for this record, then send $resp := (salt_b, (\theta_{bj})_j, (Enc(z_{bj}))_{\theta_{bj}=1})$

6: P_a: Check $H_{salt}(tr, rid, b, salt_b) = com_b$; compute ctx, μ, $(c_j)_j$, and its own z_{aj}; run Combine, outputting (σ, ctx); delete the private randomness

Submit the signature σ along with the 32-bytectx to the verifier, who then calls $Verify(pk, m, \sigma, ctx)$. This requires the verifier to accept the context string provided with the signature. If ctx is used as a fixed domain separator in the deployment, the signature can instead be applied to $m \parallel (salt_0 \oplus salt_1)$; in this case, the signed message is a salted byte string, and the authorization set in Section 5.5 also records this byte string accordingly. The application layer parses m after verifying the signature.

In Schnorr-style signatures, if the commitment is made public before the message is finalized, an attacker can indirectly select challenges by choosing messages in concurrent sessions, thereby launching a ROS-type forgery (Drijvers et al. 2019; Benhamouda et al. 2021). The preprocessing mode prevents such attacks through the following three design elements.

(1) Random salt. The responder commits to its salt offline and reveals it only after marking the record as SPENT. The corrupt initiator therefore does not know ctx before submitting its request, and the honest responder produces at most one response for that record. The corrupt responder's commitment fixes its salt. The participant identifier i separates the two commitments, preventing a corrupt party from copying the honest party's commitment and opening it with the honest salt to force $ctx = 0$.

(2) Checking that the high bits differ. Since there is only one round in the offline phase, a corrupt party can choose its own commitment after seeing the honest party's commitment, ensuring that the two candidate aggregate commitments for the same record have the same high-order bits; these two candidates will then receive the same challenge. No attacks exploiting this case have been discovered to date; however, it would undermine the simulation method for pre-sampled challenges described in Section 5.5. Therefore, Step 3 of Algorithm 9 invalidates such records. This check uses only publicly available information and can be completed during the offline phase.

(3) One-time use by the initiator. The signature output by the initiator contains $z_{aj} + z_{bj}$, from which the responder can compute the initiator's partial response; therefore, the initiator can also use each record only once.

4.7 Canonical Encoding of Partial Responses

For each coefficient of the partial response $z_{ij} \in Z^{d_z}$, we have $|x| \le (q-1)/2$. Let $b_q = \lceil \log_2 q \rceil$ (21 for parameter set I and 22 for parameter sets II and III). Define the bijection by $ZZ(x) = 2x$ for $x \ge 0$ and $ZZ(x) = -2x - 1$ for $x < 0$; it maps $[-(q-1)/2,\ (q-1)/2]$ to $[0, q-1]$. The first byte of the encoding specifies the mode, and all bit strings are written with the most significant bit first.

N mode (mode byte $0x00$): The second byte specifies the bit width $b_N \in [1, b_q]$. Each coefficient is then stored using b_N bits as $x + 2^{b_N - 1}$. All coefficients must belong to $[-2^{b_N-1}, 2^{b_N-1})$, and the encoder selects the smallest b_N satisfying this condition.

$ZZ(x)$ is the value encoded in R mode (mode byte $0x01$), using b_q bits per coefficient.

C mode (mode byte $0x02$): Each block contains 64 coefficients, and its first byte specifies the parameter a. If $a < b_q$, let $u = ZZ(x)$ and set $t = \lfloor u/2^a \rfloor$: when $t < 15$, write t ones, one zero, and the lower a bits of u; otherwise, write 15 ones followed directly by the b_q bits representing u. If $a = b_q$, write the entire block without compression. Pad the end of each block with zeros to the next byte boundary.

For each block, the encoder selects the parameter that minimizes the encoded length in bytes, choosing the smaller value of a in a tie. It then selects the mode with the shortest total length, breaking ties in the order N, R, C. The decoder checks that the mode byte, bit width, and block parameters are in their valid ranges; $u < q$; $\lfloor u/2^a \rfloor \geq 15$ in the escape branch; all padding bits are zero; and the coefficient count and total length are correct. Finally, it re-encodes the decoded result and compares it with the input byte by byte. If any check fails, it rejects the input.

Lemma 8 (Lossless and Canonical Encoding)

For any $x \in [-(q-1)/2,\ (q-1)/2]^{d_z}$, $Dec\ (Enc(x)) = x$, and $|Enc(x)| \leq 1 + \lceil d_z b_q/8 \rceil$. For any byte string e, if $Dec(e) = x \neq \perp$, then $e = Enc(x)$.

Proof: ZZ is a bijection. N mode and R mode use fixed-length encodings that are invertible over their respective domains. In C mode, the ordinary branch is identified by the first zero, which determines the quotient t; the escape branch is identified by 15 leading ones, followed by the original value encoded in b_q bits. Each block contains 64 coefficients and is padded to a byte boundary, so block boundaries are uniquely determined. R mode applies to all inputs, and the encoder selects the shortest encoding, so its length does not exceed the R mode length, $1 + \lceil d_z b_q/8 \rceil$. The canonical encoding property follows from the re-encoding comparison at the end of decoding. Q.E.D.

In Round 3 of the interactive mode or in an online response in the preprocessing mode, after the header fields listed in Appendix A, the sender transmits a bitmap of $\lceil K/8 \rceil$ bytes. Unused bits must be zero. For each accepted candidate, in ascending order of candidate index, the sender writes a 2-byte big-endian length followed by the encoded vector. The receiver determines the expected number of vectors from the bitmap and rejects missing, duplicate, or extra vectors. Integrity relies on message authentication between the parties.

5 Correctness and Security

5.1 Statistical Properties of Local Rejection

Lemma 9 (Local Rejection) Let u be an output of ShareGen, let $c \in C_\tau$, and suppose the parameters satisfy $r'^2 \geq r^2 + B^2 + 2rB/\varphi$ and $2\varepsilon = I_{1-1/\varphi^2}((D+1)/2, 1/2)$. Let $Real_{u,c}$ denote the distribution of the complete response $\lfloor L_\nu(\xi + v) \rceil$ (or $\perp$ in case of rejection) for the LocalRej output when $\xi \leftarrow U(B_D(r'))$. Then (1) $R_\infty(Real_{u,c} \parallel Ideal(\chi_z, M)) \leq 1 + \varepsilon/(M-1)$; (2) the acceptance probability lies between $(1-\varepsilon)/M$ and $1/M$; (3) the statistical distance between the distribution of the output upon acceptance and χ_z does not exceed ε.

Proof: By Lemma 6, $\| v \|_2 \leq B$. Let $x = \xi + v$; then $x \leftarrow U(B_D(r', v))$, and LocalRej accepts when $x \in B_D(r)$. For uniform distributions over the two balls, the density ratio $\chi_z(x)/(M\chi_r(x-v))$ equals 1 at $B_D(r) \cap B_D(r', v)$ and 0 (or has a zero denominator) elsewhere; thus, LocalRej is the continuous version of Rej. Lemma 4 gives $R^\varepsilon_\infty(U(B_D(r)) \parallel U(B_D(r', v))) \leq M$, and Lemma 3 yields (1), (2), and (3). The mapping $x \mapsto \lfloor L_\nu x \rceil$ is deterministic; by the data processing inequality in Lemma 2 and Equation (4), all three conclusions also hold for the rounded output. Q.E.D.

Lemma 9 holds for every c in the challenge set, so the proof need not account for bad events in which the shift bound is exceeded. An alternative is to establish a rigorous tail bound for $\| L_\nu^{-1}(cu) \|_2$ only for random challenges: in the proof game, the simulator samples challenges in honest sessions uniformly and independently of the honest shares, then applies a union bound over Q candidate challenges. This approach may yield a smaller B (Section 6.3), but it requires a separate derivation of a tail bound for quadratic forms, which is not pursued here.

5.2 Correctness and Probability of Termination

Theorem 1 (Output Correctness) If Combine outputs $\sigma = (z, h, c_j)$, then $Verify(pk, m, \sigma, ctx) = 1$.

Proof: Combine has verified that $\| z \|_\infty < \gamma_1 - \beta_1$ and $wt(h) \leq \omega$; the signature can be canonically encoded, and both the format check and norm check for signature verification will pass. The verifier computes $Az - 2^d c_j t_1$ using the same pk, z, and c_j; this value equals $\widetilde{w}$ in Combine. Since $\| \delta \|_\infty \leq \gamma_2 = \alpha/2$, Lemma 1 yields $UseHint_q(h, \widetilde{w}) = HighBits_q(\widetilde{w} + \delta) = HighBits_q(w_j)$. The verifier computes the same μ according to Equation (1) using the same tr, m, and ctx, so $H_c(\mu, HighBits_q(w_j)) = c_j$. Q.E.D.

