Naor-Yung Transform for IND-CCA Probing Security with Lattice Instantiations

Katharina Boudgoust, Laurent Imbert, Loïc Masure, and Laz Panard

CNRS, Univ. Montpellier, LIRMM, France
firstname.lastname@lirmm.fr

Abstract. In this work, we propose novel security notions for encryption schemes that combine the power of a typical black-box adversary with additional side-channel access to sensitive algorithms (key generation and decryption). We then extend the Naor-Yung (NY) transform to this setting, enabling generic upgrades from IND-CPA to IND-CCA security under probing attacks. We instantiate this framework with the IND-CPA scheme Rutile and its IND-CCA counterpart Topaz, two lattice-based encryption schemes inspired by Kyber but redesigned with probing security and masking-friendliness in mind. By replacing Kyber's masking-unfriendly components (centered binomial distributions and the FO-transform) with sum-of-uniforms sampling and the NY-transform, we achieve better side-channel resistance by design, resulting in a practical, provably secure encryption scheme that resists both theoretical and side-channel attacks in the probing model.

Keywords: Probing Security · Lattices · Naor-Yung-Transform · M-LWE

1 Introduction

> *In general any channel which can carry information from a secure area to the outside should be studied as a potential risk. [...] Vulnerable algorithms, protocols, and systems need to be revised to incorporate measures to resist timing cryptanalysis and related attacks.*

It is with these words that P. Kocher concluded his seminal article "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS and Other Systems" 30 years ago [Koc96]. At the time, the reaction of the cryptographic community was as significant as Kocher's new threat was unexpected. Constant-time implementation soon became the norm. And with the advent of differential side-channel attacks [KJJ99], advanced protections such as algebraic masking techniques were proposed. In classical public-key cryptography both types of countermeasures are relatively easy to implement. For example, cryptosystems based on the discrete logarithm problem in finite groups – such as the multiplicative subgroup of $\mathbb{Z}/p\mathbb{Z}$ or the group of rational points on an elliptic curve – can be protected by adding a random multiple of the group order to any secret value.

It was around the same time that P. W. Shor proved that the problems underlying group-based cryptography (as well as RSA) can be solved in polynomial time on a quantum computer [Sho94]. After this discovery, cryptographers began to explore quantum-safe cryptographic primitives based on error correcting codes (McEliece [McE78]), euclidean lattices (NTRU [HPS98]) or multivariate systems (HFE [Pat96]). However, active and organized research did not take off until the mid-2000s, when governments and industry began discussing migration to post-quantum cryptography (PQC). In 2016, NIST launched the PQC standardization project. In the original call for proposals and the status reports published after each round, resistance to side-channel attacks was suggested but never considered as a primary objective. For example, after the second round [AASA+20], one could read:

> *NIST hopes to see more and better data for performance in the third round. This performance data will hopefully include implementations that protect against side-channel attacks, such as timing attacks, power monitoring attacks, fault attacks, etc.*

During the third round, the community responded rigorously, proposing a large number of side-channel attacks on post-quantum schemes, along with corresponding countermeasures (see [HPA21] for a comprehensive survey). This phase underscored a critical insight: many foundational components of post-quantum cryptography, while secure against adversaries in standard black-box threat models, could introduce significant vulnerabilities in leaky threat models [UXT+22]. This revelation highlights the urgency of integrating leakage resistance into the design of cryptographic primitives from the outset, rather than treating it as a secondary consideration. Despite this growing awareness, NIST launched an additional call for digital signature proposals in 2022 [oST22], notably omitting resistance to side-channel analysis as a primary selection criterion.[1]

Among the first-round candidates, del Pino, Katsumata, Prest, and Rossi introduced Raccoon [dPKPR24], a digital signature scheme based on structured lattice assumptions. Raccoon stands out as the first scheme to address both provable security (EUF-CMA) and side-channel resistance *by design*, and it does so at a reasonable cost. Its core innovation lies in *masking-friendliness*: the scheme is intentionally crafted to facilitate protection via the *masking* countermeasure [ISW03].

Masking involves encoding sensitive variables using secret sharing schemes and adapting subsequent computations accordingly. A well-designed masking scheme must satisfy *probing security*, i.e. any set of intermediate values smaller than the number of shares remains statistically independent of secret values. This, in turn, provides resilience against more realistic leakage models [DDF14], with a security level scaling exponentially with the number of shares.

However, leakage resilience comes at a cost. While masking linear operations is efficient due to the homomorphic properties of additive secret sharing, masking

[1] The term "side-channel" appears only once in the NIST status report on the first round [ABC+24].

non-linear operations typically incurs a quadratic overhead in runtime, memory, and randomness [ISW03]. Raccoon's key advantage is to use only quasi-linear functions for operations involving shared variables, as well as allowing for some controlled leakage in the generation of small secrets. The authors demonstrated that any attack a probing adversary could mount with leakage access could also be executed with comparable success using only black-box access to (a variant of) Raccoon, [DDF14]. As a result, the overall overhead remains quasi-linear with the number of shares [dPEK+23, Sec. 3.3]. This stands in stark contrast to masked implementations of Dilithium, one of the two lattice-based signatures selected by NIST for standardization. For Dilithium, the complexity of masked implementations scales at least quadratically with the number of shares and involves impractically large constants [CGTZ23]. Thus, Raccoon's masking-friendly design offers a promising path forward, bypassing the challenge of retrofitting side-channel resistance into already-standardized primitives.

Contributions: In this work, we extend the line of work initiated by Raccoon [dPKPR24] on masking-friendly signatures to *masking-friendly encryption schemes*. We introduce Rutile, a lattice-based public-key encryption scheme that achieves both IND-CPA security and resistance to probing attacks *by design*, and its IND-CCA extension Topaz.

Our primary contribution is a design philosophy: we demonstrate that addressing provable security and side-channel resistance simultaneously – rather than incorporating side-channel defenses into existing schemes – leads to more efficient, secure, and practical primitives. This approach avoids the pitfalls of masking complex, legacy components (e.g. FO-transform in Kyber) and instead relies on leakage-resistant building blocks from the ground up. To this end, in Section 3,

- we introduce two novel security notions called *probing* IND-CPA and *probing* IND-CCA, that explicitly account for adversaries with probing access to intermediate computations.
- In order to replace Kyber's FO-transform, we extend the well-known NY-transform to these new security notions. Our main theorem states that if equipped with a fitting NIZK, any *probing* IND-CPA secure encryption scheme can be generically lifted to *probing* IND-CCA.
- As an additional contribution, we extend the NY-transform to non-perfectly correct encryption schemes. This is particularly relevant for lattice-based instantiations, which usually allow for small decryption failures.

In Sections 4-6, we then instantiate the above framework from lattices. In particular, we introduce:

- Rutile: A *probing* IND-CPA-secure encryption scheme based on M-LWE, designed for masking-friendliness. Its security reduces to a new probing M-LWE variant, which we prove equivalent to standard M-LWE.
- Topaz: The first masking-friendly IND-CCA encryption scheme from lattices, built by combining Rutile with a lattice-based NIZK [LNP22].

– We provide 128-bit secure parameters for both schemes using the Lattice Estimator [APS15] and give a quantitative comparison of Topaz with masked Kyber.

In terms of ease of implementation, our schemes inherit the same advantages as Raccoon, not only from an implementation efficiency perspective, but also in terms of security analysis. On the one hand, our error distributions are based on sums of uniform distributions, which makes its implementation straightforward across a wide variety of platforms. Besides, unlike Kyber, our scheme does not require masked implementations of symmetric components so that the number of distinct masking operations to implement is reduced. On the other hand, the masking-friendly parts of our scheme share common building blocks (*gadgets*) with those of Raccoon, in particular the addition of small noisy elements. It turns out that these gadgets have recently been analyzed in the so-called *random* probing model [BRR25], a more realistic variant of the probing model. This suggests that the security of our scheme similarly extends to this more realistic threat scenario.

Future Directions:

Decreasing the Efficiency Gap. We see our proposed encryption schemes and the resulting concrete parameters as a first base line, demonstrating the efficiency loss when replacing the FO-transform by the masking-friendly NY-transform. As we haven't used the full potential of optimizing our schemes, we expect that follow-up works could further reduce the public key and ciphertext sizes. Concretely, performing ciphertext compression would be a relevant first step.

Towards a full Masking Friendliness. Making the noise generation masking-friendly, like in the Raccoon signature scheme, and replacing the FO-transform by a verification proof independent of the secret is a great step towards a masking-friendly public-key encryption scheme. The remaining bottleneck, a non-linear compression operation with quadratic complexity [CGMZ23, CGMZ21, App. D] is now isolated and well identified. In this respect, finding a masking-friendly alternative – *i.e.*, with quasi-linear overhead – to this bottleneck represents a promising track for improvement. Another direction would be to additionally mask the encryption algorithm (which requires masking the LNP NIZK), hence extending Topaz probing security to the message as well.

Probing Security, beyond Encryption. We extended the probing-leakage resilience paradigm of Raccoon to public-key encryption. A potential track for future works could be to extend this paradigm to more advanced primitives. As an example, the Raccoon signature scheme has been extended to threshold signatures [DKM+24]. Given the parallel with our probing-secure encryption scheme, a natural extension could be to explore the potential of our approach for threshold encryption schemes.

Related Works: A notable related work is the lattice-based IND-CCA encryption scheme Polka [HLM+23]. In our work as well as in [HLM+23], leakage resistance was taken into account at the design level. However, if both are proven secure in the black-box model, Polka and Topaz differ in the methodology to achieve leakage resistance. Polka's approach is to look at existing side-channel attacks and to give heuristic arguments (based on novel non-standard lattice problems) how their choices reduce the attack surface. This attack surface has been recently refined in [PPS26a]. In contrast, Topaz has a full formal proof in the probing model. Another relevant direction is the use of the Encrypt-then-Sign framework to avoid the FO-transform and alleviate the impact of chosen-ciphertext side-channel attacks. This was done in [AKSvV22], where an Encrypt-then-Sign authenticated KEM is built in the outsider-security model, targeting applications in which sender authentication is available. It also provides valuable quantitative implementation estimates and heuristic arguments for side-channel security. Our work instead considers ordinary public-key encryption and establishes an end-to-end formal proof in the probing model. We think it would be interesting to combine the insights of [HLM+23], [AKSvV22] with our approach in future works.

Another recent and similar approach to ours is the recent proposal of Pay *et al.* [PPS26b]. The authors also leverage the Naor-Yung transform in order to get leakage resilience of post-quantum encryption in a masking-friendly manner. They are able to prove CCA security with resilience to leakage in decryption, under the assumption that a long-term secret (called selector bit) remains hidden with leakage. This trades off a reduction of the side-channel attack surface against a leak-free assumption that, the authors argue, could be addressed with masking in the (more realistic) probing model, the kind of leakage model we concretely consider in this paper for our security proof.

