IND-CCA Lattice Threshold KEM under 30 KiB

Katharina Boudgoust[1], Rafaël del Pino[2], Oleksandra Lapiha[3], and Thomas Prest[2]

katharina.boudgoust@lirmm.fr, rafael.del.pino@pqshield.com, sasha.lapiha.2021@live.rhul.ac.uk, thomas.prest@pqshield.com

[1] CNRS, Université Montpellier, LIRMM – Montpellier, France
[2] PQShield – Paris, France
[3] Royal Holloway University London – London, United Kingdom

Abstract. At Asiacrypt'25, Lapiha and Prest proposed a lattice-based IND-CCA threshold key-encapsulation mechanism (TKEM) obtained from a threshold identity-based encryption (TIBE) and a signature scheme. Their construction relies on a variant of the Boneh-Canetti-Halevi-Katz (BCHK) transform, instantiated with a lattice-based TIBE. However it suffers from large ciphertexts at 540 KiB for $\kappa = 128$ bits of security. We present substantial improvements to their TIBE, resulting in the first concretely efficient lattice-based IND-CCA TKEM, with ciphertexts just under 30 KiB for a threshold $T = 32$, $Q = 2^{45}$ queries, and the same κ. Our design simplifies the original framework by leveraging the power of random oracles already present in their construction. We further enhance efficiency by adopting approximate computations where appropriate and by replacing module-NTRU trapdoors with NTRU trapdoors, achieving a remarkable eighteenfold reduction in ciphertext size. Finally, leveraging recent developments in secret sharing, we ensure the verifiability of key-extraction shares even in the presence of malicious parties.

1 Introduction

Lattices have emerged as the main mathematical family when it comes to building post-quantum cryptographic schemes. In particular, they allow to build efficient key encapsulation mechanisms (KEMs) and public key encryption schemes (PKEs). In 2022, NIST selected a lattice scheme, Kyber [42], as its main standard ML-KEM [38] for a post-quantum KEM.

While IND-CPA security (indistinguishability under chosen-*plaintext* attacks) constitutes a minimal security baseline for KEMs and PKEs, the standard notion is the stronger IND-CCA security (indistinguishability under chosen-*ciphertext* attacks). Many "natural" PKE constructions achieve IND-CPA security out of the box, but not IND-CCA security. To resolve this issue, generic compilers have emerged which take as input an IND-CPA scheme, and output an IND-CCA one. The most popular CPA-to-CCA compiler is the Fujisaki-Okamoto transform [23,24], a very efficient transform that only relies on symmetric building blocks, and is used in ML-KEM and the upcoming NIST standard HQC.

Threshold cryptosystems. Threshold cryptography is a subfield of cryptography in which cryptographic material is split across several parties, who then perform sensitive operations in a distributed manner. In a threshold KEM, the decapsulation key is distributed across N parties, of which at least $T \leq N$ are needed to decapsulate valid ciphertexts. This increases the robustness of the system as well as its resilience to individual users' compromise.

Threshold cryptosystems have many applications. E-voting protocols use them to guarantee tally confidentiality until the close of voting, see for example Cramer et al. [16], or the lattice-based protocol of Aranha et al. [4]. In the blockchain setting, Choudhuri et al. [14] propose them as a way to enhance the privacy of mempools. NIST has released a call for threshold cryptosystems [11], and we expect it will foster more applications.

Building lattice-based threshold KEMs or PKEs has shown to be challenging. While there exist efficient IND-CPA lattice constructions [10,37], achieving IND-CCA security in an efficient manner seems difficult. Indeed, thresholdising the Fujisaki-Okamoto transform requires expensive generic multi-party computation (MPC). Alternative IND-CCA lattice-based threshold constructions have been proposed, usually using heavy machinery such as non-interactive zero-knowledge arguments (NIZKs) [20], threshold fully homomorphic encryption [8] or MPC [32,15]; thus their efficiency is unclear.

A promising CPA-to-CCA compiler is the Boneh-Canetti-Halevi-Katz transform [12,7], or BCHK transform. Boneh, Boyen and Halevi [6] have shown that it enables IND-CCA TPKEs under suitable conditions. To our knowledge, the only BCHK-based lattice threshold construction is due to Lapiha and Prest [33], using a variant of the BCHK transform which they call the BCHK+ transform. However, their ciphertext sizes are prohibitive at 540 KiB.

Our contributions. Our main contribution is an IND-CCA lattice TKEM which significantly outperforms the Lapiha-Prest construction [33].

The BCHK+ transform requires a threshold identity-based encryption scheme (TIBE) with suitable properties. For a threshold T and a number Q of decryption queries, the TIBE ciphertexts of [33] contain 10 ring elements, for a ring $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^d + 1)$ with q scaling roughly in $\Theta(d^5 \sqrt{QT})$. We design a TIBE that is much simpler and more efficient than the one in [33], requiring only 4 ring elements with q scaling roughly in $\Theta(d^{3/2} \sqrt{QT})$. As a result, for a similar $T = 32$ and $Q = 2^{45}$, our encapsulation key (resp. ciphertexts) is around 6.6 KiB (resp. 30 KiB), which is about 7 times (resp. 18 times) shorter than in [33].

Our second contribution is a TIBE construction which provides robustness for almost identical parameters, although it supports a smaller number of decapsulation queries $Q = 2^{25}$. This feature is underpinned by our use of Vandermonde secret sharing [9], a type of secret sharing that guarantees short shares, as opposed to Shamir secret sharing used by [33]. Using Vandermonde secret sharing allows identifying misbehaving parties via very simple procedures (checking a linear identity and a norm bound). Once again, this construction avoids the use of heavy machinery.

1.1 Technical Overview

The BCHK transform takes as input a signature scheme and a threshold identity-based encryption (TIBE) with suitable properties, and outputs an IND-CCA TKEM. The BCHK+ variant by [33] also adds decapsulation consistency, a property that would otherwise be difficult to obtain in a lattice setting. In [33], the TIBE ciphertexts are 540 KiB and therefore account for an overwhelming portion of the TKEM ciphertext size.

ABB- and CHKP-style constructions. We first provide a reminder on the ABB [1] and CHKP [13] (H)IBEs, which underpin the TIBE of [33] and ours. In all four works, the master public key is a pseudorandom matrix $\mathbf{A}$ and a uniform vector t. A trapdoor for $\mathbf{A}$ allows, for any identity id, to compute a short preimage $\mathbf{z}$ such that $\mathbf{F}_{\mathsf{id}} \cdot \mathbf{z} \approx t$, where $\mathbf{F}_{\mathsf{id}}$ is a matrix related to $\mathbf{A}$ which embeds id. Encryption computes a ciphertext:

$$\begin{bmatrix}\mathbf{u}^T & v\end{bmatrix} = r \cdot \begin{bmatrix}\mathbf{F}_{\mathsf{id}} & t\end{bmatrix} + \begin{bmatrix}\mathbf{e}^T & e'\end{bmatrix} + \begin{bmatrix}0 & \mathsf{Encode}(\mathsf{msg})\end{bmatrix}. \qquad (1)$$

Finally, decryption computes $v - \mathbf{u}^T \cdot \mathbf{z} \approx \mathsf{Encode}(\mathsf{msg})$. [1,13] sample $\mathbf{z}$ using Gaussian sampling, while [33] and we sample $\mathbf{z}$ using noise flooding. This has a big impact on parameters, but is orthogonal to our improvements over [33], so we won't go into more details here.

The [33] TIBE. In [1], we have $\mathbf{F}_{\mathsf{id}} = \begin{bmatrix}\mathbf{A}_0 \mid \mathbf{A}_1 + E(\mathsf{id}) \cdot \mathbf{B}\end{bmatrix}$, where $\mathbf{A}_0$ is a (trapdoored) pseudorandom matrix, and $\mathbf{A}_1$ is uniformly random and $\mathbf{B}$ is a gadget matrix. In [33], $\mathbf{F}_{\mathsf{id}} = \begin{bmatrix}\mathbf{A}_0 \mid \mathbf{A}_1 + E(\mathsf{id}) \cdot \mathbf{B} \mid \mathbf{A}_2\end{bmatrix}$ is expanded to the right with an additional matrix $\mathbf{A}_2$, and each matrix $\mathbf{A}_i$ is in $\mathcal{R}_q^{1\times 3}$. $\mathbf{A}_0$ embeds a trapdoor à la Eagle/Plover [43,22] (essentially, a RLWE sample), used in the preimage sampling procedure. $\mathbf{A}_2$ is used to embed a module-NTRU trapdoor in the security proof, which allows to simulate the preimages. Therefore the ciphertext in Eq. (1) has $3 \cdot 3 + 1 = 10$ ring elements.

A more efficient TIBE. In our paper, we take as a starting point a much more efficient TIBE construction, the "ROHIBE" from [13, Section 5.3], then we heavily optimise it.

1. *Simpler embedding.* The main difference of the ROHIBE construction compared to the ABB IBE is that identities are embedded via $\mathbf{F}_{\mathsf{id}} = \begin{bmatrix}\mathbf{A}_0 \mid H(\mathsf{id})\end{bmatrix}$, where the matrix $\mathbf{A}_0$ still embeds a Plover trapdoor and H is modelled as a random oracle. In [1], the identity is embedded as a "tag" of the gadget matrix. The difference of any two tags has to be invertible for preimage sampling to work, hence it requires picking a low-splitting ring that has exponentially many unit elements. As $H(\mathsf{id})$ only requires one ring element, we save two ring elements compared to the embedding $\mathbf{A}_1 + E(\mathsf{id})$ from [33], and do not need any splitting property from the underlying ring.

2. *Fewer matrix blocks.* Our second major design optimisation compared to [33] is that we do not have a matrix $\mathbf{A}_2$. The simpler embedding implies that $H(\mathsf{id})$ can fulfil the role of $\mathbf{A}_2$ and $\mathbf{A}_1 + E(\mathsf{id}) \cdot \mathbf{B}$ at the same time. In the previous construction the matrices $\mathbf{F}_{\mathsf{id}}$ and $\mathbf{F}_{\mathsf{id}'}$ for $\mathsf{id} \neq \mathsf{id}'$ were related by an additive shift so constructing a trapdoor that works for every shift was more complex and incompatible with Hint-LWE leakage. In the current construction each id has its independent matrix extension and its own NTRU trapdoor embedded in $H(\mathsf{id})$. This trapdoor can be used both for Hint-LWE and for the IBE security.
3. *Smaller modulus.* Our simpler design also allows reducing the modulus. Our parameter selection process (Section 7) allows taking $q = \Theta(d^{3/2}\sqrt{QT}\log\delta)$, where δ is an upper bound on the probability that decryption fails. Carrying out a similar analysis for the [33] TIBE would result in a much larger $q = \Theta(d^5\sqrt{QT})$. Thus, our construction allows a smaller q, which in turn allows decreasing d, compounding bandwidth gains.
4. *NTRU trapdoors.* As a result of the previous improvements, an NTRU trapdoor becomes tight enough for the reduction to go through. The security of NTRU trapdoors is better-understood than module-NTRU trapdoors, which is especially important in our setting since q is large and therefore overstretched NTRU attacks need to be considered [2,31]. In our parameter regimes such attack do not apply (Section 7).
5. *Additional optimisations.* We perform a few more optimisations. Recall that [33] takes $\mathbf{A}_0 = \begin{bmatrix}1 & a_0 & b_0\end{bmatrix} \cdot d_0$, where the role of d_0 was to randomise the first entry of $\mathbf{A}_0$. Instead we simply set $d_0 = 1$ and drop the first coefficient of $\mathbf{A}_0$ when generating a ciphertext.
6. *Bit dropping.* While it is well-known that we can perform heavy bit-dropping on the v part of the ciphertext in Eq. (1), we also notice that we can do the same on two coefficient of $\mathbf{u}$. The reason for that is more subtle: decryption computes $\mathbf{u}^T \cdot \mathbf{z}$, where $\mathbf{z} \in \mathcal{R}^4$ is heavily unbalanced, with the first two coefficients being very large, the third very small, and the last in-between. We adapt the bit-dropping of $\mathbf{u}$ accordingly to gain in ciphertext size without impacting correctness.

To summarise, we save 6 ring elements: 2 in Item 1, 3 in Item 2 and 1 in Item 5. Our modulus q and ring degree d are smaller as well (Item 3) due to the simpler construction, and bit-dropping (Item 6) provides further gains.

Obtaining share robustness. Let us recall the joint interactive key extraction of [33]: in the first round, each party i computes their committing value w_i and sends a commitment $\mathsf{cmt}_i = H_{\mathsf{cmt}}(w_i)$ to the other parties. Upon receiving the commitments of the others, everyone reveals in the next round their value w_i. In the last round, each party computes their response z_i. This response prepares a correct reconstruction by scaling each party's Shamir secret key share by the corresponding Lagrange coefficient. Directly giving out such scalings would be vulnerable to key-recovery attacks.

To mitigate this, [33] copies the strategy of Threshold Raccoon [18] and adds blinding one-time masks m_i, which sum to zero during reconstruction. While simple from a proof perspective, such masks come with two main drawbacks. First, they require the parties to possess mutual secret seeds. And more importantly, they make it difficult to detect which party is misbehaving when a failure occurs during reconstruction. Put differently, a party can simply send garbage in the last round, causing a systematic failure in the combine stage, without ever being caught. This problem has been addressed for Threshold Raccoon by either relying on non-interactive zero-knowledge proof systems [40] or by using different types of secret sharing [9].

In this work, we follow the latter approach and replace the Shamir secret sharing used in [33] by the Vandermonde secret sharing (VSS) [9], a lattice-friendly adaptation of a secret sharing by Desmedt et al. [19]. This technique allows for sharing short values into short shares and with short reconstruction coefficients. The security relies on the Hint-RLWE problem. As the VSS allows for short responses z_i, we can now verify the correctness of each party's response z_i, detecting misbehaving parties. This leads to what we coined extraction share robustness: even if the adversary maliciously generates the key extraction shares, they can not cause the reconstruction of honestly generated ciphertexts to fail. We actually achieve a slightly stronger notion, where this is true even for ciphertexts that are generated with biased randomness.

Note that the security proof of the resulting TIBE now needs to handle the leakage coming from the VSS. To make the proof simpler, we blend the Coset-Hint-RLWE to Hint-RLWE reduction from [33] directly into the security proof. Hence, the Coset-Hint-RLWE now only appears indirectly in our paper.

Stronger decapsulation consistency. Finally, we realise that the extraction share robustness of the underlying TIBE allows us to generically reach a stronger notion of decapsulation consistency in the resulting TKEM from the BCHK+ transform compared to [33]. Our notion corresponds to the one defined in [20] in the context of threshold public-key encryption schemes.

Whereas [33] only counted as decapsulation inconsistencies when two unequal but *valid* keys $K \neq K'$ were recovered, we now also count as decapsulation inconsistency when one of the reconstructed keys is *invalid*, i.e. led to $\perp$. Let us try to give some intuition on why robustness of the TIBE and consistency of the TKEM are related. If exactly one of K or K' is equal to $\perp$, we can use the valid key and the BCHK+ transform to recover the underlying message and encryption randomness used to generated the TIBE ciphertext. Then, the shares associated to the other (invalid) key define a forgery for the extraction share robustness game of the underlying TIBE. If neither K nor K' are equal to $\perp$, we fall back to the case considered in [33]. We provide more intuition on the different decapsulation consistency notions in Section 6.

2 Preliminaries

We assume familiarity with Landau's asymptotic notations: $O(n)$, o(n), $\omega(n)$, $\Omega(n)$, $a(n) \sim b(n)$, and Knuth's Theta notation $\Theta(n)$. We denote by log the natural logarithm and by $\log_2$ the base 2 logarithm.

Algorithms. To denote the *assign* operation, we use $y \coloneqq f(x)$ (resp. $y \leftarrow f(x)$) when f is deterministic (resp. randomised). We note **assert** the function that takes a proposition x as input and $\perp$ if x is false, else it does nothing.

Norms. For a vector $\mathbf{x} = (x_0, \ldots, x_{n-1}) \in \mathbb{R}^n$ we define the norms as $\|\mathbf{x}\| \coloneqq \sqrt{\sum_i x_i^2}$ and $\|\mathbf{x}\|_\infty \coloneqq \max_i |x_i|$. For a matrix we denote the Gram-Schmidt orthogonalisation of $\mathbf{M} \in \mathbb{R}^{n\times m}$ as matrix $\tilde{\mathbf{M}} \in \mathbb{R}^{n\times m}$ with columns $\tilde{\mathbf{m}}_i$, $i \in [n]$. Then the GS norm of a matrix $\mathbf{M} \in \mathbb{R}^{n\times m}$ is defined as $\|\mathbf{M}\|_{GS} = \max_i \|\tilde{\mathbf{m}}_i\|$.

Set and distributions. For an integer $N > 0$, we denote $[N] = \{0, \ldots, N-1\}$. We recall that the union $S \cup T$ of two sets writes $S \sqcup T$ when disjoint. For a set S, we denote by $\mathcal{U}(S)$ the *uniform distribution* over S, and write $x \leftarrow S$ as shorthand for $x \leftarrow \mathcal{U}(S)$.

Decomposition function. For an odd prime q and any positive $\beta < q$, we define the function $\mathsf{Decomp}_\beta : \mathbb{Z}_q \to \mathbb{Z} \times \mathbb{Z}$ in Algorithm 1. We extend it to $\mathcal{R} = \mathbb{Z}[x]/(x^n + 1)$ and its quotient $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$ by decomposing the elements coefficient-wise. It holds that $\|c_1\|_\infty \leq \beta/2$, $\|c_0\|_\infty \leq \left\lceil \frac{q-1}{2\beta} \right\rceil$ and $\mathbb{E}[c_0] = 0$.

Algorithm 1 $\mathsf{Decomp}_\beta(c) \to (c_0, c_1)$ ▷ Assume $c \in \{-\frac{q-1}{2}, \ldots, \frac{q-1}{2}\}$.

```
1: $c_0 \coloneqq \mathrm{sgn}(c) \cdot \lfloor |c|/\beta \rceil$    ▷ sgn(·) is the sign function. Round half up.
2: $c_1 \coloneqq c - c_0 \cdot \beta$
3: return $(c_0, c_1)$
```

Gaussians and lattices. For $\Sigma \in \mathbb{R}^{n\times n}$ a positive definite matrix, we denote $\rho_{\sqrt{\Sigma},\mathbf{c}}$ the mass function over $\mathbb{R}^n$ of the Gaussian distribution centered in $\mathbf{c}$:

$$\rho_{\sqrt{\Sigma},\mathbf{c}}(\mathbf{x}) = \exp(-(\mathbf{x} - \mathbf{c})^{\mathsf{T}} \cdot \Sigma^{-1} \cdot (\mathbf{x} - \mathbf{c})/2)$$

When $\sigma \in \mathbb{R}^+$, we note $\rho_\sigma = \rho_{\sqrt{\sigma^2 I},\mathbf{0}}$.