Theorem 1 does not require either party to be honest: the signature output by Combine is necessarily a valid Aigis-sig signature. The following identity explains the origin of δ when the algorithm is executed honestly.

Lemma 10 (Residue Identity). Let both parties be honest and suppose candidate j is accepted by both parties. Let $(z_{ij}^{(1)}, z_{ij}^{(2)}) = y_{ij} + c_j u_i$, and define the aggregate second component by $z^{(2)} = z_{0j}^{(2)} + z_{1j}^{(2)}$, where the local response of P_i includes its unsent second component. Then, in Combine, $\delta = z^{(2)} - c_j t_0 \bmod q$.

Proof: By $t = As_1 + s_2 = 2^d t_1 + t_0$, $\widetilde{w} = A(y_{0j}^{(1)} + y_{1j}^{(1)}) + c_j As_1 - c_j(t - t_0) = A(y_{0j}^{(1)} + y_{1j}^{(1)}) - c_j s_2 + c_j t_0$.

Furthermore, $w_j = A(y_{0j}^{(1)} + y_{1j}^{(1)}) + y_{0j}^{(2)} + y_{1j}^{(2)}$; subtracting the two yields the result. Q.E.D.

Theorem 2 (Termination Probability; Celi et al. 2026). Let $\varepsilon = o(1)$, and suppose that, for two independent pairs (y_0, y_1) and (y_0', y_1'), with each vector sampled from χ_r, the probability that $[A \mid I](y_0 + y_1)$ and $[A \mid I](y_0' + y_1')$ have the same high bits is $o(1)$. When both parties act honestly, the probability that a single signing attempt produces a signature is

$$p = E_{k \leftarrow Bin(K, M^{-2})}[v_k] + o(1). \quad (5)$$

where v_k is the probability that, when the two complete responses for each of k candidates are independently sampled from χ_z, at least one candidate passes all checks in Combine.

Proof (sketch): By Lemma 9, when both parties' local rejection sampling is replaced with Ideal, the joint distribution of all $2K$ outputs has a Rényi divergence of at most $(1 + \varepsilon/(M-1))^{2K} = 1 + o(1)$; therefore, the difference between the signing success probabilities in the two experiments is $o(1)$. In the ideal experiment, each party independently accepts each candidate with probability $1/M$; the number of candidates accepted by both parties has distribution $Bin(K, M^{-2})$, and the accepted responses have distribution χ_z. The probability of passing the checks in Combine then follows from Lemma 10; the details are the same as in Theorem 3.1 of Celi et al. (2026). Q.E.D.

In the parameter analysis, we adopt the approximation from Reference (Celi et al. 2026): we treat the checks in Combine for different candidates as independent events. Let the probability that a single candidate passes the checks in Combine be p_{glob}, and the probability of acceptance by one

party be $p_{acc} \approx 1/M$. Then

$p \approx 1 - (1 - p_{acc}^2\, p_{glob})^K$. (6)

The checks in Combine for different candidates share the same public-key low bits t_0, so the checks need not be independent. The measurements in Section 7.3 are consistent with the approximation in Equation (6).

5.3 Unforgeability of the Interactive Mode

Theorem 3. Let an adversary A mount a $TS-UF$ attack against the interactive ThAigis-2P protocol, establish at most Q_s signing sessions with the honest party, and make at most Q_{cmt}, Q_{crh}, and Q_c queries to H_{cmt}, CRH, and H_c, respectively. Let $Q = KQ_s$ be the total number of candidates generated by the honest party and let H_w be a lower bound on the min-entropy of a single honest commitment (see G_1 in the proof). Then there exist algorithms B_s, B_r, B_z, and B_{UF} with running times comparable to that of A such that

$$Adv^{TS-UF}(A) \leq$$

$$\epsilon_{RO} + Q(2Adv^{MLWE}_{\chi_r}(B_r) + Adv^{MLWE}_{\chi_z}(B_z) + \varepsilon) + (1 + \tfrac{\varepsilon}{M-1})^Q \tfrac{1}{p_E}(2Adv^{MLWE}_{\chi_s}(B_s) + Adv^{UF}_{Aigis}(B_{UF})), \quad (7)$$

where

$$\epsilon_{RO} = \frac{(Q_{cmt} + 2Q_s)^2 + Q_s Q_{cmt}}{2^{256}} + Q_s Q_{cmt} 2^{-H_w} + \frac{Q_{crh}^2}{2^{384}} + Q(Q_c + Q)\left(\frac{2\gamma_2 + 1}{q}\right)^{nk}.$$

When two-party distributed key generation is used, ϵ_{RO} increases by $(Q_\rho + Q_{key} + 2) \cdot 2^{-255}$, where Q_ρ and Q_{key} are the numbers of queries to H_ρ and H_{key}, respectively.

Proof. Let the honest party be P_h and the corrupt party be $P_{\bar{h}}$. The honest party's public key component is $t^{(h)} = [A \mid I]u_h$. Let $\Pr[G_\iota]$ be the probability that A wins game G_ι. For distributed key generation, the external-matrix hybrids require the ability to embed a given matrix as the output of $ExpandA(\rho)$ at the derived seed. Programming H_ρ to select that seed alone does not establish this embedding for a fixed deterministic matrix expander. The key embedding in G_{11} uses the seed supplied with the Aigis-sig public key.

G_0: The real $TS - UF$ game, with the honest party's share sampled as $u_h \leftarrow \chi_s^E$.

G_1 (Delayed Commitment Generation): In Round 1, the honest party outputs a uniformly random string cmt_h; W_h is generated only in Round 2, at which point $H_{cmt}(tr, sid, h, W_h)$ is programmed to cmt_h. The two games differ only if A has queried this input before Round 2. Given A and $y^{(1)}$, since $\| y^{(2)} \|_\infty \leq r' < (q-1)/2$, the commitment w uniquely determines $y^{(2)}$; furthermore, since the probability of any point in χ_r does not exceed $\nu^{-nl}/Vol(B_D(r'))$, the min-entropy of a single honest commitment is at least $H_w := \log_2 Vol(B_D(r')) + nl \log_2 \nu - nl \log_2(2\nu r' + 1)$. For the three sets of parameters, H_w is greater than or equal to 13,030, 16,816, and 20,000 bits, respectively. Therefore, $|Pr[G_0] - \Pr[G_1]| \leq Q_s Q_{cmt} 2^{-H_w}$.

G_2 (Commitment collisions and openings). The game aborts on a collision in H_{cmt}. It also excludes a preimage query by A for the honest party's commitment digest after the digest is sent and before the honest commitment is revealed, as well as a valid opening by A of its own digest when the corresponding oracle query has not been made before the honest reveal. The stated bound for these

events is $((Q_{cmt}+2Q_s)^2+Q_sQ_{cmt})/2^{256}$. In G_2, a corrupt commitment $W_{\bar{h}}$ that passes verification in Round 3 must thus be fixed before the honest party reveals W_h.

G_3 (Excluding Message Representative Collisions): If CRH gives the same μ for different inputs, the game aborts, at a cost of at most $Q_{crh}^2/2^{384}$. In G_3 and subsequent games, the message representative of $(m, ctx) \notin Q$ differs from all previously recorded message representatives.

G_4 (Programming the Challenge): In Round 2, when the honest party reveals W_h, sample $c_j \leftarrow U(C_\tau)$ for each $j \in [K]$, and program $H_c(\mu, HighBits_q(w_j))$ to c_j; if that point has previously been queried or programmed, the game aborts. The challenge is never disclosed to the attacker prior to programming, and uniform sampling does not alter the distribution of H_c. By G_2, the corrupted party's portion in $w_j = w_{hj} + w_{\bar{h}j}$ is fixed before w_{hj} is revealed; after replacing w_{hj} with a uniform value (the MLWE assumption under χ_r), w_j is uniformly distributed over R_q^k, and the probability that its high bits take any given value does not exceed $((2\gamma_2+1)/q)^{nk}$. Therefore, $|\Pr[G_3] - \Pr[G_4]| \le Q(Q_c+Q)\ ((2\gamma_2+1)/q)^{nk} + QAdv_{\chi_r}^{MLWE}$.

G_5 (Generate responses first): Compute the honest party's local rejection results concurrently with programming the challenge in Round 2 and return them directly in Round 3. By G_2 and G_4, the challenge used in Round 3 is precisely the one programmed in Round 2, so $\Pr[G_5] = \Pr[G_4]$.

G_6 (Deriving Commitments from Responses): For each candidate, first run hyperball rejection sampling to obtain a complete response z_{hj} or a rejection; upon acceptance, set $w_{hj} := [A \mid I]z_{hj} - c_j t^{(h)}$; upon rejection, sample y from $(y \mid rej)_{c_j u_h}$ and set $w_{hj} := [A \mid I]y$. From Equation (4), $[A \mid I]\ (y + c_j u_h) = w_{hj} + c_j t^{(h)}$. This is another sampling order for the same joint distribution, so $Pr[G_6] = \Pr[G_5]$.