Besides Raccoon [dPKPR24], other masking-friendly and presumably post-quantum signature schemes have been proposed, namely lattice-based Plover [EEN+24], and masking-friendly variants of the Threshold-Computation-in-the-Head Framework [FRW25] and of the Encrypt-then-Sign Framework [AKSvV22].

In our model, the IND-CCA adversary can probe values during the executions of the key generation and decryption algorithms. We can thus see it as *computation leakage*. A different model is to provide the adversary with a separate leakage oracle on the secret key, for arbitrary (but bounded) leakage functions, as studied in [AGV09, NS09]. This can be categorized as *memory leakage* (as for instance done in this survey on leakage-resilient cryptography [KR19]). In an analogue manner to our work, it was shown in [NS09] that one can use the Naor-Yung transform to lift IND-CPA to IND-CCA security in the memory leakage model. Note that the impact of computation leakage on the NY-transform is more important than the impact of memory leakage, as in the computation leakage case the adversary is allowed to probe the additional decryption oracle.

2 Preliminaries

In this work, we always assume the adversary $\mathcal{A}$ to be stateful, that is, they maintain an internal state throughout different phases of the security game.

For $d \in \mathbb{N}^*$, we note $[d] = \{1, \ldots, d\}$. For $x \in \mathbb{R}$, we note $\lfloor x \rceil$ the nearest integer to x, half rounded up. Let $i, j \in (\mathbb{N}^*)^2$, we note $\mathbf{0}_{i,j}$ the $i \times j$ zero matrix. For $n = 2^\ell > 1$, we denote by $\mathcal{R}$ the ring $\mathbb{Z}[X]/\langle X^n + 1\rangle$, and for modulus $q \in \mathbb{N}^*$, we denote by $\mathbb{Z}_q$ the ring $\mathbb{Z}/q\mathbb{Z}$, and by $\mathcal{R}_q$ the ring $\mathbb{Z}_q[X]/\langle X^n + 1\rangle$. For $f = \sum_{i=0}^{n-1} f_i X^i \in \mathcal{R}$, we note the l_∞-norm $\|f\|_\infty = \max_{i\in[n]}(|f_i|)$ and the l_2-norm $\|f\|_2 = \sqrt{\sum_{i\in[n]} f_i^2}$. For $k \in \mathbb{N}^*$ and $\mathbf{f} = (f_1, ..., f_k) \in \mathcal{R}^k$, we note the l_∞-norm $\|\mathbf{f}\|_\infty = \max_{i\in[k]}(\|f_i\|_\infty)$ and the l_2-norm $\|\mathbf{f}\|_2 = \sqrt{\sum_{i\in[k]} \|f_i\|_2^2}$. Both for f and $\mathbf{f}$ we sometimes write $\|\cdot\|$ instead of $\|\cdot\|_2$. When writing the norm of an element in $\mathcal{R}_q$, we implicitly take the centered representation modulo q, interpret it as element in $\mathcal{R}$ and then compute the norm. We provide a proof in Appendix A.2 for completeness.

Lemma 1. *Let $k \in \mathbb{N}^*$ and $\mathbf{a}, \mathbf{b} \in (\mathcal{R}^k)^2$, we have $\|\mathbf{a}^t\mathbf{b}\|_\infty \leq \|\mathbf{a}\|_2 \cdot \|\mathbf{b}\|_2$.*

For $u \in \mathbb{N}^*$, $\mathsf{Unif}(u)$ denotes the uniform distribution over the set $\{-2^{u-1}, \ldots, 2^{u-1}-1\}$. By interpreting every element in $\mathcal{R}$ as a polynomial of degree less than n with coefficients in $\mathbb{Z}$, for $k \in \mathbb{N}^*$, we denote by $\mathsf{Unif}(\mathcal{R}, k, u)$ the distribution over $\mathcal{R}^k$ which samples vectors of k polynomials whose n coefficients are independently sampled from $\mathsf{Unif}(u)$. Put differently, $\mathsf{Unif}(\mathcal{R}, k, u) = \mathsf{Unif}(u)^{n\cdot k}$. By $\mathsf{NSUnif}(u, l)$ we denote the (non-zero-centered) distribution where l elements are sampled independently and identically from $\mathsf{Unif}(u)$ and summed together. For a random variable $X \sim \mathsf{NSUnif}(u, l)$, one can check that $X + l/2$ is sub-Gaussian for $\sigma^2 = 2^{2u}\, l/6$. By $\mathsf{SUnif}(u, l) = \mathsf{NSUnif}(u, l) + l/2$ we denote the sum of uniform distribution which is now zero-centered, while preserving the variance. We extend it in the same way over $\mathcal{R}^k$, denoting $\mathsf{NSUnif}(\mathcal{R}, k, u, l)$ the sum of l copies of $\mathsf{Unif}(\mathcal{R}, k, u)$. Similarly, we define $\mathsf{SUnif}(\mathcal{R}, k, u, l) = \mathsf{NSUnif}(\mathcal{R}, k, u, l) + l/2$, where by abuse of notation, the latter means the vector of polynomials whose coefficients are all $l/2$. Note that $\mathsf{SUnif}(\mathcal{R}, k, u, l) = \mathsf{SUnif}(u, l)^{n\cdot k}$.

The tail bound from [LPR13, Lem. 2.2] applies to $\mathbf{x} \leftarrow \mathsf{SUnif}(\mathcal{R}, k, u, l)$, thus:

$$\Pr\left[\|\mathbf{x}\|_2 > 2^u\sqrt{3lnk}\right] \leq \exp\left(-2nk\right) \tag{1}$$

For two distributions χ and ψ, we note $\chi \overset{i.d.}{=} \psi$ when the two distributions are identically distributed. This notation naturally extends to random variables that are identically distributed, as well as random functions. We write $x \leftarrow \chi$ when x is a value sampled according to χ and $x \leftarrow\!\!\$\ S$ when x is sampled uniformly at random in the finite set S.

Let f be a random function, we note $x \leftarrow f(\cdot)$ when x is a random output of f. To explicit the random coins R used in f, we note $x := f(\cdot; R)$, where $:=$ highlights that x is deterministically computed once the randomness R is fixed.

2.1 The (Probing) M-LWE Problem

The security of our scheme relies on the Module Learning With Errors (M-LWE) problem, as introduced in [BGV12, LS15]. In this paper, we are interested in M-LWE instances where both the secret and the noise are sampled as sums of small uniforms. With a suitable parameter selection, for instance derived using the Lattice Estimator [APS15], this is considered a standard hardness assumption in lattice-based cryptography. Putting additional parameter restrictions, its hardness can be derived from worst-case problems on module lattices [BJRW23, BJTW26].

In Definition 1, we introduce a generalization of M-LWE, where up to $d-1$ summands of the secret and noise are provided to the adversary. The case of $d = 1$ corresponds to the standard M-LWE problem. We call this new formulation the *(d-1)-probing* M-LWE problem and show that it is equivalent to standard M-LWE where secret and error follow a (smaller) sum of uniform distribution. For simplicity, we stick to power-of-two cyclotomic rings and square matrices in this section, but everything generalizes to arbitrary cyclotomic rings and rectangular matrices as well. In this section, we only use the word "probes" in an informal manner, in contrast to the formal definition of the term in Section 2.3.

Definition 1 (Probing M-LWE). *Let $n \in \mathbb{N}^*$ be a power of two and $q \in \mathbb{N}^*$ be a modulus, defining the ring $\mathcal{R}_q$. Let $l \in \mathbb{N}^*$ be the number of small uniforms to sum, $u \in \mathbb{N}^*$ their size parameter and $k \in \mathbb{N}^*$ the dimensions of the matrix over $\mathcal{R}_q$. Further, let $d-1 \in \mathbb{N}$ with $l > d-1$, the number of "probes" allowed to the attacker. The (d-1)-probing* M-LWE$_{\mathcal{R}_q,d,k,l,u}$ *experiment is defined in Figure 1. For an adversary $\mathcal{A}$, the respective advantage is defined as*

$$\mathsf{Adv}(\mathcal{A}) = \left|\Pr[\textit{(d-1)-probing } \text{M-LWE}_{\mathcal{R}_q,d,k,l,u}(\mathcal{A}) = 1] - \frac{1}{2}\right|$$

If $d = 1$, we recover standard M-LWE.

Proposition 1 shows that *(d-1)-probing* M-LWE is equivalent to M-LWE with a reduced number of summed small uniforms. The proof only uses the fact that the secret and error distributions of M-LWE are sums of *independently and identically distributed* samples of the same distribution. In our case, this distribution is the uniform distribution, as it is suitable in the additive masking setting, but it can easily be exchanged with other distributions if needed. The proof essentially uses the linearity of M-LWE and can be found in Appendix A.3.

Proposition 1. *Let $n \in \mathbb{N}^*$ be a power of two and $q \in \mathbb{N}^*$ be a modulus, defining the ring $\mathcal{R}_q$. Let $l \in \mathbb{N}^*$ be the number of small uniforms to sum, $u \in \mathbb{N}^*$ their size parameter, $k \in \mathbb{N}^*$ the dimension of matrix and $d \in \mathbb{N}^*$ the number of "probes" plus one allowed to the attacker, with $l > d-1$. Then,*

$$\text{M-LWE}_{\mathcal{R}_q,k,(l-d+1),u} \Leftrightarrow \textit{(d-1)-probing } \text{M-LWE}_{\mathcal{R}_q,d,k,l,u}$$

More precisely, there exists an adversary $\mathcal{A}$ against M-LWE$_{\mathcal{R}_q,k,(l-d+1),u}$ *with advantage* Adv *if and only if there exists an adversary $\mathcal{B}$ against the problem (d-1)-probing* M-LWE$_{\mathcal{R}_q,d,k,l,u}$ *with the same advantage* Adv.

(d-1)-probing $\text{M-LWE}_{\mathcal{R}_q,d,k,l,u}(\mathcal{A})$

$\mathbf{A} \leftarrow\!\!\$\ \mathcal{R}_q^{k\times k}$
$\{1,\ldots,l\} \supsetneq \overline{\mathcal{P}} \leftarrow \mathcal{A}(\mathbf{A})$
assert $\left|\overline{\mathcal{P}}\right| < d$
for $i \in \{1,\ldots,l\}$
  $\boldsymbol{\rho}_i^{(s)}, \boldsymbol{\rho}_i^{(e)} \leftarrow\!\!\$\ (\mathsf{Unif}(\mathcal{R},k,u))^2$
$\mathcal{L} = (\boldsymbol{\rho}_i^{(s)}, \boldsymbol{\rho}_i^{(e)})_{i\in\overline{\mathcal{P}}}$
$b \leftarrow\!\!\$\ \{0,1\}$
if $b = 0$:
  $\mathbf{s} = \sum_{i\in[l]} \boldsymbol{\rho}_i^{(s)},\ \mathbf{e} = \sum_{i\in[l]} \boldsymbol{\rho}_i^{(e)}$
  $\mathbf{b} := \mathbf{A}\mathbf{s} + \mathbf{e} \bmod q$
else :
  $\mathbf{b} \leftarrow\!\!\$\ \mathcal{R}_q^k$
$b' \leftarrow\!\!\$\ \mathcal{A}(\mathbf{b}, \mathcal{L})$
return $b = b'$

Fig. 1: The *(d-1)-probing* M-LWE experiment

2.2 Cryptographic Primitives

We recall the relevant cryptographic definitions.