A lattice is a discrete subgroup of $\mathbb{R}^n$. The dual of a lattice Λ, noted $\Lambda^\vee$, is the set $\Lambda^\vee = \{\mathbf{x} \in \mathsf{Span}_\mathbb{R}(\Lambda) \mid \forall \mathbf{v} \in \Lambda, \langle x, v\rangle \in \mathbb{Z}\}$. Discretizing the Gaussian mass function $\rho_{\sqrt{\Sigma},\mathbf{c}}$ over Λ and normalizing its sum to 1, we obtain the discrete Gaussian distribution $D_{\Lambda,\sqrt{\Sigma},\mathbf{c}}$. For any lattice Λ, and any real $\varepsilon > 0$, the smoothing parameter $\eta_\varepsilon(\Lambda)$ is the smallest real $s > 0$ such that $\rho_{1/(\sqrt{2\pi}s)}(\Lambda^\vee) \leq 1+\varepsilon$. For $n > 0$ and the integer lattice $\mathbb{Z}^n$ we have a bound $\eta_\varepsilon(\mathbb{Z}^n) \leq \frac{1}{\sqrt{2\pi}} \cdot \sqrt{\ln(2n \cdot (1+1/\varepsilon))/\pi}$.

Lemma 1 ([25, Lemma 3.1]). *For a full-rank matrix $\mathbf{B} \in \mathbb{R}^{n\times n}$ and a lattice $\Lambda = \Lambda(\mathbf{B})$ we have*

$$\eta_\varepsilon(\Lambda(\mathbf{B})) \leq \|\mathbf{B}\|_{GS} \cdot \eta_\varepsilon(\mathbb{Z}^n)$$

Lemma 2 ([36] Corollary 7.5). *For $k > 1$, let $\mathbf{a} \leftarrow \mathcal{R}_q^k$, let $\sigma \geq \sqrt{2\pi} \cdot 2d \cdot q^{1/k+2/dk}$. Then with $\tilde{\mathbf{a}} = [1\ \mathbf{a}]$, and $\mathbf{x} \leftarrow D_{\mathcal{R}^4,\sigma}$:*

$$\mathrm{SD}(\tilde{\mathbf{a}}^T\mathbf{x}, \mathcal{U}(\mathcal{R}_q)) = 2^{-\Omega(d)}.$$

Lemma 3 ([25, Corollary 2.8]). *For $n > 0$ and $\varepsilon \in (0,1)$, let $\Lambda' \subseteq \Lambda \subset \mathbb{R}^n$ be full-rank lattices. Let $\sigma \geq \eta_\varepsilon(\Lambda')$. Then*

$$\mathrm{SD}(\mathcal{D}_{\Lambda,\sigma} \bmod \Lambda', \mathcal{U}(\Lambda/\Lambda')) \leq 2\,\varepsilon.$$

Lemma 4 (Implicit in [13]). *Let $\varepsilon \in (0,1/2)$, and $n, m_0, m_1 > 0$ be integers such that $n < m_0 + m_1$. Let $\mathbf{F} = [\mathbf{F}_0 \mid \mathbf{F}_1] \in \mathcal{R}_q^{n\times(m_0+m_1)}$, $\mathbf{t} \in \mathcal{R}_q^n$. Let $\mathbf{\Sigma}_0 \in \mathbb{R}^{dm_0\times dm_0}, \mathbf{\Sigma}_1 \in \mathbb{R}^{dm_1\times dm_1}$ be positive definite such that $s_{\min}(\mathbf{\Sigma}_1) \geq \eta_\varepsilon(\mathbf{F}_1)$. Denote $\mathbf{\Sigma} := \begin{bmatrix} \mathbf{\Sigma}_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{\Sigma}_1 \end{bmatrix}$. Then the following distributions are 2ε-statistically close:*

$$\mathcal{D}_0 = \{(\mathbf{z},\mathbf{x}) \leftarrow \mathcal{D}_{\Lambda_q^{\mathbf{t}}(\mathbf{F}),\mathbf{\Sigma}}\}$$
$$\mathcal{D}_1 = \{(\mathbf{z},\mathbf{x}) \mid \mathbf{z} \leftarrow \mathcal{D}_{\mathcal{R}^{m_0},\mathbf{\Sigma}_0}, \mathbf{x} \leftarrow \mathcal{D}_{\Lambda_q^{\mathbf{t}-\mathbf{F}_0\mathbf{z}}(\mathbf{F}_1),\mathbf{\Sigma}_1}\}$$

Lemma 5 (Adapted [39]). *Let $m, \mathbf{\Sigma}_0, \mathbf{\Sigma}_1 > 0$, $\varepsilon \leq 1/2$ s.t. $\sqrt{\mathbf{\Sigma}_0} \geq \eta_\varepsilon(\mathcal{R}^m)$. Denote $\mathbf{\Sigma}_2^{-1} := \mathbf{\Sigma}_0^{-1} + \mathbf{\Sigma}_1^{-1}$ and assume that $\sqrt{\mathbf{\Sigma}_2} \geq \eta_\varepsilon(\mathcal{R}^m)$ and $\mathbf{x}_i \leftarrow \mathcal{D}_{\mathcal{R}^m,\sqrt{\mathbf{\Sigma}_i}}$, $i \in [2]$. Then*

$$\mathrm{SD}(\mathbf{x}_0 + \mathbf{x}_1, \mathcal{D}_{\mathcal{R}^m,\sqrt{\mathbf{\Sigma}_0+\mathbf{\Sigma}_1}}) \leq 8\varepsilon.$$

Definition 1 (NTRU). *For an integer q, a distribution χ over $\mathcal{R}$, and an algorithm $\mathcal{A}$, the advantage of the decisional NTRU problem of $\mathcal{A}$ is defined as:*

$$\mathsf{Adv}^{\mathsf{NTRU}}(\mathcal{A}) = \left|\Pr\left[1 \leftarrow \mathcal{A}(f \cdot g^{-1})\right] - \Pr\left[1 \leftarrow \mathcal{A}(h)\right]\right|$$

where $(f,g) \leftarrow \chi^2$ conditioned on g being invertible in $\mathcal{R}_q$, and $h \leftarrow \mathcal{R}_q$. We say the $\mathsf{NTRU}_{\chi,q}$ assumption holds if $\mathsf{Adv}^{\mathsf{NTRU}}(\mathcal{A})$ is negligible for all PPT $\mathcal{A}$.

Lemma 6 (NTRU Trapdoor quality). *Let $\sigma = O(\sqrt{q})$, there exists a randomised algorithm $\mathsf{TrapGenNTRU}(1^d, q)$ that outputs a vector $\mathbf{a} := \begin{bmatrix} 1 & a' \end{bmatrix} \in \mathcal{R}_q^2$ and a full rank matrix $\mathbf{T_a} \in \mathcal{R}^{2\times 2}$, where $\mathbf{T_a}$ is a basis for $\Lambda_q^\perp(\mathbf{a})$. Moreover, $\|\mathbf{T_a}\|_{GS} = O(\sqrt{q})$ and a' is indistinguishable from random based on the $\mathsf{NTRU}_{\mathcal{D}_\sigma,q}$ assumption. In practice we can use $\|\mathbf{T_a}\|_{GS} = 1.17\sqrt{q}$*

2.1 Syntax and Security Definitions

We recall relevant cryptographic primitives used in this work.

Definition 2 (Signature). *A signature scheme is a tuple of PPT algorithms* $\mathsf{SIG} = (\mathsf{SIG.Keygen}, \mathsf{SIG.Sign}, \mathsf{SIG.Verify})$ *s.t.:*

1. $\mathsf{SIG.Keygen}(\kappa) \to (\mathsf{sk}, \mathsf{vk})$ *takes as input the security parameter* κ *and outputs a pair of secret key* sk *and verification key* vk.
2. $\mathsf{SIG.Sign}(\mathsf{sk}, \mathsf{msg}) \to \mathsf{sig}$ *takes as input a secret key* sk *and a message* msg *and outputs a signature* sig.
3. $\mathsf{SIG.Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig}) \to b \in \{0, 1\}$ *takes as input a verification key* vk*, a message* msg *and a signature* sig *and outputs either* 1 *(meaning accept) or* 0 *(meaning reject).*

Definition 3 ((Q, ε)-sEU-CMA Security). *A signature scheme* SIG *satisfies* (Q, ε)-$\mathsf{sEU\text{-}CMA}$ *if any PPT adversary* $\mathcal{A}$ *that makes at most* Q *signing queries wins* $\mathsf{Exp}^{\mathsf{sEU\text{-}CMA}}_{\mathcal{A},\mathsf{SIG}}(1^\kappa)$ *(see Fig. 1a) with probability at most* ε*. When* $Q = 1$ *we say that* SIG *is an* ε-$\mathsf{sEU\text{-}CMA}$ *one-time signature scheme.*

Definition 4 (IBE with Threshold Key Generation). *An Identity-based Encryption Scheme with Threshold Key Generation is a tuple of PPT algorithms* $\mathsf{TIBE} = (\mathsf{TIBE.Setup}, \mathsf{TIBE.Encrypt}, (\mathsf{TIBE.ShareExtract}_r)_r, \mathsf{TIBE.ShareVerify}, \mathsf{TIBE.Combine})$ *s.t.:*

1. $\mathsf{TIBE.Setup}(1^\kappa) \to (\mathsf{ek}, \{\mathsf{dk}_i\}_{i \in N})$ *takes as input the security parameter. It outputs the encryption key* ek *and* N *decryption key shares* $\{\mathsf{dk}_i\}_{i \in N}$.
2. $\mathsf{TIBE.Encrypt}(\mathsf{ek}, \mathsf{id}, \mathsf{msg}) \to \mathsf{ct}$ *takes as input an encryption key* ek*, identity* id *and message* msg*. It outputs a ciphertext* ct.
3. *For* $r \in \{1, \ldots, \mathsf{rounds}\}$*,* $\mathsf{TIBE.ShareExtract}_r(\mathsf{id}, \mathsf{act}, \mathsf{dk}_i, \{\mathsf{contrib}_{k,j}\}^{k \in \mathsf{act}}_{j<r}) \to \mathsf{contrib}_{i,r}$ *takes as input an identity* id*, the set of active participants* act*, a decryption key share* dk_i *and contributions from previous rounds* $\{\mathsf{contrib}_{k,j}\}^{k \in \mathsf{act}}_{j<r}$*. It outputs a contribution of the current round* $\mathsf{contrib}_{i,r}$.
4. $\mathsf{TIBE.ShareVerify}(\mathsf{ek}, \mathsf{id}, \mathsf{act}, \{\mathsf{contrib}_{j,r}\}_{j,r}, i) \to b \in \{1, \perp\}$ *takes as input an encryption key* ek*, an identity* id*, a set of active participants* act*, their key share contributions* $\{\mathsf{contrib}_{j,r}\}^{j \in \mathsf{act}}_{r \leq \mathsf{rounds}}$ *and the index of one party* $i \in \mathsf{act}$*. It outputs either* 1 *(meaning accept) or* $\perp$ *(meaning reject).*
5. $\mathsf{TIBE.Combine}(\mathsf{ek}, \mathsf{ct}, \mathsf{id}, \mathsf{act}, \{\mathsf{contrib}_{i,r}\}_{i,r}) \to \mathsf{msg}/\perp$ *takes as input a set of active participants* act*, their key share contributions* $\{\mathsf{contrib}_{i,r}\}^{i \in \mathsf{act}}_{r \leq \mathsf{rounds}}$*, an identity* id *and a ciphertext* ct*. It outputs a message* msg *or an error* $\perp$.

Definition 5 (Decryption Correctness). *A* TIBE *scheme has a* δ*-correct decryption if for every message* msg*, key* $(\mathsf{ek}, \{\mathsf{dk}_i\}_{i \in N})$ *and set* act *of key extractors of size at least* T *over the randomness of the honest encryption, key extraction and decryption algorithms*

$$\Pr_{\mathsf{ct},\mathsf{contrib}}(\mathsf{TIBE.Combine}(\mathsf{ek}, \mathsf{ct}, \mathsf{id}, \mathsf{act}, \mathsf{contrib}) = \mathsf{msg}) \geq 1 - \delta.$$

Here we denote $\mathsf{ct} := \mathsf{TIBE.Encrypt}(\mathsf{ek}, \mathsf{id}, \mathsf{msg})$ *and* $\mathsf{contrib} := \{\mathsf{contrib}_{j,r}\}^{j \in \mathsf{act}}_{r \leq \mathsf{rounds}}$ *from the partial key extraction procedure.*

Definition 6 (Extraction Share Correctness). *A* TIBE *scheme has* δ'*-correct extraction shares if for every message* msg*, key* $(\mathsf{ek}, \{\mathsf{dk}_i\}_{i\in N})$*, set* act *of key extractors of size at least* T *and every party* $i \in \mathsf{act}$ *over the randomness of the honest encryption and key extraction algorithms*

$$\Pr_{\mathsf{ct},\mathsf{contrib}}(\mathsf{TIBE.ShareVerify}(\mathsf{ek}, \mathsf{id}, \mathsf{act}, \mathsf{contrib}, i) = 1) \geq 1 - \delta'.$$

Here we denote $\mathsf{ct} := \mathsf{TIBE.Encrypt}(\mathsf{ek}, \mathsf{id}, \mathsf{msg})$ *and* $\mathsf{contrib} := \{\mathsf{contrib}_{j,r}\}^{j\in\mathsf{act}}_{r\in\mathsf{rounds}}$ *from the shared key extraction procedure.*

Definition 7 (Extraction Share Robustness). *Let* TIBE *be a threshold identity-based encryption scheme with* RDistr *beings the distribution of its encryption randomness. Further, let* RSpace *be a set such that* $R \leftarrow \mathsf{RDistr}$ *lies in* RSpace *with overwhelming probability. A* TIBE *scheme is extraction share robust for randomness space* RSpace *if no PPT adversary can win the* Robust *experiment in Figure 1e with a non-negligible probability. We write* $\mathsf{Adv}^{\mathsf{Robust}}_{\mathcal{A},\mathsf{TIBE}}$ *for the winning probability of adversary* $\mathcal{A}$ *in the* Robust *experiment.*

Definition 8 (γ-spreadness adapted from [26]). *Let* TIBE *be a Threshold IBE scheme defined in Definition 4 and let* $\mathcal{K}, \mathcal{M}, \mathcal{I}, \mathcal{C}$ *denote the sets of valid encryption keys, messages, identities, and ciphertexts respectively. For* $\gamma > 0$ *the* TIBE *is* γ*-spread if for every tuple* $(\mathsf{msg} \in \mathcal{M}, \mathsf{id} \in \mathcal{I}, \mathsf{ek} \in \mathcal{E})$

$$-\log \max_{\mathsf{ct}\in\mathcal{C}} \Pr_{\mathsf{rand}}(\mathsf{ct} = \mathsf{TIBE.Encrypt}(\mathsf{msg}, \mathsf{id}, \mathsf{ek}; \mathsf{rand})) \geq \gamma.$$

Definition 9 (Threshold Key Encapsulation Mechanism). *A threshold key encapsulation mechanism (TKEM) is a tuple of PPT algorithms* $\mathsf{TKEM} = (\mathsf{TKEM.Keygen}, \mathsf{TKEM.Encaps}, (\mathsf{TKEM.ShareDecaps}_r)_r, \mathsf{TKEM.ShareVerify}, \mathsf{TKEM.Combine})$ *s.t.:*

1. $\mathsf{TKEM.Keygen}(1^\kappa) \rightarrow (\mathsf{ek}, \{\mathsf{dk}_i\}_{i\in N})$ *takes as input the security parameter. It outputs the encapsulation key* ek *and* N *decapsulation key shares* $\{\mathsf{dk}_i\}_{i\in N}$.
2. $\mathsf{TKEM.Encaps}(\mathsf{ek}) \rightarrow \mathsf{ct}, K$ *takes as input an encapsulation key* ek*. It outputs a ciphertext* ct *and a session key from the key space* $K \in \mathcal{K}$.
3. *For* $r \in \{1, \ldots, \mathsf{rounds}\}$*,* $\mathsf{TKEM.ShareDecaps}_r(\mathsf{ct}, \mathsf{act}, \mathsf{dk}_i, \{\mathsf{contrib}_{i,j}\}_{j<r}) \rightarrow \mathsf{contrib}_{i,r}$ *takes as input a ciphertext* ct*, a set of active parties* act*, a decapsulation key share* dk_i *and all contributions by parties in* act *up to the current round* r*. It outputs a contribution* $\mathsf{contrib}_{i,r}$ *for the current round.*
4. $\mathsf{TKEM.ShareVerify}(\mathsf{ek}, \mathsf{ct}, \mathsf{act}, \{\mathsf{contrib}_{j,r}\}_{j,r}, i) \rightarrow b \in \{1, \perp\}$ *takes as input an encapsulation key* ek*, a ciphertext* ct*, a set of active parties* act*, their contributions* $\{\mathsf{contrib}_{j,r}\}_{j\in\mathsf{act},r\leq\mathsf{rounds}}$ *and the index of one party* $i \in \mathsf{act}$*. It outputs either* 1 *(meaning accept) or* $\perp$ *(meaning reject).*
5. $\mathsf{TKEM.Combine}(\mathsf{ek}, \mathsf{ct}, \mathsf{act}, \{\mathsf{contrib}_{i,r}\}_{i,r}) \rightarrow K/\perp$ *takes as input a ciphertext* ct*, a set of active parties* act *and their contributions* $\{\mathsf{contrib}_{i,r}\}_{i\in\mathsf{act},r\leq\mathsf{rounds}}$*. It outputs a key* K *or an error message* $\perp$.

Definition 10 (Selective-ID/CCA2 Security). *A* TIBE *(resp. a* TKEM*) scheme achieves selective-ID indistinguishability, selective-ID one-wayness (resp.*

selective-CCA2 security) if any PPT adversary wins the experiment in Fig. 1b, Fig. 1d (resp. Fig. 1c) with negligible probability. We denote the winning probability of $\mathcal{A}$ as $\mathsf{Adv}^{\mathsf{sel\text{-}ID\text{-}IND}}_{\mathcal{A},\mathsf{TIBE}}$, $\mathsf{Adv}^{\mathsf{sel\text{-}ID\text{-}OW}}_{\mathcal{A},\mathsf{TIBE}}$ (resp. $\mathsf{Adv}^{\mathsf{CCA2}}_{\mathcal{A},\mathsf{TKEM}}$).