G_7 (Replacing the commitment of a rejected candidate): Replace each rejected candidate's w_{hj} with a uniform value on R_q^k. Replace candidates one by one in the order they appear, conditioning on the preceding execution history and the value of (u_h, c_j). Replacement $\imath$ changes the game only if that candidate is rejected; the reduction algorithm uses fresh randomness to determine whether to reject, and if rejected, uses the external challenge sample as the commitment; it knows u_h and can simulate the rest. Thus, the gap for this step does not exceed p_{rej} multiplied by the distinguishing advantage in Lemma 5, and p_{rej} cancels out with the factor $1/p_{rej}$ from Lemma 5; furthermore, by Lemma 9, the statistical distance between the acceptance distribution and χ_z does not exceed ε. Therefore, the gap for each replacement does not exceed $Adv_{\chi_r}^{MLWE} + Adv_{\chi_z}^{MLWE} + \varepsilon$, and Q replacements result in a total of $Q(Adv_{\chi_r}^{MLWE} + Adv_{\chi_z}^{MLWE} + \varepsilon)$.

G_8 (Ideal Local Rejection): Replace the honest party's rejection sampling with $Ideal(\chi_z, M)$: with probability $1/M$, sample $z_{hj} \leftarrow \chi_z$ and set $w_{hj} := [A \mid I]z_{hj} - c_j t^{(h)}$; otherwise, output $\perp$ and set w_{hj} to be uniformly distributed. Consider the sequence of random variables formed by the outputs of all honest candidates given any execution history. The next candidate's challenge is determined by the history, and since Lemma 9 holds for every challenge, the conditional distribution has a Rényi divergence from the ideal distribution of no more than $1 + \varepsilon/(M-1)$. By the multiplicativity of Lemma 2, the data processing inequality, and probability preservation, $Pr[G_7] \le (1 + \varepsilon/(M-1))^Q \Pr[G_8]$.

G_9 (Returning to the native distribution): In G_8, the honest party's output is independent of the secret; the game depends on u_h only through $t^{(h)}$ (used for the public key and the commitments of accepted candidates) and is defined for any u_h in the support of χ_s. The only difference between G_9

and G_8 is $u_h \leftarrow \chi_s$. Treating "A wins" as an event determined by u_h and the rest of the game's random variables, by Lemma 7 and the probability preservation property of Lemma 2, $\Pr[G_8] \leq \Pr[G_9]/p_E$.

G_{10} (Randomizing the Honest Public Key Component): Replace $t^{(h)}$ with a uniform value in R_q^k. B_s embeds the AMLWE challenge (A, b) as $(A, t^{(h)})$ and simulates the rest of the game, so $|\Pr[G_9] - \Pr[G_{10}]| \leq Adv_{\chi_s}^{MLWE}$. In this case, $t = t^{(h)} + t^{(\bar{h})}$ is uniformly distributed.

G_{11} (Embedding the Aigis-sig public key): The public key $(\rho, t_1^\star)$ is generated by Aigis-sig key generation. We uniformly sample t from the elements where the high-order bits are equal to $t_1^\star$, and then set $t^{(h)} := t - t^{(\bar{h})}$. When t is uniformly distributed, its low-order bits, conditioned on its high-order bits, are uniformly distributed over the corresponding set; thus, G_{10} can equivalently be written as "first take the high-order bits of a uniformly selected element, then resample the low-order bits"; G_{11} simply replaces the uniformly selected elements with AMLWE samples, $|\Pr[G_{10}] - \Pr[G_{11}]| \leq Adv_{\chi_s}^{MLWE}$. During distributed key generation, the simulator extracts $\rho_{\bar{h}}$ and $t^{(\bar{h})}$ from queries to H_ρ and H_{key} made by A, then programs $H_\rho(seed, \rho_0, \rho_1)$ as ρ and the honest party's public key component commitment as the corresponding $t^{(h)}$; the probability of extraction or programming failure is accounted for in $(Q_\rho + Q_{key} + 2)2^{-255}$.

Finally, construct B_{UF}. It obtains $pk = (\rho, t_1^\star)$ from the Aigis-sig key-only attack challenger and simulates the view of A as in G_{11}: queries to $ExpandA$, CRH, and H_c at unprogrammed points are forwarded to the challenger's random oracle, while programmed points are maintained locally. In G_{11}, the honest party uses no secrets, and the simulation is perfect. The forgery output by A satisfies $(m^*, ctx^*) \notin Q$. By G_3, the message representative μ^* differs from those in all honest sessions; during verification, the query $H_c(\mu^*, \cdot)$ is at an unprogrammed point and is answered by the challenger's random oracle. Therefore, the forgery is also valid under pk, and $Pr[G_{11}] \leq Adv_{Aigis}^{UF}$.

Summing the above terms gives Equation (7). Q.E.D.

Discussion of Theorem 3. The random-oracle error ϵ_{RO} depends on the stated query counts. In particular, H_w exceeds 10,000 bits and $((2\gamma_2 + 1)/q)^{nk}$ is below 2^{-2600}. With $\varepsilon \leq 2^{-128}$ and $Q \leq 2^{64}$, the additive term $Q\varepsilon$ is at most 2^{-64}. The multiplier $(1 + \varepsilon/(M-1))^Q$ differs from 1 by less than 2^{-63} for the selected values of M in Table 2. The MLWE advantages under χ_r and χ_z enter the bound with coefficients $2Q$ and Q, respectively, and are additional computational assumptions. Section 6.4 reports model-based attack costs for these distributions. The factor $1/p_E \approx 2$ accounts for conditioning on the spectral check. A concrete end-to-end security estimate requires a joint bound on all the terms in Equation (7) under an explicit resource budget.

5.4 One-Time Use of Sessions and Records

Lemma 11 (One-Time Use) Let the state service satisfy the following: (1) The compare-and-swap operations for READY → SPENT are linearizable; (2) Committed states are not rolled back to earlier values; (3) A batch is used by at most one signing process; if that process fails, all unused records in the batch are set to INVALID; (4) Record identifiers remain unchanged after backup and restoration. Then, an honest party computes a response to a challenge at most once for each record. The conclusion holds equally for sessions when the session state in the interactive mode satisfies the same requirements.

Proof: An honest party computes the challenge and response only after the READY → SPENT

transition succeeds. By (1), at most one concurrent transition for the same record can succeed; by (2) and the definition of a terminal state, SPENT and INVALID cannot return to READY; by (3), the old process and the new process cannot simultaneously hold the same batch; after failure, the entire batch is invalidated; by (4), a backup or clone cannot regain usage rights with a new identifier. The conclusion follows by induction on the execution history of a batch of records. The situation is the same for sessions; the persistence of records in the second round is equivalent to a state transition. Q.E.D.

5.5 Unforgeability in the Preprocessing Mode

Definition 7 (Message Authorization in the Preprocessing Mode) When the honest party acts as the initiator, it authorizes the message m upon successful state transition in Step 1 of Algorithm 10; when acting as the responder, it authorizes the m in the request upon successful state transition in Step 3. Upon authorization, the challenger adds $(m, salt_0 \oplus salt_1)$ to the set Q, where the corrupt party's salt is uniquely determined by its query to H_{salt}; if the corrupt party's salt commitment does not yet have a known preimage at this point, only m is recorded, to be completed when the preimage becomes known. Thereafter, even if the response is lost, the signature fails, or the other party aborts, the message is still considered authorized. The winning conditions are the same as in Definition 5.

Theorem 4. Under the state assumptions of Lemma 11, suppose an adversary causes the honest party to generate at most Q_{rec} records, including unused records, and let $Q = KQ_{rec}$. Suppose the numbers of queries to H_{salt}, CRH, and H_c are at most Q_{salt}, Q_{crh}, and Q_c, respectively. Then

$$Adv^{TS-UF}(A) \le \epsilon'_{RO} + Q(2Adv^{MLWE}_{\chi_r} + Adv^{MLWE}_{\chi_z} + \varepsilon) + (1 + \tfrac{\varepsilon}{M-1})^Q \tfrac{1}{p_E}(2Adv^{MLWE}_{\chi_s} + Adv^{UF}_{Aigis}), \quad (8)$$

where $\epsilon'_{RO} = (Q^2_{salt} + Q^2_{rec} + Q_{rec}(Q_{salt} + Q_{crh} + Q_c))2^{-256} + Q^2_{crh}/2^{384}$.

Proof: The proof follows the game sequence from Theorem 3, where the arguments for G_1, G_2, and G_4 are replaced by the following arguments based on random salts; all other steps remain unchanged.