Definition 2 (PKE). *A public-key encryption scheme* PKE *over message space* $\mathcal{M}$ *is defined by the following* PPT *algorithms* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$*, parameterized by the security parameter* κ*, where:*

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$*: on input the security parameter* κ*, the key generation algorithm outputs a public key* pk *and a secret key* sk*.*
- $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$*: on input a public key* pk *and a plaintext* $m \in \mathcal{M}$*, the encryption algorithm outputs a ciphertext* ct*.*
- $m' \leftarrow \mathsf{Dec}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct})$*: given a public key* pk*, a secret key* sk *and a ciphertext* ct*, the decryption algorithm outputs a plaintext* $m' \in \mathcal{M}$ *or the symbol* $\perp$*.*

To make the random coins used in Enc explicit, we write $\mathsf{ct} := \mathsf{Enc}(\mathsf{pk}, m; R)$, with R the random coins sampled from the randomness distribution RandDistr.

We further use the primitive of one-time simulation sound NIZK's. We assume that the prover and verifier both have access to the same random oracle during the execution of the protocol. We say "proof", even though strictly speaking we are talking about "arguments".

Definition 3 (NARG). *Let* H *be a hash function modeled as a random oracle. For an algorithm* A*, we denote by* A^H *the fact that the algorithm has access to the random oracle* H*. A* non-interactive argument system (NARG) *for a relation* R *consists of a pair of probabilistic algorithms* $\Pi = (\mathsf{P}, \mathsf{V})$*, where*

– $\mathsf{P}^{\mathsf{H}}(\mathsf{x}, \mathsf{w}) \to \pi$*: The prover algorithm takes as input a statement witness pair* $(\mathsf{x}, \mathsf{w}) \in \mathsf{R}$*, and outputs a proof* π*.*
– $\mathsf{V}^{\mathsf{H}}(\mathsf{x}, \pi) \to b \in \{0, 1\}$*: The verifier algorithm takes as input a statement* x *and proof* π*, and outputs accept* ($b = 1$) *or reject* ($b = 0$)*.*

We denote by $\mathsf{L} = \{\mathsf{x} \colon \exists \mathsf{w}, (\mathsf{x}, \mathsf{w}) \in \mathsf{R}\}$ the language induced by R.

Definition 4 (Completeness). *Let* $\epsilon_{\mathsf{complete}}(\kappa)$ *be a function parameterized by the security parameter* κ *taking real values in* $[0, 1]$*. A* NARG $\Pi = (\mathsf{P}, \mathsf{V})$ *for the relation* R *and with associated random oracle* H *is called* $\epsilon_{\mathsf{complete}}(\kappa)$*-complete if for all* $k \in \mathbb{N}^*$*, all* $(\mathsf{x}, \mathsf{w}) \in \mathsf{R}$ *it holds*

$$\Pr\left[\mathsf{V}^{\mathsf{H}}(\mathsf{x}, \mathsf{P}^{\mathsf{H}}(\mathsf{x}, \mathsf{w})) \neq 1\right] \leq \epsilon_{\mathsf{complete}}(\kappa)$$

where the probability is taken over the randomness of P*,* V *and the random oracle* H*.*

Definition 5 (Zero-Knowledge). *Let* $\epsilon_{\mathsf{ZK}}(\kappa)$ *be a function parameterized by the security parameter* κ *taking real values in* $[0, 1]$*. A* NARG $\Pi = (\mathsf{P}, \mathsf{V})$ *for the relation* R *and with associated random oracle* H *is called* $\epsilon_{\mathsf{ZK}}(\kappa)$*-zero-knowledge if there exists a* PPT *simulator* $\mathsf{Sim} = (\mathsf{Sim}_{\mathsf{H}}, \mathsf{Sim}_{\pi})$ *such that for all* $\kappa \in \mathbb{N}$ *and for all* PPT *adversaries* $\mathcal{A}$ *it holds that*

$$\left|\Pr\left[\mathsf{Expt}^{\mathsf{ZK\text{-}Real}}_{\mathcal{A}}(1^{\kappa}) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{ZK\text{-}Sim}}_{\mathcal{A},\mathsf{Sim}}(1^{\kappa}) = 1\right]\right| \leq \epsilon_{\mathsf{ZK}}(\kappa)$$

where the probability is taken over the random coins of the adversary, the random oracle H *and the experiments defined in Figure 2a.*

Definition 6 (One-Time Simulation Soundness). *Let* $\epsilon_{\mathsf{OTSS}}(\kappa)$ *be a function parameterized by the security parameter* κ *taking real values in* $[0, 1]$ *and let* $\mathsf{Sim} = (\mathsf{Sim}_{\mathsf{H}}, \mathsf{Sim}_{\pi})$ *be a zero-knowledge simulator. A* NARG $\Pi = (\mathsf{P}, \mathsf{V})$ *for the relation* R *and with associated random oracle* H *is called* $\epsilon_{\mathsf{OTSS}}(\kappa)$*-one-time simulation sound if for every* $\kappa \in \mathbb{N}$*, for all* PPT *adversaries* $\mathcal{A}$*, it holds that*

$$\Pr\left[\mathsf{Expt}^{\mathsf{OTSS}}_{\mathcal{A},\mathsf{Sim}}(1^{\kappa}) = 1\right] \leq \epsilon_{\mathsf{OTSS}}(\kappa)$$

where the probability is taken over the random coins of the adversary and the experiments defined in Figure 2b.

We denote by OTSS-NIZK a NARG that is $\epsilon_{\mathsf{complete}}(\kappa)$-complete, $\epsilon_{\mathsf{ZK}}(\kappa)$-zero-knowledge and $\epsilon_{\mathsf{OTSS}}(\kappa)$-one-time simulation sound, where all three functions are negligible in κ. Sometimes we only write NIZK for simplicity.

2.3 Probing Theory

Wires, Values, Programs and Probes. We recall that an *arithmetic circuit* is a direct acyclic graph that contains addition and multiplication nodes (or gates) as well as input and output nodes. We assume that the arithmetic nodes $(+, \times)$

$\mathsf{Expt}^{\mathsf{ZK\text{-}Real}}_{\mathcal{A}}(1^\kappa)$

1 : $(\mathsf{x},\mathsf{w}) \leftarrow \mathcal{A}^{\mathsf{H}}(1^\kappa)$
2 : **if** $(\mathsf{x},\mathsf{w}) \notin \mathsf{R}$
3 : **return** $\bot$
4 : $\pi \leftarrow \mathsf{P}(\mathsf{x},\mathsf{w})$
5 : $b \leftarrow \mathcal{A}(\pi)$
6 : **return** b

$\mathsf{Expt}^{\mathsf{ZK\text{-}Sim}}_{\mathcal{A},\mathsf{Sim}}(1^\kappa)$

1 : $(\mathsf{x},\mathsf{w}) \leftarrow \mathcal{A}^{\mathsf{Sim}_{\mathsf{H}}}(1^\kappa)$
2 : **if** $(\mathsf{x},\mathsf{w}) \notin \mathsf{R}$
3 : **return** $\bot$
4 : $\pi \leftarrow \mathsf{Sim}_\pi(\mathsf{x})$
5 : $b \leftarrow \mathcal{A}(\pi)$
6 : **return** b

(a) Zero-knowledge

$\mathsf{Expt}^{\mathsf{OTSS}}_{\mathcal{A},\mathsf{Sim}}(1^\kappa)$

1 : $\mathsf{x} \leftarrow \mathcal{A}^{\mathsf{Sim}_{\mathsf{H}}}(1^\kappa)$
2 : $\pi \leftarrow \mathsf{Sim}_\pi(\mathsf{x})$
3 : $(\mathsf{x}',\pi') \leftarrow \mathcal{A}^{\mathsf{Sim}_{\mathsf{H}}}(\pi)$
4 : $b = \mathsf{V}(\mathsf{x}',\pi') \wedge (\mathsf{x}' \notin \mathsf{L})$
5 : $\wedge\, ((\mathsf{x}',\pi') \neq (\mathsf{x},\pi))$
6 : **return** b

(b) One-time simulation soundness

Fig. 2: Security experiments of non-interactive proof systems.

have input degree 2 and output degree 1. Input nodes have input degree zero and output nodes have output degree zero. These nodes are connected by edges or *wires* that we note with overhead letters (e.g. $\overline{x}$). For an execution of the algorithm, the *value* carried by the wire labeled $\overline{x}$ is simply denoted x. These notations naturally extend to a set of wires (e.g. $\overline{\mathcal{X}}$) and its corresponding tuple of values $\mathcal{X}$.

In our work, we use the notation $[\![x]\!] = (x_i)_{i\in[d]}$ to denote masking via additive secret sharing, where the values x_i belong to a group equipped with addition. That is, $x_1, \ldots, x_{d-1}$ are sampled uniformly at random in the group and $x_d = x - \sum_{i\in[d-1]} x_i$. We use the following privacy property of additive secret sharing: Let $I \subsetneq [d]$ be any index subset of size at most $d-1$ and $[\![x]\!] = (x_i)_{i\in[d]}$ be the masking of an arbitrary value x over some group G. Then, the distribution $((x_i)_{i\in I}, x)$ is identical to the distribution $((y_i)_{i\in I}, x)$, where $y_i \leftarrow G$.

We denote by $\mathsf{KeyUpdate}$ a function which takes as input a d-sharing $[\![K]\!] = (K_i)_{i\in[d]}$ of some value K and outputs another d-sharing of the same value $[\![K]\!] = (K'_i)_{i\in[d]}$ such that for every two sets of indices $I_1, I_2 \subsetneq [d]$, we have that $(K_i)_{i\in I_1}$ and $(K'_i)_{i\in I_2}$ are independent random variables. This implies that having access to at most $d-1$ values of $(K_i)_{i\in[d]}$ does not leak any information about $(K'_i)_{i\in[d]}$ nor about K.

Likewise, for an arithmetic circuit which takes as input a masked value $[\![x]\!]$, we denote $[\![\overline{x}]\!] = (\overline{x_i})_{i\in[d]}$ the d-tuple of input wires carrying the d shares of $[\![x]\!]$.