2.2 The BCHK+ Transform

We recall in Fig. 2 the BCHK+ transform proven in [33].

Theorem 1 ([33]). *Let* TIBE *be a threshold identity-based encryption from Definition 10. Let $\mathcal{M}$ and* RSpace *denote the message space and randomness space of the* TIBE*, respectively. Let $\mathcal{K}$ be an arbitrary set. Let* SIG *be a one-time signature satisfying $\varepsilon_{\mathsf{SIG}}$-sEU-CMA and $G_{\mathsf{fo}} : \{0,1\}^* \to \mathcal{R}$ and $H_{\mathsf{fo}} : \{0,1\}^* \to \mathcal{K}$ be modelled as random oracles. Then* TKEM *as described in Fig. 2 is a CCA-secure TKEM with:*

$$\mathsf{Adv}^{\mathsf{CCA2}}_{\mathcal{A},\mathsf{TKEM}}(\kappa) \le \varepsilon_{\mathsf{SIG}} + (Q_{G_{\mathsf{fo}}} + Q_{H_{\mathsf{fo}}}) \cdot \mathsf{Adv}^{\mathsf{sel\text{-}ID\text{-}IND}}_{\mathcal{A},\mathsf{TIBE}} + \frac{(Q_{G_{\mathsf{fo}}} + Q_{H_{\mathsf{fo}}})}{|\mathcal{M}|}.$$

2.3 Bounds on Distributions

We use the following results by Lyubashevsky [35] and implications thereof.

Lemma 7 (Lemma 4.3 in [35]). *Let $\mathbf{v} \in \mathbb{R}^d$.*

$$\mathbb{P}[|\langle \mathbf{z}, \mathbf{v}\rangle| > \tau\sigma\|\mathbf{v}\|; \mathbf{z} \leftarrow D_{\mathbb{Z}^d,\sigma}] \le 2e^{-\tau^2/2}. \qquad (2)$$

In particular, if $\tau \ge \sqrt{2(\kappa+1)\log 2}$, then the right side of Eq. (2) is at most $2^{-\kappa}$.

Lemma 8 (Lemma 4.4 in [35], extended). *For any $\tau > 1$ and any lattice $\Lambda \subseteq \mathbb{R}^d$,*

$$\mathbb{P}\left[\|\mathbf{z}\| > \tau\sigma\sqrt{d}; \mathbf{z} \leftarrow D_{\Lambda,\sigma}\right] \le C^d. \qquad (3)$$

*where $C = \tau \cdot e^{\frac{1}{2}(1-\tau^2)} < 1$. If $\left(\tau \ge 1 + \frac{\kappa \log 2}{3d}\right)$ **or** $\left(\tau \ge 1 + \sqrt{\frac{4\,\kappa \log 2}{d}}\right)$, we note that $C^d \le 2^{-\kappa}$.*

The original formulation of [35, Lemma 4.4] takes $\Lambda = \mathbb{Z}^d$, but we can see that its proof goes through for any lattice $\Lambda \subset \mathbb{R}^d$. Corollary 1 is a direct corollary of Lemma 7.

Lemma 9. *For $\mathbf{u} \in \mathcal{R}^k$ and $\mathbf{v} \leftarrow \mathcal{D}^k_{\mathcal{R},\sigma}$, we have:*

$$\mathbb{P}\left[\left\|\mathbf{u}^T\mathbf{v}\right\|_\infty > \|\mathbf{u}\|\sigma\sqrt{2\log\left(\frac{2dk}{t}\right)}\right] \le t$$

```
$\mathsf{Exp}^{\mathsf{sEU\text{-}CMA}}_{\mathcal{A},\mathsf{SIG}}(1^\kappa)$
$(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SIG.Keygen}(1^\kappa)$
$(\mathsf{msg}^*, \mathsf{sig}^*) \leftarrow \mathcal{A}^{\mathsf{SIG.Sign}(\mathsf{sk},\cdot)}(\mathsf{vk})$
Let $\{(\mathsf{msg}_i, \mathsf{sig}_i)\}_{i=0}^{Q-1}$ be the list of $\mathcal{A}$'s oracle queries
return $(\mathsf{SIG.Verify}(\mathsf{vk}, \mathsf{msg}^*, \mathsf{sig}^*) = 1) \wedge (\mathsf{msg}^*, \mathsf{sig}^*) \notin \{(\mathsf{msg}_i, \mathsf{sig}_i)\}_{i=0}^{Q-1}$
```

(a) The (Q, ε)-sEU-CMA security game

```
$\mathsf{Exp}^{\mathsf{sel\text{-}ID\text{-}IND}}_{\mathcal{A},\mathsf{TIBE}}(1^\kappa)$
$(\mathsf{id}^*, \mathsf{cor}) \leftarrow \mathcal{A}$
assert $|\mathsf{cor}| = T - 1$
$(\mathsf{ek}, \{\mathsf{dk}_i\}_{i \in N}) \leftarrow \mathsf{TIBE.Setup}(1^\kappa)$
$(\mathsf{msg}_0, \mathsf{msg}_1)$
$\leftarrow \mathcal{A}^{\mathsf{TIBE.ShareExtract}(\cdot,\mathsf{id}),\mathsf{id} \neq \mathsf{id}^*}(\{\mathsf{dk}_i\}_{\mathsf{cor}})$
$b \leftarrow \{0, 1\}$
$\mathsf{ct}^* = \mathsf{TIBE.Encrypt}(\mathsf{ek}, \mathsf{id}^*, \mathsf{msg}_b)$
$b' \leftarrow \mathcal{A}^{\mathsf{TIBE.ShareExtract}(\cdot,\mathsf{id}),\mathsf{id} \neq \mathsf{id}^*}(\{\mathsf{dk}_i\}_{\mathsf{cor}})$
return $(b = b')$
```

(b) TIBE selective-ID security

```
$\mathsf{Exp}^{\mathsf{CCA2}}_{\mathcal{A},\mathsf{TKEM}}(1^\kappa)$
$\mathsf{cor} \leftarrow \mathcal{A}$
assert $|\mathsf{cor}| = T - 1$
$(\mathsf{ek}, \{\mathsf{dk}_i\}_{i \in N}) \leftarrow \mathsf{TKEM.Keygen}(1^\kappa)$
$1 \leftarrow \mathcal{A}^{\mathsf{TKEM.ShareDecaps}(\cdot)}(\{\mathsf{dk}_i\}_{\mathsf{cor}})$
$(\mathsf{ct}^*, K_0) = \mathsf{TKEM.Encaps}(\mathsf{ek})$
$K_1 \leftarrow \mathcal{K}$
$b \leftarrow \{0, 1\}$
$b' \leftarrow \mathcal{A}^{\mathsf{TKEM.ShareDecaps}(\mathsf{ct},\cdot),\mathsf{ct} \neq \mathsf{ct}^*}((\mathsf{ct}^*, K_b))$
return $(b = b')$
```

(c) TKEM selective security

```
$\mathsf{Exp}^{\mathsf{sel\text{-}ID\text{-}OW}}_{\mathcal{A},\mathsf{TIBE}}(1^\kappa)$
$(\mathsf{id}^*, \mathsf{cor}) \leftarrow \mathcal{A}$
assert $|\mathsf{cor}| = T - 1$
$(\mathsf{ek}, \{\mathsf{dk}_i\}_{i \in N}) \leftarrow \mathsf{TIBE.Setup}(1^\kappa)$
$\mathcal{A}^{\mathsf{TIBE.ShareExtract}(\cdot,\mathsf{id}),\mathsf{id} \neq \mathsf{id}^*}(\{\mathsf{dk}_i\}_{\mathsf{cor}})$
$\mathsf{msg}^* \leftarrow \{0, 1\}^d$
$\mathsf{ct}^* = \mathsf{TIBE.Encrypt}(\mathsf{ek}, \mathsf{id}^*, \mathsf{msg}^*)$
$\mathsf{msg} \leftarrow \mathcal{A}^{\mathsf{TIBE.ShareExtract}(\cdot,\mathsf{id}),\mathsf{id} \neq \mathsf{id}^*}(\{\mathsf{dk}_i\}_{\mathsf{cor}}, \mathsf{ct}^*)$
return $(\mathsf{msg} = \mathsf{msg}^*)$
```

(d) TIBE Selective-ID One-Way Security

```
$\mathsf{Exp}^{\mathsf{Robust}}_{\mathcal{A},\mathsf{TIBE}}(1^\kappa)$
$(\mathsf{ek}, \{\mathsf{dk}_i\}_{i \in N}) \leftarrow \mathsf{TIBE.Setup}(1^\kappa)$
$(\mathsf{msg}, \mathsf{id}, \mathsf{rand}, \mathsf{ct}, \mathsf{act}, S) \leftarrow \mathcal{A}(\mathsf{ek}, \{\mathsf{dk}_i\}_{i \in [N]})$
assert $\mathsf{rand} \in \mathsf{RSpace}$
assert $\mathsf{ct} = \mathsf{TIBE.Encrypt}(\mathsf{ek}, \mathsf{id}, \mathsf{msg}; \mathsf{rand})$
for $i \in \mathsf{act}$
    assert $\mathsf{TIBE.ShareVerify}(\mathsf{ek}, \mathsf{act}, S, i)$
return $\mathsf{msg} \neq \mathsf{TIBE.Combine}(\mathsf{ct}, \mathsf{act}, S)$
```

(e) TIBE Extraction Share Robustness

Fig. 1: Security experiments

```
TKEM.Keygen($1^\kappa$)
    $(\{\mathsf{dk}_i\}_{i\in[N]}, \mathsf{ek}) \leftarrow$ TIBE.Setup($1^\kappa$)
    return $(\{\mathsf{dk}_i\}_{i\in[N]}, \mathsf{ek})$

TKEM.Encaps(ek)
    $\mathsf{msg} \leftarrow \{0,1\}^d$
    $\mathsf{rand} = G_{\mathsf{fo}}(\mathsf{msg})$
    $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SIG.Keygen}(1^\kappa)$
    $\mathsf{id} := \mathsf{vk}$
    $\mathsf{ct} =$ TIBE.Encrypt$(\mathsf{ek}, \mathsf{id}, \mathsf{msg}; \mathsf{rand})$
    $\mathsf{sig} \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}, \mathsf{ct})$
    $K = H_{\mathsf{fo}}(\mathsf{msg}||\mathsf{ct})$
    return $(\mathsf{ct}, \mathsf{vk}, \mathsf{sig}), K$

TKEM.ShareDecaps$_j(\mathsf{dk}_i, (\mathsf{ct}, \mathsf{vk}, \mathsf{sig})), j \leq r$
    assert $\mathsf{SIG.Verify}(\mathsf{vk}, \mathsf{ct}, \mathsf{sig}) = 1$
    $\mathsf{contrib}_{i,j} =$ TIBE.ShareExtract$_j(\mathsf{dk}_i, \mathsf{id} = \mathsf{vk})$
    return $\mathsf{contrib}_{i,j}$

TKEM.ShareVerify$(\mathsf{ek}, (\mathsf{ct}, \mathsf{vk}, \mathsf{sig}), \mathsf{act}, \{\mathsf{contrib}_{j,r}\}_{j,r}, i)$
    assert $\mathsf{SIG.Verify}(\mathsf{vk}, \mathsf{ct}, \mathsf{sig}) = 1$
    $b =$ TIBE.ShareVerify$(\mathsf{ek}, \mathsf{id} = \mathsf{vk}, \mathsf{act}, \{\mathsf{contrib}_{j,r}\}_{j,r}, i)$
    return b

TKEM.Combine$(\{\mathsf{contrib}_{i,r}\}_{i\in\mathsf{act}}, (\mathsf{ct}, \mathsf{vk}, \mathsf{sig}))$
    assert $\mathsf{SIG.Verify}(\mathsf{vk}, \mathsf{ct}, \mathsf{sig}) = 1$
    $\mathsf{msg} =$ TIBE.Combine$(\mathsf{ct}, \mathsf{vk}, \mathsf{act}, \{\mathsf{contrib}_{i,r}\}_i)$
    $\mathsf{rand} = G_{\mathsf{fo}}(\mathsf{msg})$
    assert $\mathsf{ct} =$ TIBE.Encrypt$(\mathsf{ek}, \mathsf{vk}, \mathsf{msg}; \mathsf{rand})$
    $K = H_{\mathsf{fo}}(\mathsf{msg}||\mathsf{ct})$
    return $K$
```

Fig. 2: The BCHK+ transform of [33].

Proof. Observe that for $\mathbf{z} = \mathbf{u}^T\mathbf{v}$, all the coefficients of $\mathbf{z}$ are of the form $\mathbf{z}_i = \langle \tilde{\mathbf{u}}_i, \mathbf{v}\rangle$ where $\tilde{\mathbf{u}}_i$ is a permutation of $\mathbf{u}$ where some coefficients are multiplied by -1 (this comes directly from the formula of the coefficient of degree $k < d$ of a product of polynomials in $\mathcal{R}$). From Lemma 7, we have:

$$\mathbb{P}\left[|\langle \tilde{\mathbf{u}}_i, \mathbf{v}\rangle| > r\right] \leq 2e^{-\frac{r^2}{2\|\mathbf{u}\|^2\sigma^2}}$$

By union bound and since $\forall i, \|\tilde{\mathbf{u}}_i\| = \|\mathbf{u}\|$:

$$\mathbb{P}\left[\left\|\mathbf{u}^T\mathbf{v}\right\|_\infty > r\right] \leq 2dke^{-\frac{r^2}{2\|\mathbf{u}\|^2\sigma^2}}$$

Solving for r gives the desired result. □

Corollary 1. *Let $\mathbf{a} \in \mathbb{Z}^d$ be an arbitrary vector. For $\mathbf{x} \leftarrow D_{\mathcal{R}^d,\sigma}$, the following holds with probability at least $1 - 2^{-\kappa}$:*

$$\|\mathbf{a}+\mathbf{x}\|^2 \leq \|\mathbf{a}\|^2 + 2\sqrt{2(\kappa+1)\log 2}\,\|\mathbf{a}\|\,\sigma + \|\mathbf{x}\|^2 \quad (4)$$
$$\sim \|\mathbf{a}\|^2 + \|\mathbf{x}\|^2 \qquad \textit{when } \kappa = o(n), \quad (5)$$

Proof. We start from the identity $\|\mathbf{a}+\mathbf{x}\|^2 = \|\mathbf{a}\|^2 + \|\mathbf{x}\|^2 + 2\langle \mathbf{a}, \mathbf{x}\rangle$, then apply Lemma 8 on $\|\mathbf{x}\|$. □

```
$\mathcal{P}_\beta(\mathbf{s})$
$u \leftarrow \mathcal{R}_q$
$(c_0, c_1) := \mathsf{Decomp}_\beta(u)$
$\mathbf{p} \leftarrow \mathcal{D}_{\mathcal{R}^{1+m},\sigma_\mathbf{p}}$
$\mathbf{z} = \mathbf{s} \cdot c_0 + \mathbf{p}$
return $\{(\mathbf{c} = (c_0, c_1), \mathbf{z})\}$
```

Fig. 3: Hint distribution for Hint-RLWE

2.4 Hint Ring Learning With Errors

We rely on the Hint-RLWE assumption, introduced by Kim et al. [30]. More specifically, we use a variant by Esgin et al. [22], recalled in Definition 11. While the distribution of the $\mathbf{c}_i$'s in Definition 11 may seem artificial, it is a natural byproduct of the signing procedure used in [22], which we re-use in this work.

Definition 11 (Hint-RLWE). *Let $m > 0$, $Q_1, Q_2 \geq 0$ and q be integers, and $\sigma_\mathbf{s}, \sigma_1, \sigma_2 > 0$. Let $\mathbf{a} \leftarrow \mathcal{R}_q^m$. The $\mathsf{Hint\text{-}RLWE}^{Q_1,Q_2,\sigma_1,\sigma_2,\beta}_{\mathcal{R},q,m,\sigma_\mathbf{s}}$ problem requires the Adversary to distinguish distributions $\mathcal{D}_{\mathsf{real}}, \mathcal{D}_{\mathsf{ideal}}$ where $\mathbf{s} \leftarrow \mathcal{D}_{\mathcal{R}^{m+1},\sigma_\mathbf{s}}$ and:*

$$\mathcal{D}_{\mathsf{real}} = \left\{ \left(\mathbf{a}, \mathbf{b}, \{\mathbf{c}_i, \mathbf{z}_i\}_{i\in[Q_1+Q_2]}\right) \middle| \mathbf{b} = [\mathbf{a}\ \mathbf{I}_m] \cdot \mathbf{s}, \begin{array}{c} \{\mathbf{c}_i, \mathbf{z}_i\}_{i\in[Q_1]} \leftarrow \mathcal{P}_\beta(\mathbf{s}) \\ \{\mathbf{c}_i, \mathbf{z}_i\}_{i\in(Q_1,Q_2]} \leftarrow \mathcal{P}(\mathbf{s}) \end{array} \right\}$$

$$\mathcal{D}_{\mathsf{ideal}} = \left\{ \left(\mathbf{a}, \mathbf{b}, \{\mathbf{c}_i, \mathbf{z}_i\}_{i\in[Q_1+Q_2]}\right) \middle| \mathbf{b} \leftarrow \mathcal{R}_q^m \cdot \mathbf{s}, \begin{array}{c} \{\mathbf{c}_i, \mathbf{z}_i\}_{i\in[Q_1]} \leftarrow \mathcal{P}_\beta(\mathbf{s}) \\ \{\mathbf{c}_i, \mathbf{z}_i\}_{i\in(Q_1,Q_2]} \leftarrow \mathcal{P}(\mathbf{s}) \end{array} \right\}$$

where the distribution $\mathcal{P}_\beta(\mathbf{s})$ is described in Fig. 3 and $\mathcal{P}(\mathbf{s}) = \mathbf{s} + \mathcal{D}_{\mathcal{R}^{m+1},\sigma_2}$.

When the adversary receives no hints ($Q_1 = Q_2 = 0$) in Definition 11, we recover the Ring-LWE problem $\mathsf{RLWE}_{\mathcal{R},q,m,\sigma_\mathbf{s}}$. In [30,22] the standard hardness of Hint-RLWE for $Q_1 \geq 1$, and $Q_2 = 0$ is proven. We give a straightforward adaptation of the reduction in Lemma 10. Note that there is a factor 2 in the formula for $\frac{1}{\sigma_0^2}$, which seems to be an artefact of the proof, as is clearly the case when taking $Q = 0$.