(1) Extract the corrupt party's salt. During the offline phase, after receiving the corrupt party's salt commitment, the simulator searches for its preimage in the query table for H_{salt}. If the attacker later opens a commitment for which no preimage or more than one preimage had previously appeared in the query table, the game aborts; the probability of such events is accounted for in $(Q^2_{salt} + Q_{rec}Q_{salt})2^{-256}$. The input for salt commitments includes the participant's ID; the corrupt party cannot reuse the honest party's commitments.

(2) Timing and Freshness of Challenge Programming. For each honest record, the simulator pre-samples $c_j \leftarrow U(C_\tau)$ for each candidate during the offline phase and derives the honest commitment from the response according to the method described in G_6. At authorization, before the honest initiator sends $salt_a$ and before the honest responder sends $salt_b$, the simulator calculates ctx and μ using the extracted salt, and programs $H_c(\mu, HighBits_q(w_j))$ asc_j. If the corrupt party's salt cannot yet be extracted, by (1), this record cannot receive a valid response, and the simulator need not program it. Prior to authorization, the attacker only sees the commitment for the honest party's salt; ctx is uniform from its perspective; the probability that it has previously queried the corresponding CRH input or H_c programming point does not exceed $Q_{rec}(Q_{crh} + Q_c)2^{-256}$. The probability that ctx is the same across different records does not exceed $Q^2_{rec}2^{-256}$; outside these collision events, the programming points for different records are distinct.

(3) The programming points within the same record are all distinct. An attacker, after observing the honest commitment, can choose their own commitment such that the high-order bits of the two candidate aggregate commitments are identical; in this case, the genuine protocol uses the same challenge for both candidates, whereas the two challenges pre-sampled by the simulator may differ. Step 3 of Algorithm 9 invalidates such a record; honest parties never use it, so the simulator need not program it either; this check relies solely on public data and can likewise be performed by the simulator.

(4) Used versus unused records. By Lemma 11, each pre-sampled challenge is programmed at most once, and the response is computed at most once. Candidates that have been used and rejected are handled as G_7; honest commitments in unused records are unconditional χ_r samples, which are directly replaced with uniform values using the MLWE assumption under χ_r, with each candidate incurring a loss of $Adv^{MLWE}_{\chi_r}$. The sum of these two parts does not exceed the second term of Equation (8).

(5) Message freshness. By G_3 and (2), when $(m^*, ctx^*) \notin Q$, the message representative μ^* in the forgery differs from that of any authorized request, and the corresponding H_c query has not been programmed. The remaining steps are the same as in Theorem 3. Q.E.D.

The two modes can share the same set of shares. In this case, the honest candidates from both types of sessions are counted together in Q. The message representative in the preprocessing mode is computed using a fresh ctx, and the probability that it matches the message representative from the interactive session has already been accounted for in the CRH collision and salt guessing terms; thus, the ϵ_{RO} and ϵ'_{RO} in Equations (7) and (8) can simply be added together.

The authorization semantics defined in Definition 7 are weaker than the definition where "a signature query is considered valid only if the honest party actually sends a response": when an honest initiator has authorized a message, but a corrupt responder refuses to respond, that message is still included in Q. In a two-party scenario, once an honest initiator authorizes a message, it indicates a willingness to sign that message; therefore, this difference is acceptable.

6. Parameter Design and Analysis

6.1 Parameter Selection Method

ThAigis-2P retains all public parameters from Aigis-sig and adds only ν, r, r', S_{thr}, and K. The goal of parameter selection is to maximize the success probability $p_{cand} = p_{acc}^2 p_{glob}$ for each candidate, provided that the statistical parameters satisfy the requirements of Theorem 3. The steps are as follows.

Step 1: Determine the statistical objective. We use $\varepsilon \le 2^{-128}$. Under the candidate budget $Q \le 2^{64}$, the additive statistical term in Equation (7) is at most 2^{-64}. Inverting $2\varepsilon = I_{1-1/\varphi^2}((D+1)/2, 1/2)$ gives the maximum allowed φ, truncated to four decimal places: 3.3202, 3.7450, and 4.1265 for the three parameter sets. The allowed φ increases with dimension for a fixed ε.

Step 2: Determine the displacement bound. For a given ν, sample 20,000 shares from the native distribution, take the median of $Spec_\nu$ as S_{thr}, and let $B = \lceil \sqrt{\tau S_{thr}} \rceil$. Use another 10,000 independent shares for testing; the values of p_E for the three parameter sets are 0.505, 0.493, and 0.497, respectively (95% confidence interval ± 0.01), and the values of $\log_2(1/p_E)$ are all approximately 1. For reference, the typical magnitude of the displacement is

$$B_{typ}^2 = E \parallel L_\nu^{-1}(cu) \parallel_2^2 = \tau n(\frac{l\sigma_1^2}{\nu^2} + k\sigma_2^2), \quad \sigma_i^2 = \frac{\eta_i(\eta_i+1)}{3}. \text{ (9)}$$

Step 3: Search the grid (ν, r). For each r, let $r' = \lceil \sqrt{r^2 + B^2 + 2rB/\varphi} \rceil$ and $M = (r'/r)^D$. When $r \gg B$, $\ln M \approx DB/(\varphi r)$; the larger r is, the higher the acceptance probability. p_{glob} is estimated by Monte Carlo simulation: the accepted responses from the two parties are independently drawn from χ_z, the coefficients of t_0 are sampled uniformly from $(-2^{d-1}, 2^{d-1}]$ (consistent with the distribution of the low bits of the actual public key), and the challenge is sampled uniformly from C_τ. Each sample is checked using Combine. Across different t_0 values, the standard deviation of p_{glob} is approximately 0.01; therefore, the average is taken over 16 to 24 t_0 values for each parameter set.

The main factors influencing p_{glob} can be captured by a normal approximation. Each coordinate of the sum of the two parties' first response components approximately follows a normal distribution with standard deviation $\sigma_z = \nu r\sqrt{2/(D+2)}$, and the probability of passing the norm check is approximately $(1-2\Phi(-(\gamma_1-\beta_1)/\sigma_z))^{nl}$, where Φ is the standard normal cumulative distribution function. The per-coordinate standard deviation of $\delta = z^{(2)} - ct_0$ and the expected hint weight are approximately:

$$\sigma_\delta^2 \approx \frac{2r^2}{D+2} + \tau\frac{2^{2d}}{12}, E[wt(h)] \approx nk\frac{\sqrt{2/\pi}\,\sigma_\delta}{2\gamma_2}. \text{ (10)}$$

The rationale behind the latter expression is as follows: w is approximately uniform and adding a coordinate of δ changes the corresponding high bits with probability approximately $|\delta|/(2\gamma_2)$. Hence, the hint weight constrains only r, the norm check constrains only νr, and a high acceptance probability requires r to be as large as possible. Therefore, the optimal point lies roughly at the maximum r allowed by the hint constraint, with νr chosen close to the limit imposed by the norm constraint.

Step 4: Select K. Using Equation (6), calculate the K required for the success probability of a single attempt to reach 1/2, 0.9, and 0.99. In interactive mode, where communication overhead is the primary cost, $K = 3$ is recommended; in preprocessing mode, where a single failure results in an additional online round trip and consumes one record, $K = 10$ is recommended.

6.2 Values for the Three Sets of Parameters

Table 2 lists the values for the three parameter sets and the model estimates, while Table 3 provides a breakdown of the checks performed by Combine. The values of p_{glob} and p_{cand} in Table 2 are averages over multiple t_0 values.

Table 2: ThAigis-2P Parameters ($\varepsilon \le 2^{-128}$,$p_E \approx 1/2$)

Parameters	**I**	**II**	**III**
Scale Factor ν	5/4	3/4	3/5
Dimension $D = n(k+l)$	1792	2,304	2,816
Typical Displacement B_{typ}	552.0	993.2	1031.4
Spectral Threshold S_{thr}	12,259	36,099	38,423
Displacement Upper Bound B	858	1,472	1,519
B/B_{typ}	1.55	1.48	1.47
φ	3.3202	3.7450	4.1265

Parameters	I	II	III
Target radius r	670,000	1,300,000	1,600,000
Sampling radius r'	670,259	1,300,394	1,600,369
$M = (r'/r)^D$	1.9989	2.0101	1.9143
Acceptance probability per party $p_{acc} \approx 1/M$	0.500	0.497	0.522
Probability of passing Combine p_{glob}	0.895	0.912	0.918
Probability of Success for a Single Candidate p_{cand}	0.224	0.226	0.250
Target success probability per attempt $\geq 1/2$; required K	3	3	3
Target success probability per attempt ≥ 0.9; required K	10	9	8
Target success probability per attempt ≥ 0.99; required K	19	18	16
Ratio of the maximum partial response to $(q-1)/2$ $\nu r'/((q-1)/2)$	0.829	0.504	0.496

Note: For parameter sets II and III with $K = 9$ and $K = 8$, respectively, the model predicts success probabilities of 0.9001 and 0.9003, both close to 0.9.