Definition 7 (Randomized Arithmetic Circuit). *A randomized arithmetic circuit is an arithmetic circuit which also contains nodes with input degree zero and output degree one that output random values.*

Definition 8 (Program). *We define a program* Prog *as a randomized arithmetic circuit, and denote by* $\overline{\mathcal{W}}_{\mathsf{Prog}}$ *(or* $\overline{\mathcal{W}}$ *when clear from context) its set of wires. The set of input wires is partitioned into* $\overline{\mathcal{X}}_{masked}$ *whose wires carry sharings of secret values, and* $\overline{\mathcal{X}}_{unmasked}$ *whose wires carry unmasked secret values. We have* $\overline{\mathcal{X}}_{\mathsf{Prog}} = \overline{\mathcal{X}}_{masked} \cup \overline{\mathcal{X}}_{unmasked} \subset \overline{\mathcal{W}}_{\mathsf{Prog}}$*. Similarly, we denote by* $\overline{\mathcal{Y}} = (\overline{\mathcal{Y}}_{public}, \overline{\mathcal{Y}}_{secret})$ *the mutually disjoint sets of respectively public and secret*

(masked or unmasked) output wires. We define $\mathcal{X}$ and $\mathcal{Y}$ the sets of input and output values such that $\mathcal{Y} \leftarrow \mathsf{Prog}(\mathcal{X})$.

We note that in the above definition, public inputs are treated as implicit parameters of the program.

We now introduce the notion of probes on a program. On a high level, the notion of probes formalizes the idea that an attacker retrieves a number of intermediate values during the execution of the program, thus allowing for a potential leakage of secret values. In the so-called probing model, a *probe* is the label of an algorithm's wire targeted by the attacker, and a *leakage* is the value carried by this wire that is thus provided to the attacker.

Definition 9 (Probes). *Let* Prog *be a program as in Def. 8, with set of input wires $\overline{\mathcal{X}} = (\overline{\mathcal{X}}_{masked}, \overline{\mathcal{X}}_{unmasked})$. A set of probes is a subset $\overline{\mathcal{P}}$ of* Prog*'s wires. Given a set of probes $\overline{\mathcal{P}}$ and $\mathcal{X}$ the values carried by the program's input wires, we note $\mathsf{Prog}^{\mathfrak{L}}(\mathcal{X}, \overline{\mathcal{P}})$ the joint distribution*

$$(\mathcal{Y}, \mathcal{L}) \leftarrow \mathsf{Prog}^{\mathfrak{L}}(\mathcal{X}, \overline{\mathcal{P}})$$

where $\mathcal{Y} = \mathcal{Y}_{public} \cup \mathcal{Y}_{secret}$ denotes the public and secret outputs of $\mathsf{Prog}(\mathcal{X})$ for some internal random coins, and where the leakage $\mathcal{L}$ is the set of values carried by the wires in $\overline{\mathcal{P}}$ for these same random coins.

Simulatability and Non-Interference Properties. We now introduce the notions of non-interference without unmasked inputs [BBD+16] and with unmasked inputs [dPKPR24].

Definition 10 (NI/NIu property, [BBD+16, dPKPR24]). *Let $d \in \mathbb{N}^*$ and* Prog *be a program with input wires $\overline{\mathcal{X}} = (\overline{\mathcal{X}}_{masked}, \overline{\mathcal{X}}_{unmasked})$. We say that* Prog *is* (d-1)*-Non-Interfering with Unshared inputs (*(d-1)-NIu *for short) if and only if there exist two PPT simulators* SimProb *and* SimLeak *such that for any input values $\mathcal{X}$ and any set of probes $\overline{\mathcal{P}}$ with $|\overline{\mathcal{P}}| < d$, we have that:*

- $\mathsf{SimProb}(\overline{\mathcal{P}})$ *outputs $\overline{\mathcal{X}}' = (\overline{\mathcal{X}}'_{masked}, \overline{\mathcal{X}}'_{unmasked})$ where:*
  - $\overline{\mathcal{X}}'_{masked} \subset \overline{\mathcal{X}}_{masked}$ *and* $\overline{\mathcal{X}}'_{unmasked} \subset \overline{\mathcal{X}}_{unmasked}$
  - *for any tuple of wires of a masked variable $[\![\overline{x}]\!]$, $|[\![\overline{x}]\!] \cap \overline{\mathcal{X}}'_{masked}| \leq |\overline{\mathcal{P}}|$*
  - $|\overline{\mathcal{X}}'_{unmasked}| \leq |\overline{\mathcal{P}}|$
- $\mathsf{SimLeak}(\mathcal{X}', \overline{\mathcal{P}})$ *output's distribution is identical to that of $\mathcal{L}$, where $\mathcal{X}' \subset \mathcal{X}$ is the set of values carried by the wires in $\overline{\mathcal{X}}'$ and $(\mathcal{Y}, \mathcal{L}) \leftarrow \mathsf{Prog}^{\mathfrak{L}}(\mathcal{X}, \overline{\mathcal{P}})$*

When Prog *has no unmasked secret input (i.e. $\overline{\mathcal{X}}_{unmasked} = \emptyset$), we simply call it* (d-1)*-Non-Interfering (*(d-1)-NI *for short).*

In general, $\overline{\mathcal{X}}'$ is a strict subset of $\overline{\mathcal{X}}$ as masking will be done on level d, but the simulator is only allowed to simulate using less than d wires.

We now introduce non-interference with public outputs properties where the public output of the program is set and not random as in definitions above. This notion is introduced in the random probing model in [BRR25, Def. 5]. We reformulate this definition and extend it to the d-probing model.

Definition 11 (NIo/NIuo **property, [BRR25]).** *Let* $d \in \mathbb{N}^*$ *and* Prog *be a program with input wires* $\overline{\mathcal{X}} = (\overline{\mathcal{X}}_{masked}, \overline{\mathcal{X}}_{unmasked})$ *and public output wires* $\overline{\mathcal{Y}}_{public}$*. We say that* Prog *is* (d-1)*-Non-Interfering with Unshared inputs and public Outputs (*(d-1)-NIuo *for short) if and only if there exists two PPT simulators* SimProb *and* SimLeak *such that for any input values* $\mathcal{X}$ *and public output values* $\mathcal{Y}_{public}$ *for which* $\Pr(\overline{\mathcal{X}} = \mathcal{X}, \overline{\mathcal{Y}}_{public} = \mathcal{Y}_{public}) \neq 0$*, and any set of probes* $\overline{\mathcal{P}}$ *where* $|\overline{\mathcal{P}}| < d$*,* SimProb *and* SimLeak *meets the same criteria than in Def. 10 when* SimLeak *is given access to* $\mathcal{Y}_{public}$*.*

When $\overline{\mathcal{X}}_{unmasked} = \emptyset$*, we simply call* Prog (d-1)*-Non-Interfering with public Outputs (*(d-1)-NIo *for short).*

We now introduce a useful result for composition of a NIu program with a NIo program whose public output is deterministic in the NIu gadget input.

Proposition 2 (NIu+NIo = NIuo)**.** *Let* $d \in \mathbb{N}^*$*,* Prog_1 *a* (d-1)-NIu *program with a single d-shared masked input variable* $\overline{\mathcal{X}}_1 = (\overline{[\![x_1]\!]}, \overline{\mathcal{X}}_{unmasked})$*, no public output and a single d-shared masked variable as secret output* $\overline{\mathcal{Y}}_1 = (\emptyset, \overline{[\![y]\!]})$ *and* Prog_2 *a* (d-1)-NIo *program with a single d-shared masked input variable* $\overline{\mathcal{X}}_2 = (\overline{[\![x_2]\!]}, \emptyset)$ *and output* $\overline{\mathcal{Y}_2} = (\overline{\mathcal{Y}_{public}}, \emptyset)$*. If there exist a deterministic function* $f : \mathcal{R}^{|\overline{\mathcal{X}}_1|} \to \mathcal{R}^{|\overline{\mathcal{Y}_2}|}$ *such that for every input values* $\mathcal{X}$ *carried by* $\overline{\mathcal{X}}$ *we have* $\mathcal{Y}_2 = f(\mathcal{X})$ *with* $\mathcal{Y}_2$ *the output values carried by* $\overline{\mathcal{Y}_2}$ *wires, then* Prog *is* (d-1)-NIuo *where, for every input* $\mathcal{X} = ([\![x]\!], \mathcal{X}_{unmasked})$*,*

$$\mathsf{Prog}([\![x]\!], \mathcal{X}_{unmasked}) := \mathsf{Prog}_2(\mathsf{Prog}_1([\![x]\!], \mathcal{X}_{unmasked}))$$

Proof Sketch. Without loss of generality we assume there are d_2 probes on Prog_2 and d_1 probes on Prog_1 with $d_1 + d_2 < d$. The overall idea of our proof is that the d_2 probes on Prog_2 may be simulated with values carried by up to d_1 wires of $\overline{\mathcal{X}}_2$, which can in turn be simulated along with the d_1 probes on Prog_1 — i.e. simulating less than d values — by Prog_1's NIu simulator. A full proof is provided in Appendix A.4.

Simulating with public output only The following result shows that the leakage of a *(d-1)*-NIo program can be simulated by a simulator which only takes the set of probes and public output values as input. The proof essentially leverages the privacy property of the additive secret sharing used for the masked values. For simplicity, we show our result in the case of one single masked value, but it easily generalizes with some additional notational overhead to multiple masked input values. We do not claim this to be a novel observation, but we were not able to find a result in the literature which exactly proves what we need. Hence, we provide a proof for completeness in Appendix A.5.

Proposition 3. *For* $d \in \mathbb{N}^*$*, let* Prog *be a* (d-1)-NIo *program with no unmasked input (i.e.* $\overline{\mathcal{X}}_{unmasked} = \emptyset$*), only one masked input* $\overline{\mathcal{X}}_{masked} = \overline{[\![x]\!]}$ *with d wires and such that the public output* $\mathcal{Y}_{public}$ *of* Prog *is uniquely determined by its public input and the value for which* $\overline{[\![x]\!]}$ *carries the d-sharing (and thus not*

by its concrete sharing). There exists a simulator Sim *such that for every set of probes* $\overline{\mathcal{P}}$ *s.t.* $\left|\overline{\mathcal{P}}\right| < d$*, for any* $x \in \mathcal{R}$ *and any privacy-preserving additive sharing* $[\![x]\!]$ *of* x *we have that*

$$\mathcal{L} \overset{i.d.}{=} \mathsf{Sim}(\mathcal{Y}_{public}, \overline{\mathcal{P}}) \text{ where } (\mathcal{Y}_{public}, \mathcal{Y}_{secret}, \mathcal{L}) \leftarrow \mathsf{Prog}^{\mathfrak{L}}(\mathsf{KeyUpdate}([\![x]\!]), \overline{\mathcal{P}})$$

3 Semantic Probing Security for PKE

In this section, we propose new security notions for public-key encryption (PKE) schemes. They can be seen as a combination of (black-box) cryptographic semantic and (grey-box) probing security notions, which are usually considered separately.