Lemma 10 (Theorem 1 and Lemma 2 from [22]). *Assume $Q_1, Q_2 = \omega(\kappa\, d \log d)^2$, and let $B_{\mathsf{hint}} = \frac{Q\,M^2}{12}$, where $M = 2\left\lceil \frac{q-1}{2\beta} \right\rceil + 1$. Let $\sigma_0, \sigma_\mathbf{s}, \sigma_1, \sigma_2 > 0$ such that $\frac{1}{\sigma_0^2} = 2\left(\frac{1}{\sigma_\mathbf{s}^2} + \frac{B_{\mathsf{hint}}(1+o(1))}{\sigma_1^2} + \frac{Q_2}{\sigma_2^2}\right)$. If $\sigma_0 \geq \sqrt{2}\eta_\epsilon(\mathbb{Z}^d)$ for $0 < \epsilon \leq 1/2$, where $\eta_\epsilon(\mathbb{Z}^d)$ is the smoothing parameter of $\mathbb{Z}^d$, then there is an efficient reduction from $\mathsf{RLWE}_{\mathcal{R},q,m,\sigma_0}$ to $\mathsf{Hint\text{-}RLWE}^{Q_1,Q_2,\sigma_1,\sigma_2,\beta}_{\mathcal{R},q,m,\sigma_\mathbf{s}}$ that reduces the advantage by at most 4ϵ.*

2.5 Short Secret Sharing

We recall the Vandermonde secret sharing (VSS) [9], a lattice-friendly adaptation of a secret sharing by Desmedt et al. [19].

Dictionaries. We recall that a dictionary, or associative array, is a collection of ordered (key, value) pairs with unique keys. We note $\{:\}$ the empty dictionary, and $\mathsf{Dict}[k]$ the value of Dict corresponding to the key k. We note $\mathsf{Dict}_1 \cup \mathsf{Dict}_2$ the union of two dictionaries. When a key k is present in both Dict_1 and Dict_2 and $\mathsf{Dict}_1[k], \mathsf{Dict}_2[k]$ are either both sets or both dictionaries, the new value $(\mathsf{Dict}_1 \cup \mathsf{Dict}_2)[k]$ sets as $\mathsf{Dict}_1[k] \cup \mathsf{Dict}_2[k]$.

Algorithm 2 $\mathsf{VandShare}(\mathbf{s}, \mathcal{P}, T, \mathsf{idx} = \varnothing) \to \mathsf{Dict}$

```
$N = |\mathcal{P}|$
if $T = 1$ then
    return $\mathsf{Dict} := \{\mathsf{user} : \{\mathsf{idx} : \mathbf{s}\} \mid \mathsf{user} \in \mathcal{P}\}$
else
    $\mathsf{Dict} = \{\mathsf{user} : \{:\} \mid \mathsf{user} \in \mathcal{P}\}$, $b = \lfloor N/2 \rfloor$
    Parse $\mathcal{P} = \mathcal{P}_L \sqcup \mathcal{P}_R$, with $\mathcal{P}_L$ the $b$ smallest elements of $\mathcal{P}$
    for $k = \max(0, T - N + b), \ldots, \min(b, T)$ do
        $\mathsf{idx}_L := \mathsf{idx} \parallel$ ":L:" $\parallel k$,  $\mathsf{idx}_R := \mathsf{idx} \parallel$ ":R:" $\parallel (T - k)$
        if $k = 0$ then
            $\mathsf{Dict} := \mathsf{Dict} \cup \mathsf{VandShare}(\mathbf{s}, \mathcal{P}_R, T, \mathsf{idx}_R)$
        else if $k = T$ then
            $\mathsf{Dict} := \mathsf{Dict} \cup \mathsf{VandShare}(\mathbf{s}, \mathcal{P}_L, T, \mathsf{idx}_L)$
        else
            $\mathbf{s}_0 \leftarrow \chi$                                ▷ This guarantees short shares
            $\mathbf{s}_1 := (\mathbf{s} - \mathbf{s}_0) \bmod q$
            $\mathsf{Dict}_L := \mathsf{VandShare}(\mathbf{s}_0, \mathcal{P}_L, k, \mathsf{idx}_L)$
            $\mathsf{Dict}_R := \mathsf{VandShare}(\mathbf{s}_1, \mathcal{P}_R, T - k, \mathsf{idx}_R)$
            $\mathsf{Dict} := \mathsf{Dict} \cup \mathsf{Dict}_L \cup \mathsf{Dict}_R$
    return $\mathsf{Dict}$                        ▷ The values of Dict are also dictionaries.
```

Lemma 11. *Let $\varepsilon \in (0, 1/2)$, $m, N, T \geq 1$, $\sigma \geq \sqrt{2}\eta_\epsilon(\mathcal{R}^m)$, $\mathbf{s} \leftarrow D_{\mathcal{R}^m,\sigma}$, and $\mathsf{Shares} \leftarrow \mathsf{VandShare}(\mathbf{s}, [N], T)$. In Algorithm 2, let $\chi = D_{\mathcal{R}^m,\sigma}$. For any user $i \in [N]$ and any share $\mathbf{s}^{(\mathsf{idx})} := \mathsf{Shares}[i][\mathsf{idx}]$, we have $\mathrm{SD}(\mathbf{s}^{(\mathsf{idx})}, D_{\mathcal{R}^m,\sqrt{k}\cdot\sigma}) \leq (k-1) \cdot 8\varepsilon$ for a value $k \in \mathbb{N}, 1 \leq k \leq \log N + 1$ determined by Algorithm 2.*

3 Improved TIBE Construction From Lattices

Let $\mathsf{H}_{\mathsf{cmt}}(\cdot) : \mathcal{R}_q \to \{0,1\}^{2\kappa}$ and $\mathsf{H}_{\mathsf{id}}(\cdot) : S_{\mathsf{id}} \to \mathcal{R}_q$ be hash functions modelled as random oracles.

Public parameters. These include $(\mathbf{A}, t, N, T)$, where N is the overall number of parties in the protocol and T is the threshold on the number of parties required for decryption.

1. The public matrix $\mathbf{A}$ is a matrix with trapdoor *à la* Eagle [43] or Plover [22].The parameter $\beta \in \mathbb{Z}$ can be set arbitrarily between 2 and $q/2$ and

Algorithm 3 $\mathsf{VandRecover}(\mathcal{P}, \mathsf{act}, \mathsf{idx} = \varnothing) \to \mathsf{Indices}$

```
$N = |\mathcal{P}|, T = |\mathsf{act}|$
if $T = 1$ then
    return $\mathsf{Indices} := \{\mathsf{user} : \mathsf{idx} \mid \mathsf{user} \in \mathcal{P}\}$
else
    $b = \lfloor N/2 \rfloor$
    Parse $\mathcal{P} = \mathcal{P}_L \sqcup \mathcal{P}_R$, with $\mathcal{P}_L$ the $b$ smallest elements of $\mathcal{P}$
    $\mathsf{act}_L = \mathsf{act} \cap \mathcal{P}_L, \mathsf{act}_R = \mathsf{act} \cap \mathcal{P}_R, k = |\mathsf{act}_L|$
    $\mathsf{idx}_L := \mathsf{idx} \parallel \texttt{":L:"} \parallel k,\ \ \mathsf{idx}_R := \mathsf{idx} \parallel \texttt{":R:"} \parallel (T - k)$
    if $k = 0$ then
        return $\mathsf{VandRecover}(\mathcal{P}_R, \mathsf{act}_R, \mathsf{idx}_R)$
    else if $k = T$ then
        return $\mathsf{VandRecover}(\mathcal{P}_L, \mathsf{act}_L, \mathsf{idx}_L)$
    else
        $\mathsf{Indices}_L := \mathsf{VandRecover}(\mathcal{P}_L, \mathsf{act}_L, \mathsf{idx}_L)$
        $\mathsf{Indices}_R := \mathsf{VandRecover}(\mathcal{P}_R, \mathsf{act}_R, \mathsf{idx}_R)$
        return $\mathsf{Indices} := \mathsf{Indices}_L \sqcup \mathsf{Indices}_R$
```

impacts the quality of the trapdoor.

$$\mathbf{A} = \begin{bmatrix} 1 & a & b \end{bmatrix}, \quad \text{where} \quad \begin{array}{l} a \leftarrow \mathcal{R}_q, \quad s, s' \leftarrow \mathcal{D}_{\mathcal{R},\sigma_\mathbf{s}}, \\ b = \beta - (a \cdot s + s') \text{ for some } \beta \in \mathbb{Z} \end{array} \tag{6}$$

2. $t \leftarrow \mathcal{R}_q$ is used as a target for preimage sampling.

Encryption procedure. In the encryption procedure, the encrypter constructs a public key $\begin{bmatrix} a & b & \mathsf{H}_{\mathsf{id}}(\mathsf{id}) & t \end{bmatrix}$ for the identity id. Then they generate a Lindner-Peikert [34] style ciphertext $(\mathbf{u}, v)$ for this public key and the message msg. This is described in Algorithm 9.

Decryption procedure. In the decryption procedure, the parties jointly compute a vector $\mathbf{z} = \begin{bmatrix} \mathbf{z}' & x \end{bmatrix}$ such that $\begin{bmatrix} a & b & \mathsf{H}_{\mathsf{id}}(\mathsf{id}) \end{bmatrix} \cdot \mathbf{z} \approx r$, where r is the encryption randomness. They then use $\mathbf{z}$ to perform a Lindner-Peikert style decryption procedure, by computing $v - \mathbf{u}^T \cdot \mathbf{z}'$ and decoding it to recover the message. This is described in Fig. 5.

The TIBE is formally described in Figs. 4 and 5. For clarity of reading, we assume that the state of party i contains (i) the secret key share of i, (ii) all contributions of all parties up to the current round, (iii) the values privately generated by i during previous rounds. In order to avoid ROS-style attacks, it is also important to ensure that the participants' views are consistent, by maintaining their internal states and checking it against received contributions, see for example [28, Figure 3]. For clarity of reading, we omit the consistency checks from our description and refer to [28] for more details.

Algorithm 4 $\mathsf{TIBE.KG}(1^\kappa) \rightarrow \mathsf{ek}, \mathsf{dk}$

```
1: $t, a \leftarrow \mathcal{R}_q \times \mathcal{R}_q$
2: $(s, s') \leftarrow \mathcal{D}_{\mathcal{R}^2,\sigma_\mathbf{s}}$
3: $b = \beta - (a \cdot s + s')$
4: $\mathbf{s} = \begin{bmatrix} s' & s & 1 & 0 \end{bmatrix}^T$        ▷ $\mathbf{s} \in \mathcal{R}^4$, and $\forall \mathsf{id}$, $\begin{bmatrix} 1 & a & b & H(\mathsf{id}) \end{bmatrix} \cdot \mathbf{s} = \beta$
5: return $\mathsf{ek} := (a, b, t), \mathsf{dk} := (s, s')$
```

Algorithm 5 $\mathsf{TIBE.Share}(\mathsf{ek}, \mathsf{dk}) \rightarrow \mathsf{aux}, \{\mathsf{dk}^{(i)}\}_{i\in[N]}$

```
1: $\mathsf{Shares} \leftarrow \mathsf{VandShare}((s, s'), [N], T, \mathsf{idx} = \{\})$
2: $\mathsf{pek} := \mathsf{Dict}\{\}$        ▷ pek lists all partial encryption keys (initially empty)
3: for $i \in [N]$ do
4:     $\mathsf{dk}^{(i)} := \mathsf{Shares}[i]$ ▷ Each user $i$ has a dictionary $\{\mathsf{idx} : \mathbf{s}^{(\mathsf{idx})}\}$of private keys.
5:     for $\mathsf{idx} \in \mathsf{Shares}[i]$ do        ▷ Compute all partial public keys
6:         $\mathsf{pek}[\mathsf{idx}] := b^{(\mathsf{idx})} = a \cdot s^{(\mathsf{idx})} + s'^{(\mathsf{idx})}$
7: $\mathsf{ek} := (a, b, t, \mathsf{pek})$
8: return $\mathsf{aux} := \mathsf{pek}, \{\mathsf{dk}^{(i)}\}_{i\in[N]}$
```

Algorithm 6 $\mathsf{TIBE.Setup}(1^\kappa) \rightarrow \mathsf{ek}, \{\mathsf{dk}^{(i)}\}_{i\in[N]}$

```
1: $(\mathsf{ek}_0, \mathsf{dk}) \leftarrow \mathsf{TIBE.KG}(1^\kappa)$
2: $(\mathsf{ek}_1, \{\mathsf{dk}^{(i)}\}_{i\in[N]}) \leftarrow \mathsf{TIBE.Share}(\mathsf{ek}, \mathsf{dk})$
3: return $\mathsf{ek} := (\mathsf{ek}_0, \mathsf{ek}_1), \{\mathsf{dk}^{(i)}\}_{i\in[N]}$
```

Algorithm 7 $\mathsf{Encode}(\mathsf{msg} = (\alpha_0, \dots, \alpha_{d-1})) \rightarrow M \in \mathcal{R}_q$

```
1: return $M = \sum_{i\in[d]} \alpha_i \cdot \lfloor q/2 \rceil \cdot X^i$
```

Algorithm 8 $\mathsf{Decode}(M = \sum_{i\in[d]} \alpha_i \cdot X^i) \rightarrow \mathsf{msg} \in \{0,1\}^d/\bot$

```
1: return $\mathsf{msg} = \sum_{i\in[d]} \lfloor \alpha_i \rceil_q X^i$        ▷
```

Algorithm 9 $\mathsf{TIBE.Encrypt}(\mathsf{msg} \in \{0,1\}^d, \mathsf{id}, \mathsf{ek}) \rightarrow \mathsf{ct}$

```
1: $(r, \mathbf{e}, e') \leftarrow D_{\mathcal{R},\sigma_\mathbf{r}} \times D_{\mathcal{R}^3,\sigma_\mathbf{r}} \times D_{\mathcal{R},\sigma_\mathbf{r}}$
2: $\begin{bmatrix} \mathbf{u}^T & v \end{bmatrix} := r \cdot \begin{bmatrix} a & b & \mathsf{H_{id}}(\mathsf{id}) & t \end{bmatrix} + \begin{bmatrix} \mathbf{e}^T & e' \end{bmatrix} + \begin{bmatrix} \mathbf{0}^{1\times 3} & \mathsf{Encode}(\mathsf{msg}) \end{bmatrix}$        ▷ $\begin{bmatrix} \mathbf{u} \\ v \end{bmatrix} \in \mathcal{R}_q^4$
3: return $\mathsf{ct} := (\mathbf{u}, v)$
```

Fig. 4: TIBE algorithms for key generation, encryption and message encoding.

Algorithm 10 $\mathsf{TIBE.ShareExtract}_0(\mathsf{id}, \mathsf{act}, \mathsf{dk}^{(i)}, \mathsf{state}^{(i)}) \to \mathsf{contrib}_0^{(i)}/\perp$

```
1: $\mathbf{p}^{(i)} \leftarrow D_{\mathcal{R}^2,\sigma_\mathbf{p}} \times \{0\} \times D_{\mathcal{R},\sigma'_\mathbf{p}}$
2: $w^{(i)} = \begin{bmatrix}1 & a & b & \mathsf{H}_{\mathsf{id}}(\mathsf{id})\end{bmatrix} \cdot \mathbf{p}^{(i)}$
3: $\mathsf{cmt}^{(i)} = \mathsf{H}_{\mathsf{cmt}}(w^{(i)})$
4: **return** $\mathsf{contrib}_0^{(i)} := \mathsf{cmt}^{(i)}$
```

Algorithm 11 $\mathsf{TIBE.ShareExtract}_1(\mathsf{id}, \mathsf{act}, \mathsf{dk}^{(i)}, \mathsf{state}^{(i)}) \to \mathsf{contrib}_1^{(i)}/\perp$

```
1: **return** $\mathsf{contrib}_1^{(i)} := w^{(i)}$
```

Algorithm 12 $\mathsf{TIBE.ShareExtract}_2(\mathsf{id}, \mathsf{act}, \mathsf{dk}^{(i)}, \mathsf{state}^{(i)}) \to \mathsf{contrib}_2^{(i)}/\perp$

```
1: **assert**$\{\forall j \in \mathsf{act} \setminus \{i\} : \mathsf{H}_{\mathsf{cmt}}(w^{(j)}) = \mathsf{cmt}^{(j)}\}$
2: $w := \sum_{j \in \mathsf{act}} w^{(j)}$
3: $(c_0, c_1) := \mathsf{Decomp}_\beta(t - w)$        ▷ $c_0 \cdot \beta + c_1 = t - w$
4: $\mathsf{Indices} := \mathsf{VandRecover}([N], \mathsf{act})$        ▷ $i$ is assigned one index $\mathsf{Indices}[i]$
5: $s^{(i)}, s'^{(i)} := \mathsf{dk}^{(i)}[\mathsf{Indices}[i]]$
6: $\mathbf{s}^{(i)} := \begin{bmatrix}s'^{(i)} & s^{(i)} & 0 & 0\end{bmatrix}^T$        ▷ $\mathbf{s}^{(i)} \in \mathcal{R}^4$ is the partial signing key assigned to $i$
7: $\mathbf{z}^{(i)} = \mathbf{p}^{(i)} + c_0 \cdot \mathbf{s}^{(i)}$
8: **return** $\mathsf{contrib}_2^{(i)} := \mathbf{z}^{(i)}$
```

Algorithm 13 $\mathsf{TIBE.ShareVerify}(\mathsf{ek}, \mathsf{id}, \mathsf{act}, \{(w^{(j)}, \mathbf{z}^{(j)})\}_{j \in \mathsf{act}}, i) \to \{1, \perp\}$

```
1: $w := \sum_{j \in \mathsf{act}} w^{(j)}$
2: $(c_0, c_1) := \mathsf{Decomp}_\beta(t - w)$
3: $\mathsf{Indices} := \mathsf{VandRecover}([N], \mathsf{act})$        ▷ User $i$ is assigned one index $\mathsf{Indices}[i]$
4: $b^{(i)} := \mathsf{pek}[\mathsf{Indices}[i]]$
5: **assert**$\{\{\begin{bmatrix}1 & a & b & \mathsf{H}_{\mathsf{id}}(\mathsf{id})\end{bmatrix} \cdot \mathbf{z}^{(i)} = w^{(i)} + c_0 \cdot b^{(i)}\}\}$        ▷ Equality
6: **assert**$\{\{\left\|\mathbf{z}^{(i)}\right\| \leq B_{\mathsf{ind}}\}\}$        ▷ Shortness
7: **return** 1        ▷ Accept if both asserts pass
```

Algorithm 14 $\mathsf{TIBE.Combine}(\mathsf{ct}, \mathsf{ek}, \mathsf{id}, \mathsf{act}, \{(w^{(i)}, \mathbf{z}^{(i)})\}_{i \in \mathsf{act}}) \to \mathsf{msg}/\perp$

```
1: **for** $i \in \mathsf{act}$ **do**
2:     **assert**$\{\mathsf{TIBE.ShareVerify}(\mathsf{ek}, \mathsf{id}, \mathsf{act}, \{(w^{(j)}, \mathbf{z}^{(j)})\}_{j \in \mathsf{act}}, i)\}$
3: $\mathbf{z} := \left(\sum_{i \in \mathsf{act}} \mathbf{z}^{(i)}\right) + \begin{bmatrix}c_1 & 0 & c_0 & 0\end{bmatrix}^T$
4: $\mathbf{z}' := \begin{bmatrix}z_1 & z_2 & z_3\end{bmatrix}^T$        ▷ $\mathbf{z}'$ drops the first coefficient of $\mathbf{z}$
5: **return** $\mathsf{msg} := \mathsf{Decode}\left(v - \mathbf{u}^T \cdot \mathbf{z}'\right)$
```

Fig. 5: TIBE algorithms for threshold decryption.