Table 3: Breakdown of the checks in Combine: Normal approximation and Monte Carlo estimation

Item	I	II	III
Standard deviation of the first response component from one party $\nu r/\sqrt{D+2}$	19,773	20,304	18,084
Combined Standard Deviation σ_z	27,963	28,714	25,575
$(\gamma_1 - \beta_1)/\sigma_z$	4.68	4.56	5.12
Norm check acceptance probability (normal approximation)	0.9978	0.9948	0.9996
Norm check acceptance probability (Monte Carlo)	0.9975	0.9947	0.9997
ct_0 Standard deviation of coordinates	18,318	36,636	36,636

Item	I	II	III
σ_δ	28,913	52,990	56,206
$E\|\delta\|$	23,070	42,280	44,846
$E[wt(h)]$(Equation (10))	70.1	83.9	106.8
$E[wt(h)]$(Monte Carlo)	70.2	84.0	106.7
Hint weight bound ω	80	96	120
Hint check acceptance probability (normal approximation)	0.892	0.916	0.908
Hint check acceptance probability (Monte Carlo)	0.897	0.917	0.918
$\| \delta \|_\infty > \gamma_2$ (normal approximation)	5.8×10^{-6}	1.5×10^{-6}	1.5×10^{-5}

Note: The normal approximation for the hint check treats $wt(h)$ as a normally distributed variable with variance equal to its mean. The Monte Carlo estimates were based on 32,000 to 48,000 samples per group, none of which yielded $\| \delta \|_\infty > \gamma_2$.

The values of the three sets of parameters can be directly interpreted using Equation (10) and the normal approximation. If the failure rate for the hint check is required to be approximately 10%, i.e., $E[wt(h)] + 1.28\sqrt{E[wt(h)]} \approx \omega$, then the values of r solved from Equation (10) are approximately 6.7×10^5, 1.30×10^6, and 1.60×10^6, respectively; if the failure rate for the norm check is required to be at most 0.5%, i.e., $(\gamma_1 - \beta_1)/\sigma_z$ takes values between 4.6 and 5.1, then the values of νr are approximately 8.4×10^5, 9.8×10^5, and 9.6×10^5, respectively. Dividing the two values yields ν, which are approximately 1.25, 0.75, and 0.60, consistent with the results of the grid search.

Parameter set I. In this parameter set, both $\gamma_2 = 168\,448$ and $\omega = 80$ is small, indicating the tightest constraints; r can only reach around 6.7×10^5. $\gamma_1 - \beta_1 = 130\,952$ allows the standard deviation of the combined first response component to be approximately 2.8×10^4; the corresponding νr is larger than r, so $\nu > 1$. The p_{cand} values obtained by grid search within $\nu \in [1.15, 1.35]$ differ by no more than 0.004, which is on the same order of magnitude as the sampling error; this paper adopts $\nu = 5/4$. At this point, $M = 1.999$, the acceptance probability per party is approximately 1/2, and nearly all Combine failures result from the hint weight exceeding the bound (approximately 10%); the failure rate of the norm check is approximately 0.25%. When $K = 3$, the success probability per attempt is 0.533, and when $K = 10$, it is 0.921.

Parameter set II. The $\gamma_2 = 322\,560$ for this parameter set is approximately 1.9 times that of Parameter set I, but $d = 14$ doubles the standard deviation of ct_0 to 36,636, accounting for a significant portion of σ_δ. The hint constraints still allow $r = 1.3 \times 10^6$. At this point, if $\nu = 1$ is used, the standard deviation of the combined first response component is approximately 3.8×10^4, $(\gamma_1 - \beta_1)/\sigma_z \approx 3.4$, and the probability that at least one of the 1,024 coordinates exceeds the bound is close to 50%. When $\nu = 3/4$ is chosen, $\nu r \approx 9.75 \times 10^5$, and the failure rate of the norm check drops to approximately 0.5%. The secret in this parameter set is dominated by the second component, for which $\eta_2 = 5$; $\nu < 1$ has a negligible effect on B; $B/B_{typ} = 1.48$. $M = 2.010$, $p_{cand} = 0.226$; the required K is comparable to that of Parameter set I.

Parameter set III. This set is identical to Parameter set II in terms of q, d, and γ_2, but $k = 6$ increases D to 2816, and the bound $\omega = 120$ is also larger, indicating that the constraints allow $r = 1.6 \times 10^6$. The increase in dimensionality offers two benefits: under the same ε, φ is larger; and under the same νr, σ_z is smaller while $(\gamma_1 - \beta_1)/\sigma_z$ is larger. After selecting $\nu = 3/5$, the norm check almost never fails, $M = 1.914$, and the single-candidate success probability of 0.250 is the highest among the three sets. Within $\nu \in [0.5, 0.65]$, the variation in p_{cand} does not exceed 0.003.

The three sets of parameters scale in different directions, primarily due to the relative strictness of the hint constraints and the norm constraints. For parameter set I, γ_2/γ_1 is approximately 1.3; the hint constraint limits r, so the first component should be scaled up. For parameter sets II and III, γ_2/γ_1 is approximately 2.5; the hint constraint allows for a larger r, so the first component needs to be scaled down. For parameter set III, ν is smaller because both its $\omega/(nk)$ and D are larger. As a comparison, for ML-DSA-44, $\gamma_2 = 95\,232$ is less than γ_1, and Celi et al. (2026) set it to $\nu = 3$.

6.3 Trade-offs Between Displacement Upper Bounds and Statistical Parameters

Table 4 compares different displacement bounds and statistical parameters using the values of ν and r from Table 2. The worst-case upper bound based on l_1 is $B = \tau\sqrt{n(l\eta_1^2/\nu^2 + k\eta_2^2)}$, which follows from $\| cu \|_2 \leq \| c \|_1 \| u \|_2$ and holds for all valid secrets; the heuristic upper bound is set to $1.3B_{typ}$, consistent with the approach in (Celi et al. 2026). In the table, $K_{1/2}$ and $K_{0.9}$ represent the numbers of candidates required to achieve success probabilities per attempt of 1/2 and 0.9, respectively.

Table 4: Effect of Displacement Upper Bounds and Statistical Parameters on the Success Probability for a Single Candidate

Configuration	Parameter Set	B/B_{typ}	φ	M	p_{acc}	p_{cand}	$K_{1/2}$	$K_{0.9}$
This paper: spectral upper bound, $p_E = 1/2$, $\varepsilon = 2^{-128}$	I	1.55	3.3202	1.999	0.500	0.224	3	10
	II	1.48	3.7450	2.010	0.497	0.226	3	9
	III	1.47	4.1265	1.914	0.522	0.250	3	8
Spectral upper bound, $p_E = 0.9$, $\varepsilon = 2^{-128}$	I	1.70	3.3202	2.137	0.468	0.196	4	11

Configuration	Parameter Set	B/B_{typ}	φ	M	p_{acc}	p_{cand}	$K_{1/2}$	$K_{0.9}$
	II	1.61	3.7450	2.135	0.468	0.200	4	11
	III	1.60	4.1265	2.022	0.495	0.225	3	10
Spectral upper bound, $p_E = 1/2$, $\varepsilon = 2^{-64}$	I	1.55	4.7164	1.631	0.613	0.336	2	6
	II	1.48	5.3341	1.634	0.612	0.342	2	6
	III	1.47	5.8875	1.577	0.634	0.369	2	6
$1.3B_{typ}$, $\varepsilon = 2^{-128}$	I	1.30	3.3202	1.787	0.560	0.281	3	7
	II	1.30	3.7450	1.846	0.542	0.268	3	8
	III	1.30	4.1265	1.775	0.563	0.291	3	7
Worst-case upper bound based on l_1, $\varepsilon = 2^{-128}$	I	11.49	3.3202	177.4	0.0056	2.8×10^{-5}	24,379	80,983
	II	11.97	3.7450	302.0	0.0033	1.0×10^{-5}	69,288	230,168
	III	11.92	4.1265	203.5	0.0049	2.2×10^{-5}	31,267	103,866

Table 4 illustrates three points. First, the upper bound based on l_1 is approximately 11.5 to 12 times the typical value, and the acceptance probability drops below 0.6%; even optimizing r again does not make this bound viable. Second, the spectral upper bound is 13%–19% higher than the heuristic upper bound; for the same ε, p_{cand} drops from 0.27–0.29 to 0.22–0.25, and $K_{0.9}$ increases by 1–3. In exchange, the resulting bound holds for all challenges, eliminating the need to handle the event that the displacement exceeds the bound in the proof. Third, the choice of statistical target has a greater impact on the success rate than the choice of bound: relaxing ε to 2^{-64} allows p_{cand} to increase to 0.34–0.37, but this applies only when the candidate budget is substantially below 2^{64}. We adopt the first configuration. Its spectral screening condition contributes approximately one bit of conditioning loss to the security reduction.