3.1 Probing IND-CPA Security

We introduce the notion of *(d-1)-probing* IND-CPA security, which is similar to the traditional IND-CPA notion except it permits the adversary to ask for probes $\overline{\mathcal{P}}_{\mathrm{KG}}$ on the KeyGen procedure, providing to the adversary up to $d-1$ values $\mathcal{L}_{\mathrm{KG}}$ of the procedure's execution. Recall from Section 2.3 that for a program Prog with input values $\mathcal{X}$ and probes $\overline{\mathcal{P}}$, we denote by $(\mathcal{Y}, \mathcal{L}) \leftarrow \mathsf{Prog}^{\mathfrak{L}}(\mathcal{X}, \overline{\mathcal{P}})$ the pair of output $\mathcal{Y}$ and leakage $\mathcal{L}$. As side-channel attacks are costly we only consider attacks on long-term secrets (i.e. the secret key). Hence we do not consider probes on Enc (as in [BC22, BGR+21]).

(d-1)-probing IND-CPA for PKE, with $b \in \{0,1\}$

1 : $\overline{\mathcal{P}}_{\mathrm{KG}} \leftarrow \mathcal{A}(1^{\kappa})$
2 : $(\mathsf{pk}, [\![\mathsf{sk}]\!], \mathcal{L}_{\mathrm{KG}}) \leftarrow \mathsf{KeyGen}^{\mathfrak{L}}(1^{\kappa}, \overline{\mathcal{P}}_{\mathrm{KG}})$
3 : $(m_0, m_1) \leftarrow \mathcal{A}(\mathsf{pk}, \mathcal{L}_{\mathrm{KG}})$
4 : $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$
5 : $b^* \leftarrow \mathcal{A}(\mathsf{ct}^*)$
6 : **return** b^*

$$\mathsf{WIN}^{\mathsf{IND\text{-}CPA}}(b) = (b = b^*) \wedge (|\overline{\mathcal{P}}_{\mathrm{KG}}| \leq d-1)$$

Fig. 3: *(d-1)-probing* IND-CPA security game for PKE

Definition 12. *A* PKE *scheme is called (d-1)-probing* IND-CPA *secure if it holds that for all security parameters* $\kappa \in \mathbb{N}$ *and all* PPT *adversaries* $\mathcal{A}$ *in the game described in Figure 3 we have*

$$\mathsf{Adv}^{\mathsf{dCPA}}_{\mathcal{A}}(\kappa) := \left|\Pr\left[\mathsf{WIN}^{\mathsf{IND\text{-}CPA}}(0)\right] - \Pr\left[\mathsf{WIN}^{\mathsf{IND\text{-}CPA}}(1)\right]\right| \leq \mathsf{negl}(\kappa)$$

We note that this definition coincide with the usual IND-CPA security definition for PKE when $d = 1$, that is, when no probing is allowed.

3.2 Probing IND-CCA Security

Similarly, we now introduce a probing version of the IND-CCA security notion, named *(d-1)-probing* IND-CCA. Strictly speaking, we consider IND-CCA2 security, but omit this nuance for simplicity. We proceed as in the previous security definition except here we also allow for probes on the Dec procedure. More precisely, when querying (up to a polynomial number of times) the decryption oracle on ciphertext ct, the adversary can additionally ask for probes $\overline{\mathcal{P}}$. As an answer, they receive the decrypted message m together with the leakage $\mathcal{L}$.

As done in the context of masked signatures [BBE+18, dPKPR24], we assume the existence of an algorithm called KeyUpdate (as formalized in Section 2.3) which refreshes the sharing of the secret key between two decryption queries. We further assume that the attacker cannot probe this algorithm. The purpose of this algorithm is to make sure that the adversary cannot learn more than $d - 1$ shares of the secret key across different decryption queries. In real life settings, it is often instantiated with refresh gadgets. Recent work develops new techniques for securely composing masking gadgets with IOS refresh gadgets [BBM26, Sec. 4], similarly to what KeyUpdate does. This suggests a possible route for extending the composition analysis to our setting in future works.

(d-1)-probing IND-CCA for PKE, with $b \in \{0, 1\}$

1 : $\overline{\mathcal{P}}_{\mathrm{KG}} \leftarrow \mathcal{A}(1^\kappa)$
2 : $(\mathsf{pk}, [\![\mathsf{sk}]\!], \mathcal{L}_{\mathrm{KG}}) \leftarrow \mathsf{KeyGen}^{\mathfrak{L}}(1^\kappa, \overline{\mathcal{P}}_{\mathrm{KG}})$
3 : $(m_0, m_1) \leftarrow \mathcal{A}^{\mathsf{DEC}_{[\![\mathsf{sk}]\!],\emptyset}}(\mathsf{pk}, \mathcal{L}_{\mathrm{KG}})$
4 : $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$
5 : $b^* \leftarrow \mathcal{A}^{\mathsf{DEC}_{[\![\mathsf{sk}]\!],\mathsf{ct}^*}}(\mathsf{ct}^*)$
6 : **return** b^*

$\mathsf{DEC}_{[\![\mathsf{sk}]\!],\mathsf{ct}^*}(\overline{\mathcal{P}}, \mathsf{ct})$

1 : **if** $(|\overline{\mathcal{P}}| \geq d) \vee (\mathsf{ct} = \mathsf{ct}^*)$:
2 : **return** $\perp$
3 : $[\![\mathsf{sk}]\!] \leftarrow \mathsf{KeyUpdate}([\![\mathsf{sk}]\!])$
4 : **return** $\mathsf{Dec}^{\mathfrak{L}}(\mathsf{ct}, \overline{\mathcal{P}})$

$$\mathsf{WIN}^{\mathsf{IND\text{-}CCA}}_{\mathsf{PKE}}(b) = (b = b^*) \wedge (|\overline{\mathcal{P}}_{\mathrm{KG}}| \leq d - 1)$$

Fig. 4: *(d-1)-probing* IND-CCA for PKE security game

Definition 13. *A* PKE *scheme is called (d-1)-probing* IND-CCA *secure if it holds that for all security parameters $\kappa \in \mathbb{N}$ and all* PPT *adversaries $\mathcal{A}$ in the game described in Figure 4 we have:*

$$\mathsf{Adv}^{\mathsf{dCCA}}_{\mathcal{A}}(\kappa) := \left|\Pr\left[\mathsf{WIN}^{\mathsf{IND\text{-}CCA}}(0)\right] - \Pr\left[\mathsf{WIN}^{\mathsf{IND\text{-}CCA}}(1)\right]\right| \leq \mathsf{negl}(\kappa)$$

We note that in the security game, we only require probing security for the KeyGen and the Dec procedures, leaving the Enc procedure as it is. We also note

that once again our definition coincides with the usual "no-probing" definition when $d = 1$. Note that the *(d-1)-probing* IND-CCA security notion gets stronger for larger allowed number of probes. In particular, probing IND-CCA security is at least as strong as standard IND-CCA security.

3.3 Lifting Probing IND-CPA to Probing IND-CCA

We conclude the section by proving our main theoretical result, showing how to generically upgrade *probing* IND-CPA to *probing* IND-CCA. Concretely, we use a variant of the Naor-Young transform [NY90] as presented in [Lin06] (in the random oracle model), which we refer to as the NY-transform. It uses the power of publicly verifiable NIZK's to guarantee the well-formedness of ciphertexts. As an additional contribution, we formally prove the Naor-Young transform for non-perfectly correct schemes, for which we introduce the notion of *NY-compatible* encryption schemes.

NY-Compatible PKE Let Σ be a public-key encryption scheme with message space $\mathcal{M}$ and randomness distribution $\mathsf{RandDistr}$, and let $\mathsf{RandSpace}$ be a set. Within the NY-transform, we switch from $\mathsf{RandDistr}$ to $\mathsf{RandSpace}$, as the NIZK only proves set membership, but nothing about the underlying distribution. This is particularly relevant in lattice-based instantiations, where the underlying scheme is not perfectly correct. We define the relation R_Σ to be used in the NY-transform as follows:

$$\mathsf{R}_\Sigma := \left\{ (\mathsf{x}, \mathsf{w}) \;\middle|\; \begin{array}{r} \mathsf{x} = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{pk}_1, \mathsf{pk}_2) \\ \mathsf{w} = (m, R_1, R_2) \in \mathcal{M} \times \mathsf{RandSpace}^2 \\ \mathsf{ct}_i := \mathsf{Enc}(\mathsf{pk}_i, m; R_i) \quad i \in \{1, 2\} \end{array} \right\} \tag{2}$$

We denote by $\mathsf{L}_\Sigma = \{\mathsf{x} \colon \exists \mathsf{w},\ (\mathsf{x}, \mathsf{w}) \in \mathsf{R}_\Sigma\}$ the language induced by R_Σ.

We now introduce the notion of *NY-compatible* encryption schemes. This property ensures two things. First, it bounds the probability that honestly sampled encryption randomness $R \leftarrow \mathsf{RandDistr}$ falls outside of $\mathsf{RandSpace}$. Second, it bounds the probability that there exist message-randomness tuples such that for honestly generated keys, ciphertexts produced using the message-randomness tuple lead to conflicting decryptions. This is the key to leverage the NY-transform to non-perfectly correct schemes, as a valid NIZK proof not only guarantees that the ciphertexts encrypt the same message, but also that they decrypt to the same message for honest key pairs. Without this condition, an adversary could produce two honestly formed ciphertexts that verify under the NIZK relation yet decrypt inconsistently due to decryption failures.

Definition 14. *Let $\Sigma = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a PKE scheme with message space $\mathcal{M}$, randomness distribution $\mathsf{RandDistr}$ and let $\mathsf{RandSpace}$ be a set. We say that Σ is NY-compatible for $\mathsf{RandSpace}$ with parameter $\delta_{\mathsf{NY}}(\kappa)$ if for all security parameters κ it holds $\max(P_1, P_2) \leq \delta_{\mathsf{NY}}(\kappa)$, where*

$$P_1 := \Pr_{R \leftarrow \mathsf{RandDistr}}[R \notin \mathsf{RandSpace}]$$

$$P_2 := \Pr\left[\exists (m, R_1, R_2) \in \mathcal{M} \times \mathsf{RandSpace}^2 \;\middle|\; \begin{array}{r} \mathsf{Dec}(\mathsf{sk}_1, \mathsf{Enc}(\mathsf{pk}_1, m; R_1)) \\ \neq \mathsf{Dec}(\mathsf{sk}_2, \mathsf{Enc}(\mathsf{pk}_2, m; R_2)) \end{array}\right]$$

with the latter probability taken over $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(1^\kappa)$ *for* $i \in \{1, 2\}$.

Note that in the above definition we implicitly require that one can efficiently verify if randomness R lies in RandSpace or not.

Theorem 1. *We denote by* κ *the security parameter.*

Let $\Sigma = (\mathsf{KeyGen}_{\mathsf{CPA}}, \mathsf{Enc}_{\mathsf{CPA}}, \mathsf{Dec}_{\mathsf{CPA}})$ *be a (d-1)-probing* IND-CPA *secure* PKE *scheme with message space* $\mathcal{M}$ *and randomness distribution* RandDistr *and space* RandSpace *such that* $\mathsf{Dec}_{\mathsf{CPA}}$ *is deterministic and* (d-1)-NIo*. Moreover, let* Σ *be NY-compatible for* RandSpace *with parameter* $\delta_{\mathsf{NY}}(\kappa)$.