4 TIBE Correctness and Share Robustness

We now study the correctness and share robustness of the TIBE described in Figs. 4 and 5. We start be providing useful equations holding by construction. Let us note $\mathbf{p}^{(i)} = \left[p_0^{(i)}\ p_1^{(i)}\ 0\ p_3^{(i)}\right]^T$, $\mathbf{p} = \sum_i \mathbf{p}^{(i)}$ and $p_j = \sum_{i \in \mathsf{act}} p_j^{(i)}$ for $j \in [4]$. Recall that $b = \beta - \sum_{i \in \mathsf{act}} b$, $\mathbf{s} = \left[\sum_i s^{(i)}\ \sum_i s'^{(i)}\ 1\ 0\right]^{\mathsf{T}}$, $w = \sum_i w^{(i)}$ and $\mathbf{z} = \sum_i \mathbf{z}^{(i)}$. Remember that by construction:

$$\forall\, \mathsf{id},\ \left[1\ a\ b\ \mathsf{H}_{\mathsf{id}}(\mathsf{id})\right] \cdot \mathbf{s} = \beta \qquad \text{(Algorithm 4, Line 4)} \tag{7}$$

$$\forall\, \mathsf{id}, \forall i \in \mathsf{act},\ \left[1\ a\ b\ \mathsf{H}_{\mathsf{id}}(\mathsf{id})\right] \cdot \mathbf{s}^{(i)} = b^{(i)} \qquad \text{(Algorithm 5, Line 6)} \tag{8}$$

$$\forall\, \mathsf{id}, \forall i \in \mathsf{act},\ \left[1\ a\ b\ \mathsf{H}_{\mathsf{id}}(\mathsf{id})\right] \cdot \mathbf{p}^{(i)} = w^{(i)} \qquad \text{(Algorithm 10, Line 2)} \tag{9}$$

$$c_0 \cdot \beta + c_1 = t - w \qquad \text{(Algorithm 12, Line 3)} \tag{10}$$

4.1 Extraction Share Correctness

We start by proving correctness of the key extraction shares, cf. Definition 6.

Lemma 12. *Let δ' be the target correctness probability of share verification in Algorithm 13 . Given* TIBE *as described in Figs. 4 and 5 with $\tau \geq 1 + \sqrt{\frac{4 \log \delta'}{3d}}$, $\sigma_{\mathbf{s}} \geq \eta_\epsilon(\mathcal{R}^2)$ and norm bound B_{ind} given as*

$$B_{\mathsf{ind}} = \tau \left(\sqrt{d\,(2\,\sigma_{\mathbf{p}}^2 + {\sigma'_{\mathbf{p}}}^2)} + \frac{d\,q}{2\beta} \cdot \sqrt{2\,d\,(\log N + 1)}\,\sigma_{\mathbf{s}} \right),$$

where d is the ring degree, ($\sigma_{\mathbf{p}}$,$\sigma'_{\mathbf{p}}$) (resp. $\sigma_{\mathbf{s}}$) is the Gaussian width of the masks (resp. secret), q is the modulus, β is the decomposition parameter and N is the number of parties. Then, TIBE *has $(\log N + 1)\epsilon + \delta' \simeq \delta'$ extraction share correctness.*

Proof. Note that TIBE.ShareVerify described in Algorithm 13 outputs 1 if both **assert** checks pass. First, for $i \in \mathsf{act}$, let us write $\mathbf{z}^{(i)}$ and $\mathbf{z}$ explicitly:

$$\mathbf{z}^{(i)} = \begin{bmatrix} p_0^{(i)} + c_0 \cdot s'^{(i)} \\ p_1^{(i)} + c_0 \cdot s^{(i)} \\ 0 \\ p_3^{(i)} \end{bmatrix},\ \mathbf{z} = \left(\sum_{i \in \mathsf{act}} \mathbf{z}^{(i)}\right) + \begin{bmatrix} c_1 \\ 0 \\ c_0 \\ 0 \end{bmatrix} = \begin{bmatrix} p_0 + c_0 \cdot s' + c_1 \\ p_1 + c_0 \cdot s \\ c_0 \\ p_3 \end{bmatrix}. \tag{11}$$

Recall that the third coefficient of $\mathbf{p}^{(i)}$ is 0. It is immediate from Eqs. (8) to (11) that:

$$\begin{aligned} \left[1\ a\ b\ \mathsf{H}_{\mathsf{id}}(\mathsf{id})\right] \cdot \mathbf{z}^{(i)} &= \left[1\ a\ b\ \mathsf{H}_{\mathsf{id}}(\mathsf{id})\right] \left(\mathbf{p}^{(i)} + c_0 \cdot \mathbf{s}^{(i)}\right) \\ &= \mathbf{w}^{(i)} + c_0 \cdot b^{(i)}. \end{aligned} \tag{12}$$

Therefore, the first **assert** checking equality in Algorithm 13 always passes.

Next, we study the shortness of $\mathbf{z}^{(i)}$. For an honestly generated $\mathbf{z}^{(i)}$, by applying Minkowski's inequality in Eq. (13), then once again in Eq. (14), and finally Lemmas 8 and 11 in Eq. (15), except with probability $\delta' := (\log N + 1)\epsilon + 2^{-\kappa}$ it holds:

$$\left\| \mathbf{z}^{(i)} \right\| \leq \left\| \mathbf{p}^{(i)} \right\| + \left\| c_0 \cdot \mathbf{s}^{(i)} \right\| \tag{13}$$

$$\leq \left\| \mathbf{p}^{(i)} \right\| + \left\| c_0 \right\|_1 \cdot \left\| \mathbf{s}^{(i)} \right\| \tag{14}$$

$$\leq \tau \sqrt{d\,(2\,\sigma_{\mathbf{p}}^2 + {\sigma'_{\mathbf{p}}}^2)} + \frac{d\,q}{2\beta} \cdot \tau \sqrt{2\,d}\,(\log N + 1)\,\sigma_{\mathbf{s}}, \tag{15}$$

where $\tau \geq 1 + \sqrt{\frac{4 \log \delta'}{3d}}$ is as defined in Lemma 8. Therefore setting:

$$B_{\mathsf{ind}} = \tau \left(\sqrt{d\,(2\,\sigma_{\mathbf{p}}^2 + {\sigma'_{\mathbf{p}}}^2)} + \frac{d\,q}{2\beta} \cdot \sqrt{2\,d}\,(\log N + 1)\,\sigma_{\mathbf{s}} \right) \tag{16}$$

guarantees that the shortness **assert** in Alg. 14 passes except with probability δ'.
□

4.2 Decryption Correctness

We now study the decryption correctness of our TIBE, cf. Definition 5.

Lemma 13. *Let δ' be the target correctness probability of share verification in Algorithm 13 . Given* TIBE *as described in Figs. 4 and 5 with $\tau \geq 1 + \sqrt{\frac{4(\log \delta' + \log 2)}{2d}}$, $\min(\sigma_{\mathbf{p}}, \sigma_{\mathbf{p}'}) \geq \sqrt{1 + 1/(T-1)} \cdot \eta_\varepsilon(\mathcal{R})$ and norm bounds B_h and B_r given as*

$$B_h = \tau \sqrt{T\,d\,(2\,\sigma_{\mathbf{p}}^2 + {\sigma'_{\mathbf{p}}}^2)} + \frac{d\,q}{2\,\beta} \left(\tau \sqrt{2\,d}\,\sigma_{\mathbf{s}} + 1 \right) + B_1,$$
$$B_r = \sqrt{2(\log \delta' + 1 + \log(8d))}\,\sigma_{\mathbf{r}},$$

where T is the number of parties, d is the ring degree, $\sigma_{\mathbf{p}}$ (resp. $\sigma_{\mathbf{s}}$ and $\sigma_{\mathbf{r}}$) is the Gaussian width of the masks (resp. secret and encryption randomness), q is the modulus, β is the decomposition parameter and B_1 is a norm bound on c_1. Furthermore assume that TIBE *is δ' extraction share correct. If $(B_h + 1) \cdot B_r \leq \lfloor q/4 \rfloor$, then,* TIBE *has $\delta := T(\delta' + 1) + 2 \cdot 2^{-\kappa} + 8\varepsilon \cdot (T-1)$ decryption correctness.*

We highlight that B_h is mainly impacted by $\sqrt{T}\sigma_{\mathbf{p}}$.

Proof. First we characterize the global preimage $\mathbf{z}$. By Lemma 12, the check in Algorithm 14, Line 2 passes for a given $i \in \mathsf{act}$ with probability δ' and by a union bound we arrive at the next line with probability $\geq 1 - T\delta'$. Hence, we assume in the following that equality and shortness checks have passed for all $i \in \mathsf{act}$.

We first aggregate Eq. (12) over all $i \in \mathsf{act}$, then observe that $\sum_i b^{(i)} = \beta - b$, and finally use Eq. (10), which gives:

$$\begin{aligned}\begin{bmatrix}1 & a & b & \mathsf{H}_{\mathsf{id}}(\mathsf{id})\end{bmatrix} \cdot \mathbf{z} &= \begin{bmatrix}1 & a & b & \mathsf{H}_{\mathsf{id}}(\mathsf{id})\end{bmatrix} \left(\sum \mathbf{z}^{(i)} + \begin{bmatrix}c_1 & 0 & c_0 & 0\end{bmatrix}^T\right) \\ &= \sum_i \left(w^{(i)} + c_0 \cdot b^{(i)}\right) + c_1 + c_0 \cdot b \\ &= w + c_0 \sum_i b^{(i)} + c_1 + c_0 \cdot b \\ &= w + c_0 \cdot \beta - c_0 \cdot b + c_1 + c_0 \cdot b = t \end{aligned} \tag{17}$$

Eq. (17) is essentially the same as Eq. (12), but for the global trapdoor instead of a partial trapdoor. It states that $\mathbf{z}$ is a valid preimage for the matrix $\begin{bmatrix}1 & a & b & \mathsf{H}_{\mathsf{id}}(\mathsf{id})\end{bmatrix}$ and the target t.

We now explain why the scheme decrypts correctly. Recall that the honest decryption procedure computes:

$$v - \mathbf{u} \cdot \mathbf{z}' = (r \cdot t + e' + \mathsf{Encode}(\mathsf{msg})) - (r \cdot \begin{bmatrix}a & b & \mathsf{H}_{\mathsf{id}}(\mathsf{id})\end{bmatrix} + \mathbf{e}) \cdot \mathbf{z}' \tag{18}$$

$$= \mathsf{Encode}(\mathsf{msg}) + r \cdot t + e' - r(t - z_0) - \mathbf{e} \cdot \mathbf{z}' \tag{19}$$

$$= \mathsf{Encode}(\mathsf{msg}) + (e' + \begin{bmatrix}r & -\mathbf{e}\end{bmatrix} \cdot \mathbf{z}). \tag{20}$$

As long as it holds for the residual noise that $\left\|e' + \begin{bmatrix}r & -\mathbf{e}\end{bmatrix} \cdot \mathbf{z}\right\|_\infty \leq \lfloor q/4 \rfloor$, decryption will be correct. In practice, the impact of e' in the residual noise is negligible, the dominating term being $\begin{bmatrix}r & -\mathbf{e}\end{bmatrix} \cdot \mathbf{z}$.

Bounding $\|\mathbf{z}\|$. By Eq. (11), after honest execution of the key share extraction, it holds $\mathbf{z} = \mathbf{p} + c_0 \mathbf{s} + \begin{bmatrix}c_1 & 0 & 0 & 0\end{bmatrix}$. By Minkowski's inequality, except with probability $2 \cdot 2^{-(\kappa+2)}$, it holds

$$\|\mathbf{z}\| \leq \left\|\mathbf{p} + \begin{bmatrix}0 & 0 & c_1 & 0\end{bmatrix}^{\mathrm{T}}\right\| + \|c_0\|_1 \cdot \|\mathbf{s}\| \tag{21}$$

$$\leq \|\mathbf{p}\| + \|c_1\| + \|c_0\|_1 \cdot \|\mathbf{s}\| \tag{22}$$

$$\leq \tau \sqrt{T\, d\, (2\, \sigma_{\mathbf{p}}^2 + {\sigma'_{\mathbf{p}}}^2)} + \frac{d\, q}{2\, \beta} \left(\tau \sqrt{2\, d}\, \sigma_{\mathbf{s}} + 1\right) + B_1 := B_z, \tag{23}$$

where B_1 is a bound on the norm of c_1. Here, we used that both $\mathbf{p}$ is close to discrete Gaussian by Lemma 5 and $\mathbf{s}$ is discrete Gaussians and their norms bounded using concentration bounds (Lemma 8). In practice, $\|\mathbf{p}\|$ dominates $\|c_0\|_1 \cdot \|\mathbf{s}\|$, but does not necessarily dominate $\|c_1\|$.

Bounding $\left\|\begin{bmatrix}r & -\mathbf{e}\end{bmatrix} \cdot \mathbf{z}\right\|_\infty$. Recall that we assume the ciphertext is honestly generated, including the noise $\begin{bmatrix}r & -\mathbf{e}\end{bmatrix}$. We now bound $\left\|\begin{bmatrix}r & -\mathbf{e}\end{bmatrix} \cdot \mathbf{z}\right\|_\infty$, using Lemma 9. Except with probability $\delta'/2$ it holds

$$\left\|\begin{bmatrix}r & -\mathbf{e}\end{bmatrix} \cdot \mathbf{z}\right\|_\infty \leq \|\mathbf{z}\| \cdot \sqrt{2(\log \delta' + 1 + \log(8d))} \cdot \sigma_{\mathbf{r}}. \tag{24}$$

Similarly, $\|e'\|_\infty$ is upper bounded by $B_r := \sqrt{2(\log\delta' + 1 + \log(2d))}\,\sigma_{\mathbf{r}}$. Overall, except with probability at most δ' it holds

$$\begin{aligned}\left\|e' + \begin{bmatrix} r & -\mathbf{e}\end{bmatrix} \cdot \mathbf{z}\right\|_\infty &\leq \sqrt{2(\log\delta' + 1 + \log(2d))}\,\sigma_{\mathbf{r}} \\ &\quad + \sqrt{2(\log\delta' + 1 + \log(8d))} \cdot \sigma_{\mathbf{r}} \cdot B_z \\ &\leq B_r(1 + B_z) \leq \lfloor q/4 \rceil,\end{aligned}$$

concluding the proof. □

4.3 Extraction Share Robustness

We now analyse whether the TIBE correctly decrypts, even if an adversary can maliciously generated key extraction shares (which still pass the share verification algorithm). Note that we still assume quasi-honestly generated ciphertexts. More concretely, we require the encryption randomness to come from RSpace (where honestly generated randomness lies in RSpace with overwhelming probability). We call this extraction share robustness, as defined in Definition 7.

First, note that our TIBE has $\mathsf{RDistr} = D_{\mathcal{R},\sigma_{\mathbf{r}}} \times D_{\mathcal{R}^{1\times 3},\sigma_{\mathbf{r}}} \times D_{\mathcal{R},\sigma_{\mathbf{r}}}$. By setting $\mathsf{RSpace} = \{(r, \mathbf{e}, e') \in R^{1\times 5} \colon \left\|(r, \mathbf{e}, e')^T\right\|_2 \leq \tau\,\sigma_{\mathbf{r}}\sqrt{5d}\}$, for $\tau \geq 1 + \sqrt{\frac{4(\kappa+1)\log 2}{5d}}$ we have by Lemma 8 that honestly sampled encryption randomness from RDistr lies in RSpace with overwhelming probability.

Lemma 14. *Let δ' be the target correctness probability of share verification in Algorithm 13 . Given* TIBE *as described in Figs. 4 and 5 with $\tau \geq 1 + \sqrt{\frac{4(\log\delta'+\log 2)}{2d}}$ and norm bounds B_m and B given as*

$$B_m = TB_{\mathsf{ind}} + B_0 + B_1, \text{ and } B = \tau\,\sigma_{\mathbf{r}}\sqrt{5d}\,\sigma_{\mathbf{r}},$$

where T is the number of parties, $\sigma_{\mathbf{r}}$ is the Gaussian width of the encryption randomness, B_0 (resp. B_1 and B_{ind}) is a norm bound on c_0 (resp. c_1 and $\mathbf{z}^{(i)}$). Furthermore assume that TIBE *is δ' extraction share correct. If $B \cdot (1 + B_m) \leq \lfloor q/4 \rceil$, then, the advantage of any PPT adversary $\mathcal{A}$ against the extraction share robustness game for* TIBE *is at most $T(\delta' + 1)$.*

Note that B_m is dominated by $T\sigma_{\mathbf{p}}$, as opposed to $\sqrt{T}\sigma_{\mathbf{p}}$ in the honest case considered for correctness in Lemma 13.

Proof. Informally, the three main differences between the extraction share robustness game and the decryption correctness, is that (i) the $\mathbf{z}^{(i)}$ are malicious, (ii) the encryption randomness might not follow the honest distribution, but only is guaranteed to lie in RSpace and (iii) the $\mathbf{z}^{(i)}$ might be colinear to the encryption randomness. This requires worse norm bounds to bound the norm of $\mathbf{z}$ and the inner product between $\mathbf{z}$ and encryption randomness.

More formally, note that Eq. (17) holds for every shares that pass the **assert** in Algorithm 14. Again we assume quasi-honest ciphertexts, so Eq. (20) holds even in the malicious key extraction parties setting. Let B_0 (resp. B_1) be a bound on the norm of c_0 (resp. c_1). Note that the adversary has no control over (c_0, c_1), so these bounds are independent whether we are looking at semi-honest or malicious parties.