For one share of parameter set I, the largest value of $\| L_v^{-1}(cu) \|_2$ among 2,000 random challenges was 0.73 times $\sqrt{\tau Spec_v(u)}$. A local search over the challenge set reached 0.93 times the same bound for this share. A bound on the tail for a random challenge, as discussed in Section 5.1, is another possible approach to reducing B.

6.4 Security Estimates

Theorem 3 reduces unforgeability to three classes of problems. The first class is AMLWE under the native distribution. This is the same distribution used for the corresponding assumption in single-party Aigis-sig. The second class is MLWE under the distributions χ_r and χ_z. The third class is the unforgeability of Aigis-sig itself, which relies on AMLWE and SelfTargetAMSIS; neither the signature verification equation nor the signature verification bounds has changed, so we follow the analysis in (Zhang et al. 2020) here.

This paper estimates the cost of MLWE instances using the core-SVP method (Alkim et al. 2016; Albrecht et al. 2015): for the primal attack, we apply the Bai–Galbraith scaling and optimize the number of samples; for the dual attack, we account for the amortized cost of the $2^{0.2075\beta}$ short vectors produced by a single sieve execution; the classical and quantum costs are taken as $2^{0.292\beta}$ and $2^{0.265\beta}$, respectively, where β is the BKZ block size. Using the same method to estimate the MLWE components of Kyber-512 and Dilithium-2, we obtain $\beta = 405$ and $\beta = 423$, respectively, which are consistent with the values in the specifications of the two schemes (Ducas et al. 2018; Bos et al. 2018). The results are shown in Table 5. The quantum costs under the native distribution are 89.8, 128.8, and 162.7 bits, consistent with the 90, 128, and 163 bits reported in (Zhang et al. 2020). The secret of the joint public key is the sum of the two shares, and the variance doubles: the model estimates that recovering the private key from the joint public key costs approximately 6–10 additional bits. The coordinate standard deviations for χ_r and χ_z exceed 1.6×10^4: the primal attack's success condition is unmet in the tested range when $\beta \leq 2500$; in the dual attack, the relative width of the inner product between the short dual vector and the noise is far greater than 1, and the cost calculated using the same model exceeds 2^{600000}. Furthermore, based on the estimates in Section 5.3 (G_1), the min-entropy of χ_r is at least $\log_2 Vol(B_D(r')) + nl \log_2 \nu$, which is 7,459, 10,201, and 13,103 bits greater than $nk \log_2 q$. These attack-cost estimates and entropy bounds provide evidence about the tested instances. The reduction still treats the two MLWE assumptions as separate computational assumptions.

Table 5: core-SVP Security Estimates

Instance	**Parameter Set**	**Standard Deviation (Secret/Error)**	**Primal attack β**	**Primal attack cost (classical/quantum)**	**Dual Attack Cost (Classical)**
Native Distribution	I	1.41 / 2.00	339	99.0 / 89.8	99.3
	II	1.41 / 3.16	486	141.9 / 128.8	141.9
	III	0.82 / 3.16	614	179.3 / 162.7	178.7
Joint public key	I	2.00 / 2.83	363	106.0 / 96.2	105.7
	II	2.00 / 4.47	517	151.0 / 137.0	150.7
	III	1.15 / 4.47	653	190.7 / 173.0	190.4
χ_r, χ_z	I–III	$\geq 1.6 \times 10^4$	> 2500	—	$> 600\,000$

Note: Cost is measured in bits. The secret in the "Joint Public Key" row is the sum of the two shares; for the rows χ_r and χ_z, no block size within $\beta \leq 2500$ satisfies the conditions for a successful primal attack.

The spectral screen contributes approximately one bit of conditioning loss. The overall unforge-

ability bound also depends on the query budget, the statistical error, and the MLWE and Aigis-sig advantages in Equations (7) and (8).

6.5 Communication and Storage Requirements

Let $|w| = nk\, b_q/8$ denote the number of bytes in a commitment; the values for the three parameter sets are 2688, 3520, and 4224 bytes, respectively. In interactive mode, each party sends a 32-byte commitment digest, commitments occupying $K|w|$ bytes, a bitmap occupying $\lceil K/8 \rceil$ bytes, and an average of Kp_{acc} encoded partial responses (each with a 2-byte length field) per attempt. In preprocessing mode, each party sends commitments occupying $K|w|$ bytes and a 32-byte salt commitment per record during the offline phase; during the online phase, requests contain 84 bytes of fixed fields plus the message itself, while responses contain a 32-byte salt, a bitmap, and encoded partial responses. The communication costs in Table 6 use the measured mean partial response lengths from Section 7 and exclude protocol headers. In interactive mode, these headers add 99 bytes per party per attempt (2 bytes for the version, 1 for the parameter set, and a 32-byte session identifier in each of the three rounds). The preprocessing entries exclude the 16-byte record identifier on each offline message and online response. The complete online request is described separately in Appendix A.

Table 6. Communication per party (bytes, excluding protocol headers, authentication tags, and transport overhead)

Mode and Item	**I**	**II**	**III**
Interactive mode $K = 3$: Per attempt	10,496	13,780	16,839
$K = 3$: Probability of success per attempt	0.533	0.536	0.579
Interactive mode $K = 3$: Expected number of attempts	1.88	1.87	1.73
$K = 3$: Expected communication per signature	19,700	25,712	29,092
$K = 10$: Expected communication per signature	37,910	49,703	59,380
$K = 10$: Offline, per record	26,912	35,232	42,272
$K = 10$: Online response	8,030	10,657	13,815
$K = 10$: Probability of success for a single round trip	0.921	0.923	0.944
Aigis-sig signature length	1852	2,445	3046

Each record stores the party's hyperball samples and the aggregated commitments. When ξ is stored in double precision, each candidate sample occupies $8D$ bytes. For $K = 10$, the samples and commitments together occupy approximately 170 KB (I), 220 KB (II), and 268 KB (III). If

masks are regenerated online from stored 32-byte seeds, the aggregated commitments alone occupy approximately 27–43 KB per record. The reported per-candidate sampling time is 0.17–0.27 ms (Section 7.4). The total storage of a seed-based implementation also depends on the number of seeds and the record metadata. The batch size depends on the storage budget and the number of records that can be discarded after a failure.

7 Reference Implementation

7.1 Implementation

The reference implementation is written in C and uses PQMagic's Aigis-sig implementation (sig/aigis-sig/std) (PQCrypto-CN 2026) for the underlying operations, including matrix expansion, NTT, packing, rounding, the hint function, challenge generation, and the signature verification function. By default, PQMagic instantiates the hash function using the SM3 extension; the compilation option USE_SHAKE allows switching to SHAKE. The experiments in this paper use the latter; the only difference between the two is the instantiation of the hash function. Apart from the header file paths, the Aigis-sig source files were not modified; the SHAKE configuration uses a standalone Keccak permutation.

The implementation adds ShareGen, HSample, LocalRej, Combine, and the encoders and decoders described in Section 4.7. ShareGen computes spectral statistics at $n/2$ roots of unity using a direct discrete Fourier transform, with a relative margin of 10^{-9} to account for floating-point error; HSample generates uniform random numbers using SHAKE-256 and applies the Box–Muller transform to obtain double-precision normal variables. Combine computes $Az - 2^d ct_1$ using the same steps as PQMagic's signature verification function. The test program simulates both parties within a single process and does not include network, authentication, or state persistence services. The implementation is not constant time and does not use vector instructions.

The experiments were run on a virtual machine with two virtual cores on an Intel Core i7-13620H, using Ubuntu 22.04 and GCC 11.4. The code was compiled with -O2 and run in a single thread.

7.2 Functional Testing

First, we verified the underlying code of Aigis-sig. For each of the three parameter sets, we generated 20 pairs of random polynomials; the results of NTT multiplication matched those of negacyclic polynomial multiplication. We performed 2,000 native signing and verification operations for each set; all signatures passed verification, and all were rejected after the message was modified. The public key, private key, and signature lengths matched those in Table 1.

For threshold signatures, 10,000 signatures were generated for each parameter set using $K = 8$, with the key changed every 50 signatures (for a total of 200 key pairs); half of these signatures used a 32-byte random context string. For each parameter set, another 1,500 signatures were generated with each of $K = 3$ and $K = 10$. All 39,000 signatures were accepted by the unmodified PQMagic signature verification function. When the combiner decoded the other party's partial response, the result matched the sender's original vector coefficient by coefficient, with no decoding failures.

The encoder test (for each set of parameters) was divided into four parts.

(1) Encoding–decoding consistency: For 300 partial responses generated by the protocol and 400 manually generated vectors, the decoding results matched the original vectors element-by-element. The manually generated vectors covered a wide range of cases, including uniform distributions, all-zero vectors, extreme values ($\pm(q-1)/2$), normal distributions with two different standard

deviations, sparse outliers, ternary values, and mixed distributions with different scales in the first and second halves.