Further, let H *be a hash function modeled as a random oracle and let* $\Pi = (\mathsf{P}^{\mathsf{H}}, \mathsf{V}^{\mathsf{H}})$ *be a* OTSS-NIZK *for the relation* R_Σ *as defined in Equation 2.*

Then, the PKE *scheme* $\Sigma' = (\mathsf{KeyGen}_{\mathsf{CCA}}, \mathsf{Enc}_{\mathsf{CCA}}, \mathsf{Dec}_{\mathsf{CCA}})$ *described in Algorithms 1-3 with the same message space* $\mathcal{M}$ *is (d-1)-probing* IND-CCA *secure.*

More concretely, assuming that Π *is* $\epsilon_{\mathsf{complete}}(\kappa)$*-complete and that* Σ *is* $\delta_{\mathsf{cor}}(\kappa)$*-correct, then the constructed* Π' *is* $\delta'_{\mathsf{cor}}(\kappa)$*-correct with* $\delta'_{\mathsf{cor}}(\kappa) \leq \delta_{\mathsf{cor}}(\kappa) + \epsilon_{\mathsf{complete}}(\kappa)$.

Moreover, assuming that Π *is* $\epsilon_{\mathsf{OTSS}}(\kappa)$*-OTSS and* $\epsilon_{\mathsf{ZK}}(\kappa)$*-ZK, every* PPT *adversary* $\mathcal{A}$ *against* Σ' *with advantage* $\mathsf{Adv}^{\mathsf{dCCA}}_{\mathcal{A}}(\kappa)$ *defines adversary* $\mathcal{B}$ *against* Σ *with advantage* $\mathsf{Adv}^{\mathsf{dCPA}}_{\mathcal{B}}(\kappa)$*, where*

$$\mathsf{Adv}^{\mathsf{dCCA}}_{\mathcal{A}}(\kappa) \leq \epsilon_{\mathsf{OTSS}}(\kappa) + \delta_{\mathsf{NY}}(\kappa) + 2(\epsilon_{\mathsf{ZK}}(\kappa) + \mathsf{Adv}^{\mathsf{dCPA}}_{\mathcal{B}}(\kappa))$$

Algorithm 1 $\mathsf{Enc}_{\mathsf{CCA}}$

Input: public key pk | message m
Output: ciphertext ct

$\mathsf{pk} = (\mathsf{pk}_1, \mathsf{pk}_2)$
$\mathsf{flag} = \mathsf{false}$
while $\mathsf{flag} = \mathsf{false}$ **do**
  $R_1 \leftarrow\!\!\$\ \mathsf{RandDistr}$
  $R_2 \leftarrow\!\!\$\ \mathsf{RandDistr}$
  if $R_1, R_2 \in \mathsf{RandSpace}$ **then**
    $\mathsf{flag} = \mathsf{true}$
  end if
end while
$\mathsf{ct}_1 \leftarrow \mathsf{Enc}_{\mathsf{CPA}}(\mathsf{pk}_1, m; R_1)$
$\mathsf{ct}_2 \leftarrow \mathsf{Enc}_{\mathsf{CPA}}(\mathsf{pk}_2, m; R_2)$
$\mathsf{x} = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1, \mathsf{ct}_2)$
$\mathsf{w} = (m, R_1, R_2)$
$\pi \leftarrow \mathsf{P}^{\mathsf{H}}(\mathsf{x}, \mathsf{w})$
$\mathsf{ct} := (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$

Algorithm 2 $\mathsf{KeyGen}_{\mathsf{CCA}}$

Input: security parameter κ
Output: public key pk | secret key sk

$(\mathsf{pk}_1, [\![\mathsf{sk}_1]\!]) \leftarrow \mathsf{KeyGen}_{\mathsf{CPA}}(1^\kappa)$
$(\mathsf{pk}_2, [\![\mathsf{sk}_2]\!]) \leftarrow \mathsf{KeyGen}_{\mathsf{CPA}}(1^\kappa)$
$\mathsf{pk} := (\mathsf{pk}_1, \mathsf{pk}_2)$
$\mathsf{sk} := [\![\mathsf{sk}_1]\!]$

Algorithm 3 $\mathsf{Dec}_{\mathsf{CCA}}$

Input: secret key sk | public key pk | ciphertext ct
Output: message m

$\mathsf{pk} = (\mathsf{pk}_1, \mathsf{pk}_2)$
$\mathsf{sk} = [\![\mathsf{sk}_1]\!]$
$\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$
$\mathsf{x} := (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1, \mathsf{ct}_2)$
if $\mathsf{V}^{\mathsf{H}}(\mathsf{x}, \pi) = 1$ **then**
  $m := \mathsf{Dec}_{\mathsf{CPA}}(\mathsf{pk}_1, [\![\mathsf{sk}_1]\!], \mathsf{ct}_1)$
else
  $m := \perp$
end if

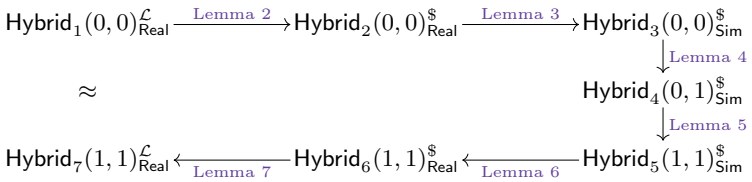

Fig. 5: Reduction Overview of NY with Probing Transformation.

The high level overview of the proof is depicted in Figure 5. The most important difference to the original NY-transform [Lin06] are the additional first and last steps (Lemma 2+7), which handle probing security related issues. Moreover, we replace the common reference string model by the random oracle model. To better follow the proof idea, we denote the hybrids by $(b_1, b_2)^{\star}_{\Delta}$, where $b_1, b_2 \in \{0,1\}$ denote which messages are encrypted in the challenge ciphertext, $\star \in \{\mathcal{L}, \$\}$ denotes whether the leakage on the decryption procedure is real or simulated and $\Delta \in \{\mathsf{Real}, \mathsf{Sim}\}$ designs whether random oracle and proof of the NIZK are real or simulated. The sub proofs which essentially follow the original NY-transform are deferred to Appendix B.1.

Proof. We start with a proof of correctness. Decryption as described in Algorithm 3 of an honestly generated ciphertext fails if either the NIZK proof is invalid or the decryption of the underlying *(d-1)-probing* $\mathsf{IND\text{-}CPA}$ secure fails (under sk_1). The first case is upper bounded by $\epsilon_{\mathsf{complete}}(\kappa)$, as the honest encryption guarantees $(\mathsf{x}, \mathsf{w}) \in \mathsf{R}_{\Sigma}$. The second case is upper bounded by $\delta_{\mathsf{cor}}(\kappa)$. A union bound gives the claimed correctness. We prove the security reduction via a sequence of hybrids.

Hybrid_1 $(0,0)^{\mathcal{L}}_{\mathsf{Real}}$: The starting hybrid corresponds to the original experiment of *(d-1)-probing* $\mathsf{IND\text{-}CCA}$ detailed in Figure 4 with bit $b = 0$.

Hybrid_2 $(0,0)^{\$}_{\mathsf{Real}}$: In the second hybrid, we change the decryption oracle, modifying how the answers $(m_i, \mathcal{L}_i)$ are produced. Everything else stays the same. Let $\mathsf{ct}_i = (\mathsf{ct}_{1i}, \mathsf{ct}_{2i}, \pi_i)$ and $\overline{\mathcal{P}}_i$ be the input to the decryption oracle. As specified in Algorithm 3, $\mathsf{Dec}_{\mathsf{CCA}}$ does essentially two things: verifying the NIZK proof π and invoking $\mathsf{Dec}_{\mathsf{CPA}}$ on input $(\mathsf{pk}_1, [\![\mathsf{sk}_1]\!], \mathsf{ct}_{1i})$. As the verification of the NIZK is a public procedure, we can assume without loss of generality that the probes $\overline{\mathcal{P}}_i$ are only on the algorithm $\mathsf{Dec}_{\mathsf{CPA}}$. Let $\mathsf{Sim}_{\mathsf{Dec}}$ be the simulator defined in Proposition 3 for the *(d-1)*-NIo program $\mathsf{Dec}_{\mathsf{CPA}}$. Instead of running the leaky decryption $(m_i, \mathcal{L}_i) \leftarrow \mathsf{Dec}^{\mathfrak{L}}_{\mathsf{CPA}}(\mathsf{pk}_1, [\![\mathsf{sk}_1]\!], \mathsf{ct}_{1i}, \overline{\mathcal{P}}_i)$, we first run the normal decryption $m_i \leftarrow \mathsf{Dec}_{\mathsf{CPA}}(\mathsf{pk}_1, [\![\mathsf{sk}_1]\!], \mathsf{ct}_{1i})$ without any probes and leakage. Then, we call $\mathsf{Sim}_{\mathsf{Dec}}$ on the input $(\mathsf{ct}_{1i}, \mathsf{pk}_1, m_i, \overline{\mathcal{P}}_i)$ to obtain the simulated output $\mathcal{L}_i$. Note that now, the leakage $\mathcal{L}_i$ does not depend on the secret key sk_1 anymore.

Hybrid_3 $(0,0)^{\$}_{\mathsf{Sim}}$: The third hybrid is the same as Hybrid_2, except how the random oracle is modeled and how the proof π during challenge ciphertext generation is constructed. Let $\mathsf{Sim}_{\mathsf{NIZK}} = (\mathsf{Sim}_{\mathsf{H}}, \mathsf{Sim}_{\pi})$ be the simulator from the $\mathsf{OTSS\text{-}NIZK}$ definition. Instead of a truly random oracle H, we let the simulator

$\mathsf{Sim}_{\mathsf{H}}$ simulate H. Moreover, we replace the real proof π^* contained in ct^* by a simulated one issued by Sim_{π}.

Hybrid_4 $(0,1)^{\$}_{\mathsf{Sim}}$: The fourth hybrid is the same as Hybrid_3, except how the challenge ciphertext is produced. Instead of encrypting twice the same message m_0 under the two different pubic keys $\mathsf{pk}_1, \mathsf{pk}_2$, we keep ct_1 as an encryption of m_0 under pk_1. However, we now produce ct_2 as the encryption of m_1 under pk_2.

Hybrid_5 $(1,1)^{\$}_{\mathsf{Sim}}$: The fifth hybrid is the same as Hybrid_4, except we modify again how the challenge ciphertext is produced. Instead of setting ct_1 as the encryption of m_0 under pk_1, we let it be the encryption of m_1, still under the public key pk_1. We thus have ct_1 and ct_2 be encryptions of the same message m_1.