Bounding $\|\mathbf{z}\|$. Here, we deviate from the analysis in the honest setting. By the equality and shortness checks in Algorithm 13 and the definition of $\mathbf{z}$, we have

$$\begin{aligned} \|\mathbf{z}\| &= \left\| \left(\sum_{i \in \mathsf{act}} \mathbf{z}^{(i)} \right) + \begin{bmatrix} c_1 & 0 & c_0 & 0 \end{bmatrix}^T \right\| && (25) \\ &\leq T\, B_{\mathsf{ind}} + B_1 + B_0 =: B_m. && (26) \end{aligned}$$

This reflects the fact that corrupted parties may collude to generate $\mathbf{z}^{(i)}$'s that are as colinear as possible.

Bounding $\left\| \begin{bmatrix} r & -\mathbf{e} \end{bmatrix} \cdot \mathbf{z} \right\|_\infty$. Note that in the malicious setting, we cannot apply Lemma 7, as we cannot guarantee that the ciphertext encryption randomness $\begin{bmatrix} r & -\mathbf{e} \end{bmatrix}$ has been sampled *honestly* and *independently* from the global preimage $\mathbf{z}$. We only know that it lies in RSpace. Hence, we use Cauchy-Schwarz instead:

$$\begin{aligned} \left\| e' + \begin{bmatrix} r & -\mathbf{e} \end{bmatrix} \cdot \mathbf{z} \right\|_\infty &= \left\| \begin{bmatrix} e' & r & -\mathbf{e} \end{bmatrix} \cdot (1, \mathbf{z}) \right\|_\infty \\ &\leq \|(e', r, -\mathbf{e})\|_2 \cdot \|(1, \mathbf{z})\|_2 \\ &\leq B \cdot (1 + B_m), \end{aligned}$$

concluding the proof. □

Regarding the norm bounds B_0 on c_0 and B_1 on c_1, w.o.p. it holds

$$B_0 = \frac{q}{\beta} \sqrt{\frac{d}{12}} (1 + o(1)), \qquad B_1 = \beta \sqrt{\frac{d}{12}} (1 + o(1)) \tag{27}$$

5 TIBE Security

Before starting with our security proof, we recall the definition of functional simulation of a pair of key generation and sharing algorithms. It is a simplified version of the one in [17], tailored to a trusted setup.

Definition 12 ([17]). *A key generation and sharing algorithms* KG *and* Share *together with a function oracle* Eval *are called functionally simulatable if there exist efficient functions* $\mathsf{SimShare}$ *and* $\mathsf{SimEval}$ *s.t. no PPT adversary* $\mathcal{A}$ *has a non-negligible advantage in the experiment defined in Fig. 6. We denote the advantage of an adversary* $\mathcal{A}$ *in* $\mathsf{Exp}^{\mathsf{KG\text{-}sim}}_{\mathcal{A},\mathsf{KG},\mathsf{Share},\mathsf{Eval}}(1^\kappa)$ *as* $\mathsf{Adv}^{\mathsf{KG\text{-}sim}}_{\mathcal{A},\mathsf{KG},\mathsf{Share},\mathsf{Eval}}$.

```
$\mathsf{Exp}^{\mathsf{KG\text{-}sim}}_{\mathcal{A},\mathsf{KG},\mathsf{Share},\mathsf{Eval}}(1^\kappa)$
$\mathsf{cor} \leftarrow \mathcal{A}(1^\kappa)$
assert $|\mathsf{cor}| = T - 1$
$(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KG}(1^\kappa)$, $b \leftarrow \{0,1\}$
if $b = 0$ : $\left(\{\mathsf{dk}^{(i)}\}_{i\in[N]}, \mathsf{aux}\right) \leftarrow \mathsf{Share}(\mathsf{dk})$
    $b' \leftarrow \mathcal{A}^{\mathsf{Eval}}\left(\mathsf{ek}, \{\mathsf{dk}^{(i)}\}_{i\in\mathsf{cor}}\right)$
else $\left(\{\mathsf{dk}^{(i)}\}_{i\in\mathsf{cor}}, \mathsf{aux}\right) \leftarrow \mathsf{SimShare}(\mathsf{ek}, \mathsf{cor})$
    $b' \leftarrow \mathcal{A}^{\mathsf{SimEval}}\left(\mathsf{ek}, \{\mathsf{dk}^{(i)}\}_{i\in\mathsf{cor}}\right)$
return $b = b'$
```

Fig. 6: Security experiment for Functional Simulatability.

Lemma 15 ([9, Lemma B.2]). *Let* KG *and* Share *be as defined in Algorithm 6 and let* $\mathsf{ShareExtract}_i$ *be as in Fig. 5. Then there exists a constant* $Q_v = Q_v(T, N, Q_{\mathsf{Dec}})$ *(see [5] Theorem 1) such that*

$$\mathsf{Adv}^{\mathsf{KG\text{-}sim}}_{\mathcal{A},\mathsf{KG},\mathsf{Share},\{\mathsf{ShareExtract}_i\}_i} \leq \mathsf{Hint\text{-}RLWE}^{TQ_{dec}+Q_v,\sqrt{\sigma_{\mathbf{p}}^2-{\sigma'_{\mathbf{p}}}^2},\sigma_{\mathbf{s}},\beta}_{\mathcal{R},q,m,\sigma_{\mathbf{s}}}.$$

Theorem 2. *Let* TIBE *be the scheme defined in Fig. 4 and Fig. 5. Let* $\mathcal{R} = \mathbb{Z}[X]/(X^d+1)$ *be a ring of degree d, and q be a prime number. Let* $\sigma, \sigma_{\mathbf{s}}, \sigma_{\mathbf{p}}, \sigma'_{\mathbf{p}}, \beta > 0$, $\varepsilon \in (0,1)$. *Suppose that:*

$$\min(\sigma_{\mathbf{p}}, \sigma'_{\mathbf{p}}) > 1.17q^{1/2} \cdot \eta_\varepsilon(R^2). \tag{28}$$

$$(\sigma'_{\mathbf{p}}/\sigma_{\mathbf{p}}) \cdot \sqrt{\sigma_{\mathbf{p}}^2 - \sigma'^2_{\mathbf{p}}} \geq \eta_\varepsilon(\mathcal{R}). \tag{29}$$

Then for any PPT adversary $\mathcal{A}$*:*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{sel\text{-}ID\text{-}IND}}_{\mathcal{A},\mathsf{TIBE}}(\kappa) \leq{}& (Q_{\mathsf{Dec}} + Q_{\mathsf{H_{id}}}) \cdot \mathsf{Adv}^{\mathsf{NTRU}_{\mathcal{R},q}}_{\mathcal{A}_1}(\kappa)\\ &+ \mathsf{Adv}^{\mathsf{Hint\text{-}RLWE}^{TQ_{dec}+Q_v,\sqrt{\sigma_{\mathbf{p}}^2-{\sigma'_{\mathbf{p}}}^2},\sigma_{\mathbf{s}},\beta}_{\mathcal{R},q,m,\sigma_{\mathbf{s}}}}_{\mathcal{A}_2}(\kappa) + \mathsf{Adv}^{\mathsf{RLWE}_{\mathcal{R},3,q,\sigma}}_{\mathcal{A}_3}(\kappa)\\ &+ Q_{\mathsf{Dec}} \cdot O(\varepsilon) + Q_{\mathsf{Dec}} \cdot (T\, Q_{\mathsf{Dec}} + Q_{H_{\mathsf{cmt}}}) \cdot (q^{-d} + \mathrm{negl}(\kappa))\\ &+ \frac{(Q_{\mathsf{Dec}} + Q_{H_{\mathsf{cmt}}})^2 + Q_{H_{\mathsf{cmt}}} \cdot T \cdot Q_{\mathsf{Dec}}}{2^{2\kappa}}\end{aligned}$$

where adversaries $\mathcal{A}_3$ *to* $\mathcal{A}_1$ *run in approximately the same time as* $\mathcal{A}$*.*

Proof. Consider a sequence of hybrids.

Hyb_0 *(*sel-ID-IND *game):* Identical to the real experiment.

Hyb_1 *(Delay w_h):* In this second hybrid we postpone sampling the noise $\mathbf{p}^{(i)}$ from $\mathsf{TIBE.ShareExtract}_0$ to the end of $\mathsf{TIBE.ShareExtract}_1$ for one of the honest parties. For each key extraction query $(\mathsf{id}, \mathsf{act})$ of the adversary we pick an arbitrary honest party in act as "last acting" honest party and denote its index as h. Now in $\mathsf{TIBE.ShareExtract}_0$ the party h samples $\mathsf{cmt}_h \leftarrow \{0,1\}^{2\kappa}$. And in $\mathsf{TIBE.ShareExtract}_1$ after every other party in act has revealed their values $\mathbf{w}^{(i)}$ the last honest party samples $\mathbf{p}^{(h)}$ honestly, computes $\mathbf{w}^{(h)} = \left[1 \ a \ b \ \mathsf{H}_{\mathsf{id}}(\mathsf{id})\right] \cdot \mathbf{p}^{(h)}$ and programs $\mathsf{H}_{\mathsf{cmt}}(\mathbf{w}^{(h)}) := \mathsf{cmt}_h$. The advantage of the adversary in this hybrid can only increase if $\mathbf{w}^{(h)}$ has been queried to $\mathsf{H}_{\mathsf{cmt}}(\cdot)$ before. However, $\mathbf{w}^{(h)}$ computed above has high entropy since by Lemma 2 over the vector $[1 \ a \ H(id)]$, with $\min(\sigma_{\mathbf{p}}, \sigma'_{\mathbf{p}}) \geq \sqrt{2\pi} \cdot 2d \cdot q^{1/3+2/3d}$ we have $\mathrm{SD}(\mathbf{w}^{(h)}, \mathcal{U}(\mathcal{R}_q)) = \mathrm{negl}(\kappa)$. Hence, the probability of querying a fixed $\mathbf{w}^{(h)}$ is bounded by $q^{-d} + \mathrm{negl}(\kappa)$ and the number of queries to $\mathsf{H}_{\mathsf{cmt}}$ is bounded by $T \cdot Q_{\mathsf{Dec}} + Q_{H_{\mathsf{cmt}}}$ where $Q_{H_{\mathsf{cmt}}}$ denotes the number of direct queries to $\mathsf{H}_{\mathsf{cmt}}$ oracle and $T \cdot Q_{\mathsf{Dec}}$ is a bound on queries made by the Challenger when answering the decryption queries. Then by the union, when we apply this change to every decryption query we get

$$\varepsilon_0 - \varepsilon_1 \leq (q^{-d} + \mathrm{negl}(\kappa)) \cdot Q_{\mathsf{Dec}} \cdot (T \cdot Q_{\mathsf{Dec}} + Q_{H_{\mathsf{cmt}}}).$$

Hyb_2 *(Trapdoors in the hash):* In this hybrid we change the answers of the $\mathsf{H}_{\mathsf{id}}(\cdot)$ oracle for all $\mathsf{id} \neq \mathsf{id}^*$. Now on query id instead of sampling a fresh uniformly random value the oracle samples a pair $(h_{\mathsf{id}}, \mathbf{T}_h) \leftarrow \mathsf{TrapGenNTRU}(1^\kappa)$ and replies with $H_{id}(\mathsf{id}) = h_{\mathsf{id}}$. The rest of the experiment is run as before. The total number of queries to $\mathsf{H}_{\mathsf{id}}(\cdot)$ is $Q_{\mathsf{Dec}} + Q_{\mathsf{H}_{\mathsf{id}}}$, hence, the adversary's advantage can only increase as

$$|\varepsilon_1 - \varepsilon_2| \leq (Q_{\mathsf{Dec}} + Q_{\mathsf{H}_{\mathsf{id}}}) \cdot \mathsf{Adv}_{\mathcal{A}_1}^{\mathsf{NTRU}_{\mathcal{R},q}}(\kappa)$$

where $\mathcal{A}_1$ runs in approximately the same time as $\mathcal{A}$.

Hyb_3 *(Hash collisions):* In this hybrid, we ensure that $\mathsf{H}_{\mathsf{cmt}}$ has no collisions and that the adversary cannot find preimages of hashes cmt_i after passing them to the second signing oracle. We introduce a table Cmt that records every cmt sampled in $\mathsf{H}_{\mathsf{cmt}}$ or in $\mathsf{TIBE.ShareExtract}_0$, as well as those passed to the second signing oracle. Whenever a new hash cmt is sampled, we ensure that it was not previously added to Cmt, or the challenger aborts the game. The view of the adversary differs in Hyb_3 if it was previously able to (i) find a pre-image of a cmt_i after passing it to $\mathsf{TIBE.ShareExtract}_1$, or (ii) if a given cmt was sampled twice. We can bound the probability of (i) due to the pre-image resistance of the hash function $\mathsf{H}_{\mathsf{cmt}}$, and with an union bound over all the commits passed by the adversary to $\mathsf{TIBE.ShareExtract}_1$ which gives a bound $Q_{H_{\mathsf{cmt}}} \cdot T \cdot Q_{\mathsf{Dec}} \cdot 2^{-2\kappa}$. As for (ii), we rely on the collision resistance of $\mathsf{H}_{\mathsf{cmt}}$. Since the commitments are sampled uniformly from $\{0,1\}$, we can bound the probability of (ii) by $(Q_{H_{\mathsf{cmt}}} + Q_{\mathsf{Dec}})^2 \cdot 2^{-2\kappa}$ We conclude that,

$$|\varepsilon_2 - \varepsilon_3| \leq \frac{(Q_{\mathsf{Dec}} + Q_{H_{\mathsf{cmt}}})^2 + Q_{H_{\mathsf{cmt}}} \cdot T \cdot Q_{\mathsf{Dec}}}{2^{2\kappa}}$$

Hyb_4 *(Uniform (c_0, c_1)):* In this hybrid, for each decryption we want to set the "challenge" $u = t - w$ to be uniform and then set the honest $\mathbf{z}^{(i)}$ accordingly. We first sample $u \leftarrow \mathcal{R}_q$, and set $\mathbf{w}^{(h)} = t - u - \sum_{i \in \mathsf{act} \setminus h} \mathbf{w}^{(i)}$, as shown in Hyb_1 the distribution of $w^{(h)}$ is close to uniform so this change is statistical. The behaviour of all parties except h is unchanged, party h however needs to still be able to sample $\mathbf{p}^{(h)}$ from the appropriate distribution. Note that sampling $\mathbf{p}^{(h)} \leftarrow D_{\mathcal{R}^2, \sigma_{\mathbf{p}}} \times \{0\} \times D_{\mathcal{R}, \sigma'_{\mathbf{p}}}$ and setting $w^{(h)} = \left[1 \ a \ b \ \mathsf{H}_{\mathsf{id}}(\mathsf{id})\right] \cdot \mathbf{p}^{(h)}$ is the same as sampling $w^{(h)}$ from the appropriate distribution (statistically close to uniform) and sampling $\mathbf{p}^{(h)} \leftarrow \mathcal{D}_{\Lambda_q^{\mathbf{w}^{(h)}}(\mathbf{F}_{\mathsf{id}}), \mathbf{\Sigma}}$ with $\mathbf{\Sigma} = \mathsf{diag}(\sigma_{\mathbf{p}}, \sigma_{\mathbf{p}}, \sigma'_{\mathbf{p}})$ and $\mathbf{F}_{\mathsf{id}} = \left[1 \ a \ \mathsf{H}_{\mathsf{id}}(\mathsf{id})\right]$. Using Lemma 4 this is statistically close to sampling $\tilde{\mathbf{p}} = (\mathbf{p}_0^{(h)}, \mathbf{p}_1^{(h)}) \leftarrow \mathcal{D}_{\mathcal{R}^2, \sigma_{\mathbf{p}}}$ and $\mathbf{p}_3^{(h)} \leftarrow \mathcal{D}_{\Lambda_q^{\mathbf{w}^{(h)} - \mathbf{F}\tilde{\mathbf{p}}}(\mathsf{H}_{\mathsf{id}}(\mathsf{id})), \sigma'_{\mathbf{p}}}$, with $\mathbf{F} = [1 \ a]$. We want to use the trapdoor introduced in the previous hybrid to sample this preimage, since the trapdoor is for the matrix $[1 \ \mathsf{H}_{\mathsf{id}}(\mathsf{id})]$ we consider the following transformation: First sample $\tilde{\mathbf{p}} = (\mathbf{p}_0^{(h)}, \mathbf{p}_1^{(h)}) \leftarrow \mathcal{D}_{\mathcal{R}, \sqrt{\sigma_{\mathbf{p}}^2 - {\sigma'_{\mathbf{p}}}^2}} \times \mathcal{D}_{\mathcal{R}, \sigma_{\mathbf{p}}}$ then sample $(\mathbf{x}, \mathbf{p}_3^{(h)}) \leftarrow \mathcal{D}_{\Lambda_q^{\mathbf{w}^{(h)} - \mathbf{F}\tilde{\mathbf{p}}}([1 \ \mathsf{H}_{\mathsf{id}}(\mathsf{id})]), \sigma'_{\mathbf{p}}}$ and output $(\mathbf{p}_0^{(h)} + \mathbf{x}, \mathbf{p}_1^{(h)}, \mathbf{p}_3^{(h)})$. By Lemma 5, as long as the smoothing condition in Eq. (29) holds, this modification is statistically unnoticeable. The NTRU trapdoor can sample such a distribution according to Lemma 6 if $\sigma_{\mathbf{p}} > \sigma'_{\mathbf{p}} \geq 1.17\sqrt{q}\eta_\epsilon(\mathcal{R}^2)$. Gathering the various bounds, the advantage of the adversary can increase by at most

$$|\varepsilon_3 - \varepsilon_4| \leq (\mathrm{negl}(\kappa) + 2\,\varepsilon) \cdot Q_{\mathsf{Dec}}.$$

Hyb_5 *(Secret Sharing):* Let $\mathsf{SimShare}$ and $\mathsf{SimShareExtract}$ be as per Lemma 15. In this hybrid we sample the public key without a trapdoor (i.e. $b \leftarrow \mathcal{R}_q$) and use $\mathsf{SimShare}$ to simulate the corrupt key shares. Using Lemma 15 we observe that all honest partial secrets $(s^{(i)}, {s^{(i)}}')$ can be written either as a linear combination of the leaked secrets, or as (s, s') plus a linear combination of the leaked secrets. Meaning that there are $Q_v(T, N)$ hints of the form $\mathbf{s} + \mathbf{r}$ (corresponding to the leakages) and at most $T \cdot Q_{\mathsf{Dec}}$ hints of the form $\mathbf{s} \cdot c + \mathbf{p}$. Since we do not have access to the secret keys, the partial decryption oracles are simulated using the corresponding $\mathsf{SimShareExtract}$. Then by Lemma 15 the advantage changes as

$$|\varepsilon_4 - \varepsilon_5| \leq \mathsf{Adv}^{\mathsf{KG\text{-}sim}}_{\mathcal{A}_2, \mathsf{KG}, \mathsf{Share}, \{\mathsf{ShareExtract}_i\}_i} \leq \mathsf{Adv}_{\mathcal{A}_2}^{\mathsf{Hint\text{-}RLWE}^{Q_v + TQ_{dec}, \sqrt{\sigma_{\mathbf{p}}^2 - {\sigma'_{\mathbf{p}}}^2}, \sigma_{\mathbf{s}}, \beta}_{\mathcal{R}, q, m, \sigma_{\mathbf{s}}}}.$$

Hyb_6 *(Ciphertext RLWE):* Note that now $\mathbf{F}_{\mathsf{id}^*} = \left[a \quad b \ \mathsf{H}_{\mathsf{id}}(\mathsf{id}^*)\right]$ is perfectly random and the experiment can be simulated efficiently. Hence, we apply the RLWE assumption to the challenge ciphertext. As a result, the challenge plaintext is masked by a uniformly random value and the advantage in this experiment is $\varepsilon_5 = 0$. The advantage change can also be bounded as

$$|\varepsilon_5 - \varepsilon_6| \leq \mathsf{Adv}_{\mathcal{A}_3}^{\mathsf{RLWE}_{\mathcal{R}, 3, q, \sigma_{\mathbf{r}}}}(\kappa).$$

where $\mathcal{A}_3$ runs in approximately the same time as $\mathcal{A}$. $\square$

On achieving adaptive corruptions. In all security notions we consider in this work, the adversary has to decide which $T-1$ parties to corrupt at the very beginning of the game. This is usually called *static* corruptions. Closer to the real world is the concept of *adaptive* corruptions where the adversary can decide during the game which parties they want to corrupt, in particular after already having seen partial key extraction shares. In general, the latter is significantly harder to achieve as it requires the shares of the yet-honest parties given to the adversary during key extraction queries to stay consistent even after the set of corrupted parties changes. Interestingly, we observe that [33] achieves security against adaptive corruptions, without making this explicit in their work. The main reason is that simulating communications of honest parties is very easy thanks to the one-time masks issued by a random oracle.