(2) All 2,800 inputs with truncated or appended bytes were rejected.

(3) All 22 purposefully constructed invalid inputs were rejected, including unknown mode bytes, invalid bit widths, noncanonical encodings written in R mode when C mode should have been used, $u \geq q$, escape branches for small values, invalid block parameters, and non-zero padding bits.

(4) After 11,200 random single-bit flips were applied to valid encodings, the number of rejections ranged from 742 to 870 across the three parameter sets; the remaining flips produced another vector whose canonical encoding was the flipped byte string. This is consistent with the bijection established for the canonical encoding. The encoding does not itself provide integrity; message authentication is required (Section 4.7).

7.3 Rejection Rate and Success Rate

Table 7 compares the model in Section 6 with the results of the $K = 8$ experiment, with approximately 90,000 candidates for each parameter set. The acceptance rate per party, combine acceptance rate, and success rate per candidate agree with the model within sampling error and variation between public keys. Among Combine failures, 93% to 99.8% were due to the hint weight exceeding the bound, while the remainder were caused by norm violations; $\| \delta \|_\infty > \gamma_2$ never occurred. Recalculating the success rate per attempt using the first k candidates from the data for $K = 8$ gave results close to the prediction in Equation (6), consistent with treating correlations between candidates as negligible in this model.

Table 7: Model Estimates vs. Experimental Results (Experimental data: $K = 8$, 10,000 signatures, 200 key pairs)

Item	**I Model**	**I Measured**	**II Model**	**II Measured**	**III Model**	**III Measured**
Acceptance rate per party	0.500	0.501 / 0.498	0.497	0.499 / 0.498	0.522	0.521 / 0.520
Fraction of candidates accepted by both parties	0.250	0.249	0.247	0.248	0.273	0.271
Combine acceptance rate	0.895	0.896	0.912	0.913	0.918	0.912
Success Rate for Single Candidate	0.224	0.223	0.226	0.226	0.250	0.247
Average hint weight (ω)	70.2	70.2 (80)	84.0	84.0 (96)	106.7	106.7 (120)

Item	I Model	I Measured	II Model	II Measured	III Model	III Measured
Combine failures: hint weight / response norm	—	2355 / 38	—	1845 / 139	—	2,114 / 5
Success rate per attempt, $K = 3$	0.533	0.531	0.536	0.538	0.579	0.566
Success rate per attempt, $K = 8$	0.869	0.866	0.871	0.869	0.900	0.898
Success rate per attempt, $K = 10$	0.921	0.930	0.923	0.925	0.944	0.949

Note: The measured value for $K = 10$ is based on a separate experiment with 1,500 signatures. The measured columns for acceptance rates per party show the values for each of the two parties.

By comparison, the native Aigis-sig, on the same implementation, required an average of 5.89, 7.69, and 6.76 iterations per signature (20,000 signatures each), which aligns with the 5.86, 7.61, and 6.67 iterations analyzed in Zhang et al. (2020), indicating a success rate per attempt of 0.13 to 0.17. A single candidate in ThAigis-2P requires simultaneous acceptance by both parties, yet the success rate reaches 0.22–0.25. One important source of this difference is that the ball distribution is more compact than the uniform hypercube distribution used by Aigis-sig (Celi et al. 2026; Devevey et al. 2022); however, the two methods have different rejection checks (the native signature also checks for consistency between low-bit and high-bit values) and different statistical parameters (rejection sampling in the native signature is exact), so these two sets of figures should not be directly compared.

7.4 Computational Overhead

Table 8 shows the average execution time for each operation. Key generation involves two ShareGen operations and public-key computation; ShareGen requires approximately two sampling attempts on average, consistent with $p_E \approx 1/2$. The offline computational overhead for each candidate is primarily due to Gaussian sampling, accounting for about 90%, while the NTT operations required for the commitment take only 14–27 µs.

Table 8: Computational Overhead (Single-threaded, in ms)

Operation	I	II	III
Key generation by a trusted dealer (including two ShareGen operations)	1.12	1.50	1.95
One spectral check	0.22	0.27	0.35
Offline HSample per candidate (including commitments)	0.183	0.244	0.291
Of which, Gaussian sampling and rounding	0.169	0.217	0.271
Online challenge computation per candidate	0.009	0.010	0.012
Online LocalRej per candidate	0.014	0.019	0.022
Encoding per accepted response	0.057	0.081	0.097
Combine (including decoding, per attempt)	0.115	0.156	0.195
Signature verification	0.140	0.225	0.309
Preprocessing mode online, responder ($K = 10$)	0.51	0.69	0.85
Preprocessing mode online, initiator ($K = 10$, including signature verification)	0.48	0.67	0.84
Native Aigis-sig: Key generation / signing / verification	0.145 / 0.442 / 0.138	0.226 / 0.689 / 0.210	0.327 / 0.863 / 0.309

In preprocessing mode, the online computational cost per party is comparable to that of one native Aigis-sig signature; during the offline phase, each record ($K = 10$) takes approximately 1.8–2.9 ms. In interactive mode with $K = 3$, the computation per attempt consists mainly of three hyperball samples in Round 1, taking approximately 0.55–0.87 ms; computations for each party in Rounds 2 and 3 (including verification of the other party's commitment digest, but excluding Combine) take less than 0.4 ms. These figures measure only computation. Actual latency also depends on network communication: preprocessing mode requires one online round trip, while each attempt in interactive mode requires three rounds.

7.5 Encoding Performance

For $K = 8$, approximately 138,000 partial responses were encoded, and the encoder selected C mode in every case. Table 9 compares the lengths of the different representations. Compared with transmitting the original values (R mode), C mode reduces the length by 21% to 25%; compared with the minimum fixed bit width selected from the observed coefficient magnitudes (N mode, 18

bits in this experiment), it reduces the length by 7.5% to 8.5%. Since each partial response is generated by one party and its coefficient magnitudes may exceed 2^{17}, the 18-bit encoding used for the signature response (z) cannot be applied directly.

Table 9: Encoding Lengths for Partial Responses (bytes, per vector, excluding the 2-byte length field)

Representation	**I**	**II**	**III**
R mode (b_q bits per coefficient)	2017	2817	3521
N mode (18-bit fixed length)	1730	2306	2882
C Mode (Measured Average)	1596.3	2,133.4	2,636.1
Reduction compared to R mode	20.9%	24.3%	25.1%
Reduction relative to fixed 18-bit encoding	7.7%	7.5%	8.5%
Standard deviation of partial response coefficients (measured)	19,775	20,303	18,084

In interactive mode, partial responses account for only 23% to 25% of the communication per attempt ($K = 3$), while commitments account for the majority; in preprocessing mode, commitments constitute an offline overhead, and online responses consist almost entirely of encoded partial responses, with the bytes saved through encoding directly reflected in the online communication.

8. Conclusion

This paper presents ThAigis-2P, a two-party threshold signature scheme that produces native Aigis-sig signatures and requires no change to the verification algorithm. Each party holds an independently sampled share, performs imbalanced hyperball rejection sampling locally, and processes multiple candidates in parallel. The combiner then constructs the native hint from public values. Spectral checks during share generation yield a displacement bound that holds for all challenges and is approximately 1.5 times the typical value; conditioning on the spectral check contributes approximately one bit of loss to the security reduction. In the random oracle model, unforgeability in interactive mode reduces to AMLWE under the Aigis-sig key distribution, MLWE under the hyperball distributions, and the unforgeability of Aigis-sig itself. Both modes require states to be used only once and protected against rollback. The success probability per candidate under $\varepsilon \leq 2^{-128}$ ranges from 0.22 to 0.25; in preprocessing mode with $K = 10$, the probability of obtaining a signature in one online round trip exceeds 0.92, and the online computational cost per party is comparable to that of one native signature.

References

Albrecht M R, Player R, Scott S. 2015. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology* 9(3): 169–203.

Alkim E, Ducas L, Pöppelmann T, Schwabe P. 2016. Post-quantum key exchange: a new hope. In: 25th USENIX Security Symposium. Berkeley: USENIX Association; 327–343.

Bai S, Langlois A, Lepoint T, Stehlé D, Steinfeld R. 2015. Improved security proofs in lattice-based cryptography: using the Rényi divergence rather than the statistical distance. In: Advances in Cryptology – ASIACRYPT 2015, Part I, LNCS 9452. Berlin: Springer; 3–24.

Bellare M, Rogaway P. 1993. Random oracles are practical: a paradigm for designing efficient protocols. In: ACM CCS 1993. New York: ACM; 62–73.

Benhamouda F, Lepoint T, Loss J, Orrù M, Raykova M. 2021. On the (in)security of ROS. In: Advances in Cryptology – EUROCRYPT 2021, Part I, LNCS 12696. Cham: Springer; 33–53.