Hybrid_6 $(1,1)^{\$}_{\mathsf{Real}}$: The sixth hybrid is the same as Hybrid_5, except we modify how the random oracle H is modeled and how the proof π during challenge ciphertext generation is constructed — the change from Hybrid_2 to Hybrid_3 in reverse order. Instead of letting the simulator $\mathsf{Sim}_{\mathsf{NIZK}}$ simulate H, we model it as a truly random oracle. Moreover, we replace the simulated proof π^* contained in ct^* by a real one issued by P.

Hybrid_7 $(1,1)^{\mathcal{L}}_{\mathsf{Real}}$: The seventh hybrid is the same as Hybrid_6, except we modify how the leakage for the decryption oracle is constructed — the change from Hybrid_1 to Hybrid_2 in the reverse order. Instead of using $\mathsf{Sim}_{\mathsf{Dec}}$ on the input $(\mathsf{ct}_i, \mathsf{pk}, \overline{\mathcal{P}}_i)$ to get the leakage $\mathcal{L}_i$ and of running the non-leaky usual decryption $\mathsf{Dec}(\mathsf{pk}, [\![\mathsf{sk}]\!], \mathsf{ct}_i)$ to get the message m_i, we run the leaky decryption $\mathsf{Dec}^{\mathfrak{L}}(\mathsf{pk}, [\![\mathsf{sk}]\!], \mathsf{ct}_i, \overline{\mathcal{P}}_i)$ to get both the message m_i and the leakage $\mathcal{L}_i$. Note that this corresponds to security game of *(d-1)-probing* IND-CCA with $b = 1$

We now prove the advantages for distinguishing between consecutive hybrids.

Lemma 2 (Hybrid_1 to Hybrid_2). *Assume the decryption algorithm* $\mathsf{Dec}_{\mathsf{CPA}}$ *of* Σ *is deterministic and* (d-1)-NIo*. Then, the advantage in distinguishing* Hybrid_1 *from* Hybrid_2 *is zero.*

Proof. Recall that the security notion of *(d-1)-probing* IND-CCA invokes before answering to a decryption query the KeyUpdate algorithm which refreshes the sharing of the secret key $\mathsf{sk} = \mathsf{sk}_1$. By the properties of KeyUpdate (and the fact that the model does not allow the adversary to probe this algorithm), we know that the (at most $d-1$) probes on the sharing $[\![\mathsf{sk}]\!]$ during a new query to the decryption oracle are independent to any information the adversary might have obtained from previous probes on $[\![\mathsf{sk}]\!]$. Hence, it is enough to argue for a single decryption query.

We now argue that $\mathsf{Dec}_{\mathsf{CPA}}$ fulfills the conditions of Proposition 3. It has no unmasked secret input, only one d-masked secret input $[\![\mathsf{sk}_1]\!]$ and its public input is given by $(\mathsf{pk}_1, \mathsf{ct}_{1i})$. Moreover, its public output m_i is uniquely defined by $(\mathsf{pk}_1, \mathsf{ct}_{1i}, \mathsf{sk}_1)$ (not depending on the exact sharing of sk_1, and no secret output). Thus, all conditions of Proposition 3 are met, guaranteeing the existence of a simulator $\mathsf{Sim}_{\mathsf{Dec}}$ which perfectly simulates leakage $\mathcal{L}_i$ without needing the masked input $[\![\mathsf{sk}_1]\!]$. ■

Proofs for Lemma 3+4 follow the usual Naor-Young reduction, they can be found in Appendix B.1 for completeness.

Lemma 3 (Hybrid_2 **to** Hybrid_3)**.** *Let Π be $\epsilon_{\mathsf{ZK}}(\kappa)$-zero-knowledge. Then, for any any* PPT *adversary $\mathcal{A}$, the advantage in distinguishing* Hybrid_2 *from* Hybrid_3 *is upper bounded by $\epsilon_{\mathsf{ZK}}(\kappa)$.*

Lemma 4 (Hybrid_3 **to** Hybrid_4)**.** *Let Σ be (d-1)-probing* IND-CPA *secure. That is, any* PPT *adversary $\mathcal{B}$ against (d-1)-probing* IND-CPA *of Σ has advantage at most $\mathsf{Adv}^{\mathsf{dCPA}}_{\mathcal{B}}(\kappa)$. Then, for any* PPT *adversary $\mathcal{A}$, the advantage in distinguishing* Hybrid_3 *from* Hybrid_4 *is upper bounded by $\mathsf{Adv}^{\mathsf{dCPA}}_{\mathcal{B}}(\kappa)$.*

Lemma 5 (Hybrid_4 **to** Hybrid_5)**.** *Let (Σ, Π) be $\delta_{\mathsf{NY}}(\kappa)$-NY-compatible and let Σ be (d-1)-probing* IND-CPA *secure. The latter is, any* PPT *adversary $\mathcal{B}$ against (d-1)-probing* IND-CPA *of Σ has advantage at most $\mathsf{Adv}^{\mathsf{dCPA}}_{\mathcal{B}}(\kappa)$. Moreover, let Π be $\epsilon_{\mathsf{OTSS}}(\kappa)$-one-time-simulation-sound. For any* PPT *adversary $\mathcal{A}$, the advantage in distinguishing* Hybrid_4 *from* Hybrid_5 *is upper bounded $\mathsf{Adv}^{\mathsf{dCPA}}_{\mathcal{B}}(\kappa) + \delta_{\mathsf{NY}}(\kappa) + \epsilon_{\mathsf{OTSS}}(\kappa)$.*

Proof. For simplicity, we omit the dependency on the security parameter κ. Let (Σ, Π) be δ_{NY}-NY-compatible and Π be ϵ_{OTSS}-OTSS. Further, let $\mathcal{A}$ be a PPT adversary distinguishing between Hybrid_4 and Hybrid_5 with advantage at least Adv, then we build a PPT adversary $\mathcal{B}$ that breaks the *(d-1)-probing* IND-CPA security of Σ with advantage $\mathsf{Adv}^{\mathsf{dCPA}}_{\mathcal{B}}\mathcal{A}$ at least $\mathsf{Adv} - \delta_{\mathsf{NY}} - \epsilon_{\mathsf{OTSS}}$.

We begin by describing the reduction. We proceed similarly to the proof of Lemma 4, except we now embed the *(d-1)-probing* IND-CPA challenge in pk_1 and ct_1 instead of pk_2 and ct_2. First $\mathcal{B}$ receives from $\mathcal{A}$ as input a set of probes $\overline{\mathcal{P}}_{KG} = (\overline{\mathcal{P}}_{KG1}, \overline{\mathcal{P}}_{KG2})$ on the key generation procedure. The key pair $(\mathsf{pk}_2, \mathsf{sk}_2)$ and leakage $\mathcal{L}_{KG2}$ are computed by $\mathcal{B}$ who runs $\mathsf{KeyGen}^{\mathfrak{L}}_{\mathsf{CPA}}$, while pk_1 and $\mathcal{L}_{KG1}$ are input to them after they output $\overline{\mathcal{P}}_{KG1}$ as their query. Then $\mathcal{B}$ sends the public key $\mathsf{pk} = (\mathsf{pk}_1, \mathsf{pk}_2)$, and the leakage $\mathcal{L}_{KG} = (\mathcal{L}_{KG1}, \mathcal{L}_{KG2})$ to $\mathcal{A}$. When $\mathcal{B}$ receives m_0 and m_1 from $\mathcal{A}$, they have the ciphertext ct_1 be input to them after forwarding m_0 and m_1 as output. Then $\mathcal{B}$ computes ct_2 by calling $\mathsf{Enc}_{\mathsf{CPA}}$ on m_1 and pk_2 and they compute π by calling the simulator Sim_π on $\mathsf{x} = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1, \mathsf{ct}_2)$. They then send the challenge ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$ to $\mathcal{A}$. When $\mathcal{B}$ is input with a decryption query, they perform decryption by first verifying the proof π, then running $\mathsf{Dec}_{\mathsf{CPA}}$ on sk_2 — instead of sk_1 — and the ciphertext queried, as they do not have access to sk_1. The leakage simulator continues to run on the same input as before. The reduction $\mathcal{B}$ outputs as their guess $b^* = 0$ if $\mathcal{A}$ guesses Hybrid_3, and $b^* = 1$ if $\mathcal{A}$ guesses Hybrid_4.

The proof of correct simulation of the views of $\mathcal{A}$ is exactly the same than in Lemma 4's proof, except for decryption queries. Let $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$ be any ciphertext queried to $\mathcal{B}$. As the NIZK proof system Π is ϵ_{OTSS}-OTSS and Σ is δ_{NY}-NY-compatible, $\mathcal{A}$ won't be able to do a decryption query which passes the NIZK verification, but where ct_1 and ct_2 are not encryptions of the same message or such that $\mathsf{Dec}_{\mathsf{CPA}}(\mathsf{ct}_1, \mathsf{sk}_1) \neq \mathsf{Dec}_{\mathsf{CPA}}(\mathsf{ct}_2, \mathsf{sk}_2)$, except with probability $\epsilon_{\mathsf{OTSS}} + \delta_{\mathsf{NY}}$. We now assume that for all queries both ciphertexts are encryptions of the same message and that both decrypt to the same element. Then the distribution of the answer to decryption queries, containing both message and leakage, is perfectly simulated. Hence $\mathcal{B}$ perfectly simulates the views of $\mathcal{A}$.

As in Lemma 4's proof, we have that whenever $\mathcal{A}$ wins, if $\mathcal{B}$ perfectly simulated the views of $\mathcal{A}$ then they also win. Therefore, as $\mathcal{A}$ has advantage Adv, $\mathcal{B}$ has advantage at least $\mathsf{Adv} - \epsilon_{\mathsf{OTSS}} - \delta_{\mathsf{NY}}$. ■

Lemma 6 (Hybrid_5 **to** Hybrid_6)**.** *Let Π be $\epsilon_{\mathsf{ZK}}(\kappa)$-zero-knowledge. Then, for any any* PPT *adversary $\mathcal{A}$, the advantage in distinguishing Hybrid_5 from Hybrid_6 is upper bounded by $\epsilon_{\mathsf{ZK}}(\kappa)$.*

Proof. The proof of Lemma 6 is essentially identical to that of Lemma 3. It suffices to replace all encryptions of m_0 by encryptions of m_1. ■

Lemma 7 (Hybrid_6 **to** Hybrid_7)**.** *Assume the decryption algorithm $\mathsf{Dec}_{\mathsf{CPA}}$ of Σ is* (d-1)-NIo*. Then, the advantage in distinguishing Hybrid_6 from Hybrid_7 is zero.*

Proof. The proof of Lemma 7 is essentially the same as for Lemma 2. It suffices to replace all encryptions of m_0 by encryptions of m_1. ■

By applying all consecutive lemmas from Lemma 2 to Lemma 7, we thus have the following bound on the advantage Adv of any adversary $\mathcal{A}$ that distinguishes between Hybrid_1 and Hybrid_7 – thus concluding the proof:

$$\mathsf{Adv} \leq \epsilon_{\mathsf{OTSS}}(\kappa) + 2(\epsilon_{\mathsf{ZK}}(\kappa) + \mathsf{Adv}^{\mathsf{dCPA}}(\kappa)) + \delta_{\mathsf{NY}}$$

□

4 Description of Rutile

In this section, we provide the description of Rutile, a lattice-based public-key encryption scheme, designed with masking-friendliness in mind. We further prove its correctness and its NY-compatibility. The probing IND-CPA security proof of Rutile is deferred to Section 6.1.