However, when replacing the masks-with-Shamir-sharing approach by the VSS as done in this work, this argument does not work anymore and we are not able to prove security against adaptive corruptions. Let us give some more intuition on why VSS seems difficult to use in an adaptive corruption context. During the security proof we simulate both the corrupted secret shares and the communication of honest parties during the interactive key extraction procedure. To do the simulation correctly, we need to know which parties are corrupted and which are honest, as it has an impact on where we require RLWE hints. Moreover, in the security proof of VSS, we simulated the honest parties' communication as a function of hints and corrupted key shares. Thus, changing the set of corrupted parties would lead to inconsistent simulation.

If one prioritises security against adaptive corruptions over share robustness, one can modify our scheme as follows (to get closer again to [33]): set $\mathbf{z}^{(i)} = \mathbf{p}^{(i)} + c_0 \cdot \mathbf{s}^{(i)} + m_i$, where m_i a blinding one-time mask. When simulating key extraction shares for honest parties, one simply samples random $\mathbf{z}^{(i)}$, conditioned on the sum of all shares, namely $\mathbf{z}$, to fulfill $\begin{bmatrix}1 & a & b & \mathsf{H}_{\mathsf{id}}(\mathsf{id})\end{bmatrix} \cdot \mathbf{z} = t$. One can again use the notion of "last acting" honest party to achieve this. This simulation does not reveal the masks m_i. When a honest party later gets corrupted, one can simply program the random oracle that controls the masks m_i according to the honest key share $\mathbf{s}^{(i)}$. Note that this is possible, as m_i changes with every key extraction query.

We leave it as an interesting future work to simultaneously achieve security against adaptive corruptions *and* extraction share robustness for our TIBE.

6 Stronger TKEM Decapsulation Consistency

We now prove that the resulting TKEM from the generic BCHK+ transform of [33] fulfills a stronger decapsulation consistency notion as the one proven in [33], if the starting TIBE additionally fulfills extraction share robustness.

6.1 Definition

In Figure 7a we recall the decryption consistency experiment used in [33] and in Figure 7b we recall the decryption consistency notion of [20, Def. A.3], which we are achieving in our work.

```
$\mathsf{Exp}^{\mathsf{weak\text{-}CD}}_{\mathcal{A},\mathsf{TKEM}}(1^\kappa)$
$\mathsf{cor} \leftarrow \mathcal{A}$
**assert** $|\mathsf{cor}| = T - 1$
$(\mathsf{ek}, \{\mathsf{dk}_i\}_{i \in N}) \leftarrow \mathsf{TKEM.Keygen}(1^\kappa)$
$(\mathsf{ct}, \mathsf{act}, \mathsf{act}', S, S')$
  $\leftarrow \mathcal{A}^{\mathsf{TKEM.ShareDecaps}(\cdot)}(\mathsf{ek}, \{\mathsf{dk}_i\}_{i \in \mathsf{cor}})$
$K = \mathsf{TKEM.Combine}(\mathsf{ct}, \mathsf{act}, S)$
$K' = \mathsf{TKEM.Combine}(\mathsf{ct}, \mathsf{act}', S')$
**return** $\perp \neq K \neq K' \neq \perp$
```

(a) [33, Def. 7]

```
$\mathsf{Exp}^{\mathsf{CD}}_{\mathcal{A},\mathsf{TKEM}}(1^\kappa)$
$(\mathsf{ek}, \{\mathsf{dk}_i\}_{i \in N}) \leftarrow \mathsf{TKEM.Keygen}(1^\kappa)$
$(\mathsf{ct}, \mathsf{act}, \mathsf{act}', S, S')$
  $\leftarrow \mathcal{A}(\mathsf{ek}, \{\mathsf{dk}_i\}_{i \in [N]})$
**for** $i \in \mathsf{act}$
  **assert** $\mathsf{TKEM.ShareVerify}(\mathsf{ek}, \mathsf{ct}, \mathsf{act}, S, i)$
**for** $i \in \mathsf{act}'$
  **assert** $\mathsf{TKEM.ShareVerify}(\mathsf{ek}, \mathsf{ct}, \mathsf{act}', S', i)$
$K = \mathsf{TKEM.Combine}(\mathsf{ct}, \mathsf{act}, S)$
$K' = \mathsf{TKEM.Combine}(\mathsf{ct}, \mathsf{act}', S')$
**return** $K \neq K'$
```

(b) [20, Def. A.3]

Fig. 7: Security experiments for TKEM Decapsulation Consistency

Note that the experiment in Figure 7b leads to a stronger security guarantee than the one in Figure 7a as the adversary wins even if one of the resulting keys is equal to $\perp$. To accommodate for this change, we require the decryption shares to pass the share verification algorithm.

Another change is that we give out all secret key shares to the adversary, whereas [33, Def. 7] only gives out corrupted key shares and unrestricted access to TKEM.ShareDecaps. In general the latter is weaker than the first. However, this limitation was not needed for the proof of [33], they could have given all secret key shares to the adversary.

To give some more intuition on why unrestricted access to TKEM.ShareDecaps is not as powerful as having the honest secret key shares, we can look at our concrete lattice constructions from Section 3: the adversary can query TKEM.ShareDecaps on the same ciphertext many times, providing them with many short preimages of the same matrix $\begin{bmatrix}1 & a & b & H_{\mathsf{id}}(\mathsf{id} = \mathsf{vk})\end{bmatrix}$ for a fixed target t. This can be seen as a trapdoor for this identity-depending matrix. However, this is not enough to recover honest secret key shares, which are small preimages of the matrix $\begin{bmatrix}1 & a & b\end{bmatrix}$.

6.2 Proof

Lemma 16. *Let $G_{\mathsf{fo}} \colon \{0,1\}^* \rightarrow \{0,1\}^\kappa$ be modeled as a random oracle and let $Q_{G_{\mathsf{fo}}} \geq 2$ denote the number of queries to it. Let* TKEM *be the scheme obtained as detailed in Figure 2. If the underlying* TIBE *is γ-spread and extraction share robust, so fulfills the resulting* TKEM *decapsulation consistency. More concretely, for any PPT adversary $\mathcal{A}$ against $\mathsf{Exp}^{\mathsf{CD}}_{\mathcal{A},\mathsf{TKEM}}$ we can build a PPT adversary $\mathcal{B}$ against $\mathsf{Exp}^{\mathsf{Robust}}_{\mathcal{B},\mathsf{TIBE}}$ such that*

$$\mathsf{Adv}^{\mathsf{CD}}_{\mathcal{A},\mathsf{TKEM}} \leq \frac{Q_{F_{\mathsf{fo}}} \cdot (Q_{F_{\mathsf{fo}}} - 1)}{2} 2^{-\gamma} + \mathsf{Adv}^{\mathsf{Robust}}_{\mathcal{B},\mathsf{TIBE}}.$$

Proof. Let $\mathcal{B}$ be an adversary in $\mathsf{Exp}^{\mathsf{Robust}}_{\mathcal{B},\mathsf{TIBE}}$ and $\mathcal{A}$ an adversary in $\mathsf{Exp}^{\mathsf{CD}}_{\mathcal{A},\mathsf{TKEM}}$. Upon receiving the public evaluation key ek and all secret key shares $(\mathsf{dk}_i)_{i \in [N]}$, the reduction $\mathcal{B}$ forwards it to $\mathcal{A}$. Now, assume $\mathcal{A}$ wins in $\mathsf{Exp}^{\mathsf{CD}}_{\mathcal{A},\mathsf{TKEM}}$. That is, they output $(\mathsf{ct}, \mathsf{act}, \mathsf{act}', S, S')$ such that all decapsulation shares verify for both (act, S) and (act', S'), and $K \neq K'$, where $K = \mathsf{TKEM.Combine}(\mathsf{ct}, \mathsf{act}, S)$ and $K' = \mathsf{TKEM.Combine}(\mathsf{ct}, \mathsf{act}', S')$.

We distinguish two cases. In the first case neither K nor K' equals $\perp$. This is essentially the case considered in [33]. In the second case, exactly one of the keys equals $\perp$.

Case 1) As neither K nor K' equals $\perp$, all assert checks in $\mathsf{TKEM.Combine}$ from the BCHK+ transform detailed in Figure 2 pass. In particular, when running $\mathsf{TKEM.Combine}$, the reduction $\mathcal{B}$ recovers $\mathsf{msg}, \mathsf{msg}'$ where $K = H_{\mathsf{fo}}(\mathsf{msg} \| \mathsf{ct})$ and $K' = H_{\mathsf{fo}}(\mathsf{msg}' \| \mathsf{ct})$ but $\mathsf{msg} \neq \mathsf{msg}'$. At the same time, both messages lead to ct, i.e., $\mathsf{TKEM.Encaps}(\mathsf{ek}, \mathsf{vk}, \mathsf{msg}; G_{\mathsf{fo}}(\mathsf{msg})) = \mathsf{TKEM.Encaps}(\mathsf{ek}, \mathsf{vk}, \mathsf{msg}'; G_{\mathsf{fo}}(\mathsf{msg}'))$ $= \mathsf{ct}$. As done in [33], one can now distinguish two subcases. Either both msg and msg' have already been queried to G_{fo} or one of them has not been queried to G_{fo}. By using the γ-spreadness of TIBE, we can upper bound the probability in both subcases by $\frac{Q_{F_{\mathsf{fo}}} \cdot (Q_{F_{\mathsf{fo}}} - 1)}{2} 2^{-\gamma}$. Note that the argument is purely probabilistic and does not depend on $\mathcal{A}$ having access to only corrupted (as proven in [33]) or all secret key shares (as proven in our work).

Case 2) Without loss of generality, we assume $K' = \perp$ and $K \neq \perp$. As $K \neq \perp$ we know as above that the assert checks in $\mathsf{TKEM.Combine}$ passed and hence $K = H_{\mathsf{fo}}(\mathsf{msg} \| \mathsf{ct})$ and $\mathsf{rand} = G_{\mathsf{fo}}(\mathsf{msg})$. The reduction $\mathcal{B}$ then forwards the tuple $(\mathsf{msg}, \mathsf{vk}, \mathsf{rand}, \mathsf{ct}, \mathsf{act}', S')$ as output for the robustness game of the TIBE as defined in Figure 1e. We now argue that this output is a valid forgery in the robustness game of the TIBE. First, note that $\mathsf{rand} = G_{\mathsf{fo}}(\mathsf{msg})$, with rand following the randomness distribution RDistr and hence lying in RSpace with overwhelming probability. Second, by the winning condition of the decapsulation consistency game, the partial shares of $\mathsf{TKEM.ShareVerify}$ have to verify, which implies that the partial shares of $\mathsf{TIBE.ShareVerify}$ verify. It also implies that the signature verification passes. Lastly, $\mathsf{TIBE.Combine}(\mathsf{ct}, \mathsf{vk}, \mathsf{act}', S') \neq \mathsf{msg}$. Else $K' = K$, leading to a contraction. $\square$

7 Instantiation

Parameter selection for our scheme with semi-honest correctness (resp. malicious correctness) is performed by ensuring that both Theorem 2 and Lemma 13 (resp. Lemma 14) provide negligible bounds for the adversary's advantage and the decryption failure probability.

Recall from Theorem 2 that the adversary's advantage is upper-bounded as in the equation below. We study each term in one paragraph, then we study correctness, then we conclude.

$$\mathsf{Adv}^{\mathsf{sel\text{-}ID\text{-}IND}}_{\mathcal{A},\mathsf{TIBE}}(\kappa) \leq \overbrace{\mathsf{Adv}^{\mathsf{Hint\text{-}RLWE}^{Q_v,Q_M,\sqrt{\sigma_{\mathbf{p}}^2-\sigma_{\mathbf{p}}'^2},\sigma_{\mathbf{s}},\beta}_{\mathcal{R},q,m,\sigma_{\mathbf{s}}}}_{\mathcal{A}_2}(\kappa)}^{\mathsf{ct}\text{ is pseudorandom}} + \overbrace{\mathsf{Adv}^{\mathsf{RLWE}_{\mathcal{R},3,q,\sigma_{\mathbf{r}}}}_{\mathcal{A}_3}(\kappa)}^{\mathsf{ek}\text{ is pseudorandom}} \quad (30)$$

$$(Q_H + Q_{\mathsf{Dec}}) \cdot \underbrace{\mathsf{Adv}^{\mathsf{NTRU}_{\mathcal{R},q}}_{\mathcal{A}_1}(\kappa)}_{\text{Embed trapdoor}} + \mathrm{negl}(\kappa). \quad (31)$$

RLWE. The pseudorandomness of the ciphertext ct is quantified by the hardness of $\mathsf{RLWE}_{\mathcal{R},3,q,\sigma_{\mathbf{r}}}$. We can estimate it by using standard tools such as the lattice estimator by Albrecht et al. [3].

Hint-RLWE. Following the reduction of Lemma 10, $\mathsf{Hint\text{-}RLWE}^{Q_v,Q_M,\sqrt{\sigma_{\mathbf{p}}^2-\sigma_{\mathbf{p}}'^2},\sigma_{\mathbf{s}},\beta}_{\mathcal{R},q,m,\sigma_{\mathbf{s}}}$ is no easier than $RLWE_{\mathcal{R},q,1,\sigma_0}$, where $\frac{1}{\sigma_0^2} \leq 2\left(\frac{1+Q_v(T,N)}{\sigma_{\mathbf{s}}^2} + \frac{B_{\mathsf{hint}}(1+o(1))}{\sigma_{\mathbf{p}}^2}\right)$, $B_{\mathsf{hint}} = \frac{Q\,d\,M^2}{12}$, $M = 2\left\lceil \frac{q-1}{2\beta}\right\rceil + 1$, and Q_v as in [5] Theorem 1. As typical in other works, we ignore the factor 2 in $1/\sigma_0^2$ which seems to be a proof artefact. We also set $\sigma_{\mathbf{s}} = \Omega(\sqrt{Q_v(T,N)}) = \omega(\sqrt{B_{\mathsf{hint}}} \cdot \sigma_{\mathbf{p}})$, so that it has a negligible influence on σ_0. All in all, we have:

$$\sigma_0 \approx \sigma_{\mathbf{p}}/\sqrt{B_{\mathsf{hint}}} \approx \sigma_{\mathbf{p}}\sqrt{\frac{12}{Q\,d\,M^2}} \approx \sigma_{\mathbf{p}}\frac{\beta}{q}\sqrt{\frac{12}{Q\,n}}. \quad (32)$$

Eq. (32) incentivises us to take $\sigma_{\mathbf{p}}$ large. This condition directly subsumes the one of Theorem 2 are fulfilled. Again, we estimate the hardness of $\mathsf{Hint\text{-}RLWE}_{\mathcal{R},q,\sigma_0}$ using standard tools [3].

NTRU. The multiple instances of NTRU allow us to embed a trapdoor in $H_{\mathsf{id}}(\mathsf{id})$ (one for each identity queried). The hardness of NTRU is well-studied, and tools for estimating it are included in Albrecht et al.'s lattice estimator [3].

Note that our choices of q are very large. This raises the question of whether overstretched NTRU attacks [2,31,21] apply. Our experiments with [3] show that it is not the case. This is not surprising since we can take a large standard deviation σ for the secrets f, g, namely $\sigma \geq \sqrt{q/2\,n}$ as in Falcon [41]. The state-of-the-art attacks [27] states that on overstretched NTRU attacks become relevant when:

$$0.0058 \cdot \sigma^2 \cdot n^{2.484}/q < 1. \tag{33}$$

In our case, the left side of Eq. (33) is no smaller than $0.0029 \cdot n^{1.484}$, which gives us a comfortable margin.

Correctness. In the semi-honest (resp. malicious) setting, we apply Lemma 13 (resp. Lemma 14) and therefore set our parameters so that their requirements apply. In practice, $B_{\mathsf{ind}} \approx \tau \sqrt{3\,n}\,\sigma_{\mathbf{p}}$. Note that Lemmas 13 and 14 guarantee decryption correctness with probability at least $1 - \delta$. In our case it suffices to set δ to be small (say, 2^{-30}), not negligible. Indeed, the BCHK$^+$ transform guarantees CCA security even in the presence of decryption failures.