Bienstock A, de Castro L, Escudero D, Polychroniadou A, Takahashi A. 2026. Quorus: efficient, scalable threshold ML-DSA signatures from MPC. In: 35th USENIX Security Symposium. Berkeley: USENIX Association.

Bos J, Ducas L, Kiltz E, Lepoint T, Lyubashevsky V, Schanck J M, Schwabe P, Seiler G, Stehlé D. 2018. CRYSTALS-Kyber: a CCA-secure module-lattice-based KEM. In: 2018 IEEE European Symposium on Security and Privacy. Los Alamitos: IEEE; 353–367.

Boschini C, Takahashi A, Tibouchi M. 2022. MuSig-L: lattice-based multi-signature with single-round online phase. In: Advances in Cryptology – CRYPTO 2022, Part II, LNCS 13508. Cham: Springer; 276–305.

Boschini C, Kaviani D, Lai R W F, Malavolta G, Takahashi A, Tibouchi M. 2025. Ringtail: practical two-round threshold signatures from learning with errors. In: 2025 IEEE Symposium on Security and Privacy. Los Alamitos: IEEE; 149–164.

Brandão L T A N, Peralta R. 2026. NIST first call for multi-party threshold schemes: NIST IR 8214C. Gaithersburg: National Institute of Standards and Technology.

Canetti R, Gennaro R, Goldfeder S, Makriyannis N, Peled U. 2020. UC non-interactive, proactive, threshold ECDSA with identifiable aborts. In: ACM CCS 2020. New York: ACM; 1769–1787.

Celi S, del Pino R, Espitau T, Niot G, Prest T. 2026. Efficient threshold ML-DSA. In: 35th USENIX Security Symposium. Berkeley: USENIX Association; 6305–6324.

Cheon J H, Choe H, Devevey J, Güneysu T, Hong D, Krausz M, Land G, Möller M, Stehlé D, Yi M. 2024. HAETAE: shorter lattice-based Fiat-Shamir signatures. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2024(3): 25–75.

Cozzo D, Smart N P. 2019. Sharing the LUOV: threshold post-quantum signatures. In: Cryptography and Coding – IMACC 2019, LNCS 11929. Cham: Springer; 128–153.

Damgård I, Orlandi C, Takahashi A, Tibouchi M. 2021. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. In: Public-Key Cryptography – PKC 2021, Part I, LNCS 12710. Cham: Springer; 99–130.

del Pino R, Niot G. 2025. Finally! A compact lattice-based threshold signature. In: Public-Key Cryptography – PKC 2025, Part III, LNCS 15676. Cham: Springer; 169–199.

del Pino R, Katsumata S, Maller M, Mouhartem F, Prest T, Saarinen M-J O. 2024. Threshold Raccoon: practical threshold signatures from standard lattice assumptions. In: Advances in Cryptology – EUROCRYPT 2024, Part II, LNCS 14652. Cham: Springer; 219–248.

del Pino R, Espitau T, Niot G, Prest T. 2025. Simple and efficient lattice threshold signatures with identifiable aborts. IACR Cryptology ePrint Archive 2025/871. (Accessed 2026-11-08)

Devevey J, Fawzi O, Passelègue A, Stehlé D. 2022. On rejection sampling in Lyubashevsky's signature scheme. In: Advances in Cryptology – ASIACRYPT 2022, Part IV, LNCS 13794. Cham: Springer; 34–64.

Drijvers M, Edalatnejad K, Ford B, Kiltz E, Loss J, Neven G, Stepanovs I. 2019. On the security of two-round multi-signatures. In: 2019 IEEE Symposium on Security and Privacy. Los Alamitos: IEEE; 1084–1101.

Ducas L, Kiltz E, Lepoint T, Lyubashevsky V, Schwabe P, Seiler G, Stehlé D. 2018. CRYSTALS-Dilithium: a lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2018(1): 238–268.

Dufka A, Kravtšenko S, Laud P, Snetkov N. 2025. Trilithium: efficient and universally composable distributed ML-DSA signing. IACR Cryptology ePrint Archive 2025/675. (Accessed 2026-11-08)

Espitau T, Niot G, Prest T. 2024. Flood and submerse distributed key generation and robust threshold signature from lattices. In: Advances in Cryptology – CRYPTO 2024, Part VII, LNCS 14926. Cham: Springer; 425–458.

Fu Y, Zhao X. 2022. Research on two-party cooperative Aigis-sig digital signature protocol. In: Security and Privacy in New Computing Environments – SPNCE 2021. Cham: Springer; 52–68.

Golomb S W. 1966. Run-length encodings. *IEEE Transactions on Information Theory* 12(3): 399–401.

Katsumata S, Reichle M, Takemure K. 2024. Adaptively secure 5 round threshold signatures from MLWE/MSIS and DL with rewinding. In: Advances in Cryptology – CRYPTO 2024, Part VII, LNCS 14926. Cham: Springer; 459–491.

Kiltz E, Lyubashevsky V, Schaffner C. 2018. A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. In: Advances in Cryptology – EUROCRYPT 2018, Part III, LNCS 10822. Cham: Springer; 552–586.

Komlo C, Goldberg I. 2021. FROST: flexible round-optimized Schnorr threshold signatures. In: Selected Areas in Cryptography – SAC 2020, LNCS 12804. Cham: Springer; 34–65.

Lindell Y. 2017. Fast secure two-party ECDSA signing. In: Advances in Cryptology – CRYPTO 2017, Part II, LNCS 10402. Cham: Springer; 613–644.

Lyubashevsky V. 2009. Fiat-Shamir with aborts: applications to lattice and factoring-based signatures. In: Advances in Cryptology – ASIACRYPT 2009, LNCS 5912. Berlin: Springer; 598–616.

Lyubashevsky V. 2012. Lattice signatures without trapdoors. In: Advances in Cryptology – EUROCRYPT 2012, LNCS 7237. Berlin: Springer; 738–755.

National Institute of Standards and Technology (NIST). 2024. Module-lattice-based digital signature standard: FIPS 204. Gaithersburg: NIST.

PQCrypto-CN. 2026. PQMagic: a post-quantum cryptographic library. Available at `https://github.com/pqcrypto-cn/PQMagic` (Accessed 2026-11-08).

Rambaud M, Roth S, Urban A. 2026. Threshold signatures as-a-service: achieving threshold ML-DSA in one online round. IACR Cryptology ePrint Archive 2026/814. (Accessed 2026-11-08)

Rice R F, Plaunt J R. 1971. Adaptive variable-length coding for efficient compression of spacecraft television data. *IEEE Transactions on Communication Technology* 19(6): 889–897.

Shoup V. 2000. Practical threshold signatures. In: Advances in Cryptology – EUROCRYPT 2000, LNCS 1807. Berlin: Springer; 207–220.

Wen J M, Wang H Z, Liu J H, Zhang H G. 2023. Aitps: a two-party cooperative signature scheme based on asymmetric module lattice problems. *Journal of Computer Research and Development* 60(9): 2137–2151.

Zhang J, Yu Y, Fan S, Zhang Z, Yang K. 2020. Tweaking the asymmetry of asymmetric-key cryptography on lattices: KEMs and signatures of smaller sizes. In: Public-Key Cryptography – PKC 2020, Part II, LNCS 12111. Cham: Springer; 37–65.

A Message Formats

All integer fields use unsigned big-endian encoding. Commitments are packed with b_q bits per coefficient, most significant bit first. The receiver rejects messages with unknown versions, inconsistent lengths, or nonzero reserved bits. The lengths in Tables A1 and A2 exclude authentication tags.

Table A1. Interactive-mode messages per party per attempt

Round	Fields	Length (bytes)
Round 1	Version (2), parameter set (1), *sid* (32), cmt_i (32)	67
Round 2	*sid* (32), W_i ($K\|\mathbf{w}\|$)	$32 + K\|\mathbf{w}\|$
Round 3	*sid* (32), bitmap ($\lceil K/8 \rceil$), then a 2-byte length and encoded vector for each accepted candidate	Variable

Table A2. Preprocessing-mode messages

Phase	Fields	Length (bytes)
Offline (per party per record)	*rid* (16), W_i ($K\|\mathbf{w}\|$), com_i (32)	$48 + K\|\mathbf{w}\|$
Online request	*rid* (16), d_{rec} (32), $salt_a$ (32), message length (4), and message	$84 + \|m\|$

Phase	Fields	Length (bytes)
Online response	rid (16), $salt_b$ (32), bitmap ($\lceil K/8 \rceil$), then a 2-byte length and encoded vector for each accepted candidate	Variable

Each state-service record contains at least rid, the record status, the batch identifier, the signing process identifier, and the digest of the bound request. The private randomness ξ is stored separately. When a record becomes INVALID, discard its private randomness. Once a record becomes SPENT, only the process that acquired it may complete the authorized attempt. Delete ξ at the corresponding step of Algorithm 10.