Bit dropping. We may apply bit dropping on four ring elements: the encryption key b, the first ciphertext v, the two last elements of the second ciphertext (u_1, u_2). This has the benefit of decreasing the public key size and ciphertext size. Let us note $\delta_b, \delta_v, \delta_{u_1}, \delta_{u_2}$ the (deterministic) errors incurred by bit-dropping. This error addition has absolutely no negative impact on the security analysis, however it might impact correctness since the decryption equation becomes:

$$v - \mathbf{u} \cdot \mathbf{z}' = \mathsf{Encode}(\mathsf{msg}) + (e' + \delta_v) + r \cdot (z_0 + c_0 \cdot \delta_b) \tag{34}$$

$$- (e_0 \cdot z_1 + (e_1 + \delta_{u_1}) \cdot c_0 + (e_2 + \delta_{u_2}) \cdot p_3) \tag{35}$$

As long as $\|\delta_v + c_0\,(r \cdot \delta_b - \delta_{u_1}) - \delta_{u_2} \cdot p_3\|_\infty = o(q)$ w.o.p., then bit-dropping will have a negligible impact on correctness. We note $\nu_b, \nu_{u_1}, \nu_{u_2}, \nu_v$ the bits dropped in b, u_1, u_2, v respectively.

Coefficient dropping. Since we encapsulate a key of κ bits, we only need to send κ coefficients of v. We can therefore drop the remaining $n - \kappa$ coefficients of v, which further decrease the ciphertext size.

How to select parameters. Selecting parameters is a multi-dimensional optimisation process, and it can seem obscure. We explain here our selection process. We recall Hardy's asymptotic notation: $a \asymp b$ is shorthand for $a = \Theta(b)$.

We select $\sigma_{\mathbf{r}} \asymp \sigma_0 \asymp 1$ (recall that σ_0 is found in the RLWE instance reduced from $\mathsf{Hint\text{-}RLWE}$ reduction), as we find that the resulting RLWE instances have κ bits of security against the most common attacks as long as $d \asymp \kappa \cdot \log q$. This seems somewhat optimal, as setting $\sigma_{\mathbf{r}} = o(1)$ quickly opens the door to hybrid attacks, while setting $\sigma_{\mathbf{r}} = \omega(1)$ degrades correctness (and similarly for σ_0).

From Eq. (32), this entails $\sigma_{\mathbf{p}} \asymp \frac{q}{\beta}\sqrt{Q\,d}$ and $\sigma_{\mathbf{s}} = \omega(1)$. We also require $\sigma_{\mathbf{s}} = o\left(\frac{\beta}{d\,q\,\sqrt{\log N}}\sigma_{\mathbf{p}}\right)$, so that $B_{\mathsf{ind}} \asymp \sqrt{d}\,\sigma_{\mathbf{p}} \asymp \frac{q\,d}{\beta}\sqrt{Q}$, see Eq. (16) .

Next, we set β. Recall that $B_1 \asymp \beta\sqrt{d}$. Setting β as high as possible is good for the $\mathsf{Hint\text{-}RLWE}$ reduction (see Eq. (32)) but may increase $\|\mathbf{z}\|$, see Eqs. (23)

and (26) . We set $B_1 \asymp \sqrt{T}\, B_{\mathsf{ind}}$ for the honest (that is, no robustness) setting, and $B_1 \asymp T\, B_{\mathsf{ind}}$ for the robust extraction shares setting, so that B_1 only increases $\|\mathbf{z}\|$ by a constant factor. This gives:

$$\textit{Non-Robust setting:} \quad \beta^2 \asymp q\sqrt{d\,T\,Q} \tag{36}$$

$$\textit{Robust setting:} \quad \beta^2 \asymp q\,T\sqrt{d\,Q} \tag{37}$$

Finally, we study correctness, and seek to minimise q. In the non-robust setting, Lemma 13 guarantee correctness if $q \asymp \sqrt{T}\, B_{\mathsf{ind}}\, \sqrt{\kappa}\, \sigma_{\mathbf{r}} \asymp \frac{q\,d}{\beta}\sqrt{Q\,T\,\kappa}$, which implies $\beta \asymp d\sqrt{Q\,T\,\kappa}$. In the robust setting, Lemma 14 guarantee correctness if $q \asymp T\, B_{\mathsf{ind}}\, \sqrt{d}\, \sigma_{\mathbf{r}} \asymp \frac{q\,T\,d}{\beta}\sqrt{Q\,d}$, which implies $\beta \asymp d\,T\sqrt{Q\,d}$. In summary:

$$\textit{Non-Robust setting:} \quad \beta^2 \asymp q\sqrt{d\,T\,Q} \asymp d^2\,Q\,T\,\kappa \quad \Rightarrow \quad q \asymp d^{3/2}\,\kappa\,\sqrt{Q\,T} \tag{38}$$

$$\textit{Robust setting:} \quad \beta^2 \asymp q\,T\sqrt{d\,Q} \asymp d^3\,T^2\,Q \quad \Rightarrow \quad q \asymp d^{5/2}\,T\,\sqrt{Q} \tag{39}$$

This gives us an asymptotic formula to determine a minimal viable q. We increase d until the underlying RLWE instances are hard. Finally, bit-dropping has a small impact on correctness as long as $\sigma_{\mathsf{bitdrop}} = O(q/\sqrt{\kappa})$, so we may take $2^{\nu_v} \asymp \sqrt{d}\frac{q}{\beta}2^{\nu_{u_1}} \asymp \sqrt{d\,q}2^{\nu_{u_2}} \asymp \frac{d\,q}{\beta}2^{\nu_b} \asymp q/\sqrt{\kappa}$.

Parameter sets. Putting everything together, we obtain parameter sets as detailed in Table 1. The encryption key and ciphertext sizes (in bytes) for our TIBE are given as:

$$|\mathsf{ek}| = \left(2 \cdot \kappa + d\left(\log q - \nu_b\right)\right)/8 \tag{40}$$

$$|\mathsf{ct}| = \left(d\left(3\log q - \nu_{u_1} - \nu_{u_2}\right) + \kappa(\log q - \nu_v)\right)/8 \tag{41}$$

For the TKEM construction, the ciphertext additionally contains a verification key vk and a signature sig. For ML-DSA at NIST security levels 2, and 5, the sum $|\mathsf{vk}| + |\mathsf{sig}|$ is equal to 3732, and 7219, respectively. For Falcon [41] at NIST security levels 1 and 5, the sum $|\mathsf{vk}| + |\mathsf{sig}|$ is equal to 1563, and 3073, respectively.

How to select parameters in [33]. For comparison, here is a succinct evaluation of [33] with the same methodology we used above. We note that [33] define Gaussians via the notation $\exp(-\pi \cdot \|\mathbf{x}\|^2/\varsigma^2)$, so their ς correspond to $\sigma\sqrt{2\pi}$. With the notations of [33], we have the following constraints:

1. **Hardness of RLWE.** $\mathsf{RLWE}_{R,1,q,\varsigma_{\mathsf{RLWE}}}$ and $\mathsf{RLWE}_{R,7,q,\varsigma}$ need to be hard. Let us take $\varsigma \asymp 1$ and $\varsigma_{\mathsf{RLWE}} \asymp 1$, we can then later set $d = \Omega(\kappa\, \log q)$ as explained previously. [33] also requires $\varsigma > \eta_\epsilon(R_3)$, but for concrete parameters $\eta_\epsilon(R_3)$ remains small.
2. **Rerandomization.** Equation (10) of [33] requires $\varsigma' \asymp d \cdot \varsigma_{\mathsf{RLWE}} \cdot \varsigma \asymp d$ in order to apply the rerandomization lemma of [29].

Table 1: κ is the bit-security, T (resp. Q) is an upper bound on the decryption threshold (resp. number of decryption queries), R = "Extraction Share Robustness", κ_{ek} (resp. κ_{ct}) quantifies the pseudorandomness of the encryption key (resp. ciphertext) in the CoreSVP metric, and $|\mathsf{ek}|, |\mathsf{ct}|$ are the sizes in bytes of the TIBE encryption key and ciphertext. Across all parameter sets, we take $T = 32$, $q \approx 2^{50}$, $\sigma_{\mathbf{s}} = 2^{15}$, $\sigma'_{\mathbf{p}} = 2^{27}$ and $\nu_{u_2} = 10$.

Goals			Scheme parameters								Security		Sizes	
κ	R	Q	d	β	$\sigma_{\mathbf{s}}$	$\sigma_{\mathbf{p}}$	$\sigma_{\mathbf{r}}$	ν_b	ν_{u_1}	ν_v	κ_{ek}	κ_{ct}	$\lvert\mathsf{ek}\rvert$	$\lvert\mathsf{ct}\rvert$
128	No	2^{46}	2048	2^{41}	2^{15}	2^{35}	1	24	29	42	128	133	6 688	28 544
128	Yes	2^{25}	2048	2^{37}	2^{15}	2^{29}	1	21	27	44	132	133	7 456	29 056
256	No	2^{46}	4096	2^{40}	2^{15}	2^{36}	$1/4$	26	30	44	263	253	12 352	56 512
256	Yes	2^{25}	4096	2^{36}	2^{15}	2^{29}	$1/4$	22	26	44	257	253	14 400	58 560

3. **Hardness of Hint-RLWE.** The hardness of Hint-MLWE requires $\varsigma_0 = \Omega(1)$, therefore we set $\varsigma_a \asymp 1$ and $\varsigma_p \asymp (q/\beta) \cdot \sqrt{Q\,d}$. Maximizing the number of decryption queries Q requires us to take a very large β. There are two other conditions on ς_p, Equations (11) and (12) in [33, Theorem 3], but both are subsumed by the one above.
4. **Correctness.** Finally, correctness is given by [33, Theorem 5], more specifically their Equation (13). The left term of this equation is minimised for $\beta \asymp d^2 \sqrt{q\,T}\, Q^{1/4}$, in which case Equation (13) of [33] simplifies to $q \asymp d^5 \sqrt{Q\,T}$.

With our methodology, for [33] we obtain the constraint $q \asymp d^5 \sqrt{Q\,T}$. In comparison, our non-robust scheme only requires $q \asymp d^{3/2}\,\kappa\sqrt{Q\,T}$, which is better by a factor $d^{7/2}/\kappa$.

Acknowledgements

We thank the anonymous reviewers as well as Diego F. Aranha for their helpful feedback. Katharina Boudgoust is supported by the French National Research Agency (ANR), under the projects ANR-21-ASTR-0016 AMIRAL, ANR-22-PECY-003 SecureCompute and ANR-25-CE39-4214-01 RELATE. Oleksandra Lapiha was supported by the EPSRC and the UK Government as part of the Centre for Doctoral Training in Cyber Security for the Everyday at Royal Holloway, University of London (EP/S021817/1). Thomas Prest and Rafaël del Pino are supported by the ANR, under the project ANR-25-CE39-4214-01 RELATE.

References

1. Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 553–572. Springer, Berlin, Heidelberg, May / June 2010.

2. Martin R. Albrecht, Shi Bai, and Léo Ducas. A subfield lattice attack on overstretched NTRU assumptions - cryptanalysis of some FHE and graded encoding schemes. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 153–178. Springer, Berlin, Heidelberg, August 2016.
3. Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of Learning with Errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.
4. Diego F. Aranha, Carsten Baum, Kristian Gjøsteen, and Tjerand Silde. Verifiable mix-nets and distributed decryption for voting from lattice-based assumptions. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 1467–1481. ACM Press, November 2023.
5. Michele Battagliola, Giacomo Borin, Giovanni Di Crescenzo, Alessio Meneghetti, and Edoardo Persichetti. Enhancing threshold group action signature schemes: Adaptive security and scalability improvements. In Ruben Niederhagen and Markku-Juhani O. Saarinen, editors, *Post-Quantum Cryptography - 16th International Workshop, PQCrypto 2025, Part I*, pages 129–161. Springer, Cham, April 2025.
6. Dan Boneh, Xavier Boyen, and Shai Halevi. Chosen ciphertext secure public key threshold encryption without random oracles. In David Pointcheval, editor, *CT-RSA 2006*, volume 3860 of *LNCS*, pages 226–243. Springer, Berlin, Heidelberg, February 2006.
7. Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. *SIAM J. Comput.*, 36(5):1301–1328, 2007.
8. Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 565–596. Springer, Cham, August 2018.
9. Giacomo Borin, Sofía Celi, Rafael del Pino, Thomas Espitau, Guilhem Niot, and Thomas Prest. Threshold signatures reloaded: ML-DSA and enhanced raccoon with identifiable aborts. Cryptology ePrint Archive, Paper 2025/1166, 2025.
10. Katharina Boudgoust and Peter Scholl. Simple threshold (fully homomorphic) encryption from LWE with polynomial modulus. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part I*, volume 14438 of *LNCS*, pages 371–404. Springer, Singapore, December 2023.
11. Luís T. A. N. Brandão and René Peralta. NIST First Call for Multi-Party Threshold Schemes, 2025.
12. Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 207–222. Springer, Berlin, Heidelberg, May 2004.
13. David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 523–552. Springer, Berlin, Heidelberg, May / June 2010.
14. Arka Rai Choudhuri, Sanjam Garg, Julien Piet, and Guru-Vamsi Policharla. Mempool privacy via batched threshold encryption: Attacks and defenses. In Davide Balzarotti and Wenyuan Xu, editors, *USENIX Security 2024*. USENIX Association, August 2024.
15. Kelong Cong, Daniele Cozzo, Varun Maram, and Nigel P. Smart. Gladius: LWR based efficient hybrid public key encryption with distributed decryption. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 125–155. Springer, Cham, December 2021.

16. Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 103–118. Springer, Berlin, Heidelberg, May 1997.
17. Rafael del Pino, Thomas Espitau, Guilhem Niot, and Thomas Prest. Simple and efficient lattice threshold signatures with identifiable aborts. Cryptology ePrint Archive, Paper 2025/871, 2025.
18. Rafaël Del Pino, Shuichi Katsumata, Mary Maller, Fabrice Mouhartem, Thomas Prest, and Markku-Juhani O. Saarinen. Threshold raccoon: Practical threshold signatures from standard lattice assumptions. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part II*, volume 14652 of *LNCS*, pages 219–248. Springer, Cham, May 2024.
19. Yvo Desmedt, Giovanni Di Crescenzo, and Mike Burmester. Multiplicative non-abelian sharing schemes and their application to threshold cryptography. In Josef Pieprzyk and Reihaneh Safavi-Naini, editors, *ASIACRYPT'94*, volume 917 of *LNCS*, pages 21–32. Springer, Berlin, Heidelberg, November / December 1995.
20. Julien Devevey, Benoît Libert, Khoa Nguyen, Thomas Peters, and Moti Yung. Non-interactive CCA2-secure threshold cryptosystems: Achieving adaptive security in the standard model without pairings. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 659–690. Springer, Cham, May 2021.
21. Léo Ducas and Wessel P. J. van Woerden. NTRU fatigue: How stretched is overstretched? In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 3–32. Springer, Cham, December 2021.
22. Muhammed F. Esgin, Thomas Espitau, Guilhem Niot, Thomas Prest, Amin Sakzad, and Ron Steinfeld. Plover: Masking-friendly hash-and-sign lattice signatures. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VII*, volume 14657 of *LNCS*, pages 316–345. Springer, Cham, May 2024.
23. Eiichiro Fujisaki and Tatsuaki Okamoto. How to enhance the security of public-key encryption at minimum cost. In Hideki Imai and Yuliang Zheng, editors, *PKC'99*, volume 1560 of *LNCS*, pages 53–68. Springer, Berlin, Heidelberg, March 1999.
24. Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Journal of Cryptology*, 26(1):80–101, January 2013.
25. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.
26. Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 341–371. Springer, Cham, November 2017.
27. Patrick Hough, Caroline Sandsbråten, and Tjerand Silde. More efficient lattice-based electronic voting from NTRU. *IACR Communications in Cryptology*, 1(4), 2025.
28. Shuichi Katsumata, Michael Reichle, and Kaoru Takemure. Adaptively secure 5 round threshold signatures from MLWE/MSIS and DL with rewinding. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VII*, volume 14926 of *LNCS*, pages 459–491. Springer, Cham, August 2024.
29. Shuichi Katsumata and Shota Yamada. Partitioning via non-linear polynomial functions: More compact IBEs from ideal lattices and bilinear maps. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 682–712. Springer, Berlin, Heidelberg, December 2016.

30. Duhyeong Kim, Dongwon Lee, Jinyeong Seo, and Yongsoo Song. Toward practical lattice-based proof of knowledge from hint-MLWE. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 549–580. Springer, Cham, August 2023.
31. Paul Kirchner and Pierre-Alain Fouque. Revisiting lattice attacks on overstretched NTRU parameters. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 3–26. Springer, Cham, April / May 2017.
32. Michael Kraitsberg, Yehuda Lindell, Valery Osheter, Nigel P. Smart, and Younes Talibi Alaoui. Adding distributed decryption and key generation to a ring-LWE based CCA encryption scheme. In Julian Jang-Jaccard and Fuchun Guo, editors, *ACISP 19*, volume 11547 of *LNCS*, pages 192–210. Springer, Cham, July 2019.
33. Oleksandra Lapiha and Thomas Prest. A lattice-based IND-CCA threshold KEM from the BCHK+ transform. In Goichiro Hanaoka and Bo-Yin Yang, editors, *ASIACRYPT 2025, Part III*, volume 16247 of *LNCS*, pages 461–494. Springer, Singapore, December 2025.
34. Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In Aggelos Kiayias, editor, *CT-RSA 2011*, volume 6558 of *LNCS*, pages 319–339. Springer, Berlin, Heidelberg, February 2011.
35. Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 738–755. Springer, Berlin, Heidelberg, April 2012.
36. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-LWE cryptography. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 35–54. Springer, Berlin, Heidelberg, May 2013.
37. Daniele Micciancio and Adam Suhl. Simulation-secure threshold PKE from LWE with polynomial modulus. *IACR Commun. Cryptol.*, 1(4):2, 2024.
38. NIST. FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard, 2024.
39. Chris Peikert. An efficient and parallel Gaussian sampler for lattices. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 80–97. Springer, Berlin, Heidelberg, August 2010.
40. Rafaël Del Pino, Shuichi Katsumata, Guilhem Niot, Michael Reichle, and Kaoru Takemure. Unmasking traccoon: A lattice-based threshold signature with an efficient identifiable abort protocol. In *CRYPTO (6)*, volume 16005 of *Lecture Notes in Computer Science*, pages 423–456. Springer, 2025.
41. Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.
42. Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, Damien Stehlé, and Jintai Ding. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.
43. Yang Yu, Huiwen Jia, and Xiaoyun Wang. Compact lattice gadget and its applications to hash-and-sign signatures. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 390–420. Springer, Cham, August 2023.