It is inspired both by the IND-CPA version of Kyber [BDK+18] and the masking-friendly signature Raccoon [dEK+23]. Without loss of generality, we assume that the ring degree n is lower bounded by the security parameter κ. Hence, proving that something is negligible in n implies that it is negligible in κ.

The main parameters of Rutile are the following: the power-of-two cyclotomic ring $\mathcal{R}$ of degree n modulo q, the dimension k of the public key matrix $\mathbf{A}$,

the masking level d and the parameters (u, rep) for uniformly sampling short elements over $\mathcal{R}$.

Algorithm 4 $\mathsf{KeyGen}_{\mathsf{CPA}}$

Input: level of security κ
Output: public key $\mathsf{pk} = (\mathbf{A} \in \mathcal{R}_q^{k\times k}, \mathbf{t} \in \mathcal{R}_q^k)$ | private key $\mathsf{sk} = [\![\mathbf{s}]\!] \in (\mathcal{R}_q^k)^d$

1: $\mathbf{A} \leftarrow\!\!\$\ \mathcal{R}_q^{k\times k}$
2: $[\![\mathbf{s}]\!] \leftarrow \mathsf{AddRepNoise}(\mathbf{0}_{d,1}, u, \mathrm{rep})$ ▷ NIuo gadget for $\mathbf{s} \leftarrow \mathsf{SUnif}(\mathcal{R}, k, u, d \cdot \mathrm{rep})$
3: $[\![\mathbf{e}]\!] \leftarrow \mathsf{AddRepNoise}(\mathbf{0}_{d,1}, u, \mathrm{rep})$ ▷ NIuo gadget for $\mathbf{e} \leftarrow \mathsf{SUnif}(\mathcal{R}, k, u, d \cdot \mathrm{rep})$
4: $[\![\mathbf{t}]\!] \leftarrow \mathbf{A} \cdot [\![\mathbf{s}]\!] + [\![\mathbf{e}]\!]$
5: $\mathbf{t} := \mathsf{Decode}([\![\mathbf{t}]\!])$ ▷ NIo gadget for $\mathbf{t} := \sum_i [\![\mathbf{t}]\!]_i$
6: $\mathsf{pk} := (\mathbf{A}, \mathbf{t})$
7: $\mathsf{sk} := [\![\mathbf{s}]\!]$

Algorithm 4 describes the key generation. It is essentially the same as the key generation in Raccoon. It generates an M-LWE instance with masking-friendly sums of uniform distributions for secret and error. Note that Kyber's key generation uses M-LWE with centered binomials instead. The key difference is that the masking level d already appears in the definition of the secret/error distribution, as it impacts the number of summands. Concretely, we use the $\mathsf{AddRepNoise}$ gadget from [dPKPR24, Alg. 5] to sample sums of uniforms in a masked way. To be precise, the gadget from [dPKPR24] is for non-zero centered sums of uniforms (which we called NSUnif in the preliminaries). It easily extends to a gadget for zero-centered sums of uniforms, simply by adding a constant off-set. The Decode gadget can be obtained by applying [CGMZ23, Cor. 1] to [BCRT23, Cor. 5]. More details on this can be found in Section 5.

The encryption procedure, detailed in Algorithm 5, is essentially the same as the one of the $\mathsf{IND\text{-}CPA}$ version of Kyber, where for simplicity we left out the ciphertext compression at the end.[2] Encryption randomness is again sampled from sums of uniform distributions, even though we do not require masking techniques for the encryption. Note that the message space is given by $\mathcal{M} = \{0,1\}^n$ and the encryption randomness is coming from the distribution $\mathsf{RandDistr} = \mathsf{SUnif}(\mathcal{R}, 2k+1, u, d \cdot \mathrm{rep})$.

Algorithm 5 $\mathsf{Enc}_{\mathsf{CPA}}$

Input: message μ | the public key $\mathsf{pk} = (\mathbf{A}, \mathbf{t})$
Output: ciphertext $\mathsf{ct} = (v, \mathbf{u}) \in \mathcal{R}_q \times \mathcal{R}_q^k$

1: $\mathbf{u} \leftarrow \mathsf{SUnif}(\mathcal{R}, k, u, d \cdot \mathrm{rep})$
2: $\mathbf{e}_1 \leftarrow \mathsf{SUnif}(\mathcal{R}, k, u, d \cdot \mathrm{rep})$
3: $e_2 \leftarrow \mathsf{SUnif}(\mathcal{R}, 1, u, d \cdot \mathrm{rep})$
4: $\mathbf{v}^t := \mathbf{u}^t \cdot \mathbf{A} + \mathbf{e}_1^t$
5: $v := \mathbf{u}^t \cdot \mathbf{t} + e_2 + \lfloor q/2 \rceil \mu$
6: $\mathsf{ct} := (v, \mathbf{v})$

[2] Compressing ciphertexts would require a more sophisticated setup of the NIZK which we leave for future works.

Algorithm 6 $\mathsf{Dec}_{\mathsf{CPA}}$

Input: ciphertext $\mathsf{ct} = (v, \mathbf{v}) \in \mathcal{R}_q \times \mathcal{R}_q^k$ | private key $\mathsf{sk} = [\![\mathbf{s}]\!] \in (\mathcal{R}_q^k)^d$
Output: message μ

1: $[\![\mu']\!] \leftarrow v - \mathbf{v}^t \cdot [\![\mathbf{s}]\!]$
2: $[\![\mu]\!] \leftarrow \mathsf{1bit\text{-}Compress}([\![\mu']\!])$ ▷ NI gadget for a d-sharing of $\left\lfloor \|\mu'\|_\infty / \lfloor q/2 \rceil \right\rceil$
3: $\mu := \mathsf{Decode}([\![\mu]\!])$ ▷ NIo gadget for $\mu := \sum_i [\![\mu]\!]_i$

Algorithm 6 states the decryption procedure. It can be seen as a masked version of the decryption algorithm from the IND-CPA version of Kyber, just without the decompress coming from the compressed ciphertexts.

To simplify notations we omit the NTT representations here, but when concretely instantiating our scheme one can add the NTT without any problems. Moreover, in practice, one usually generates the matrix $\mathbf{A}$ from a short seed $\mathsf{seed} \leftarrow \{0,1\}^\kappa$. A description of the 1bit-Compress gadget can be found in [BC22, Alg. 15]. A proof of correctness can be found in App. C.

We now argue the NY-compatibility of Σ, as described in Alg. 4-6, with message space $\mathcal{M} = \{0,1\}^n$ and encryption randomness distribution $\mathsf{RandDistr} = \mathsf{SUnif}(\mathcal{R}, 2k+1, u, d \cdot \mathrm{rep})$. Set $B^{(e)} = 2^u \sqrt{3n\,(2k+1)\,d\,\mathrm{rep}}$ and define $\mathsf{RandSpace} = \left\{\mathbf{r} \in \mathcal{R}^{2k+1} \colon \|\mathbf{r}\|_2 \le B^{(e)}\right\}$.

We can reformulate the encryption algorithm as follows. Set $k' = 2k+1$ and let $\mathbf{r} = (\mathbf{u}, \mathbf{e}_1, e_2)$ and $\mathbf{B} = \begin{pmatrix} \mathbf{A}^t & \mathbf{I}_k & \mathbf{0}_{k,1} \\ \mathbf{t}^t & \mathbf{0}_{1,k} & 1 \end{pmatrix}$, then

$$\mathbf{c} = \mathbf{B}\mathbf{r} + \lfloor q/2 \rceil \cdot (\mathbf{0}_{1,k}, \mu)^t \in \mathcal{R}_q^{k+1} \tag{3}$$

With this notation, the PKE public key space is characterized by

$$\mathcal{K} = \left\{ \mathbf{B} = \begin{pmatrix} \mathbf{A}^t & \mathbf{I}_k & \mathbf{0}_{k,1} \\ \mathbf{t}^t & \mathbf{0}_{1,k} & 1 \end{pmatrix} \in \mathcal{R}_q^{k+1 \times k'} \;\middle|\; \mathbf{A} \in \mathcal{R}_q^{k \times k}, \mathbf{t} \in \mathcal{R}_q^k \right\}$$

Let R_Σ be the relation defined in Section 3.3. With our concrete instantiation of Σ at hand, we reformulate it to

$$\mathsf{R}_\Sigma := \left\{ (\mathsf{x}, \mathsf{w}) \;\middle|\; \begin{array}{r} \mathsf{x} = (\mathbf{c}_1, \mathbf{c}_2, \mathbf{B}_1, \mathbf{B}_2) \in \left(\mathcal{R}_q^{k+1}\right)^2 \times \mathcal{K}^2 \\ \mathsf{w} = (\mu, \mathbf{r}_1, \mathbf{r}_2) \in \mathcal{M} \times \mathsf{RandSpace}^2 \\ \mathbf{c}_i = \mathbf{B}_i \mathbf{r}_i + \lceil q/2 \rceil \cdot (0_{1,k}, \mu)^t \quad i \in \{1,2\} \end{array} \right\}$$

We now show NY-compatibility of this scheme.

Lemma 8. *Let Σ be the encryption scheme above with message space $\mathcal{M}$. For $B^{(e)}$ define the randomness space* $\mathsf{RandSpace} = \left\{\mathbf{r} \in \mathcal{R}^{2k+1} \colon \|\mathbf{r}\|_2 \le B^{(e)}\right\}$. *If $\left(B^{(e)}\right)^2 \le \lfloor q/4 \rfloor$, then Σ is NY-compatible for* $\mathsf{RandSpace}$ *with parameter $\delta_{\mathsf{NY}} = 2\exp(-2n(2k+1))$.*

Proof. We set $\delta = \exp(-2n\,(2k{+}1))$ and let P_1 and P_2 as defined in Definition 14. First, we show that $\mathsf{RandSpace}$ is chosen in a way that honestly generated encryption randomness lies in $\mathsf{RandSpace}$ with overwhelming probability. By Eq. 1 applied to $\mathsf{RandDistr} = \mathsf{SUnif}(\mathcal{R}, 2k+1, u, d \cdot \mathrm{rep})$, we obtain

$$P_1 = \Pr_{R \leftarrow \mathsf{RandDistr}}[R \notin \mathsf{RandSpace}] \leq \delta$$

Second, we show that

$$\Pr_{(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KeyGen}_{\mathsf{CPA}}}[\exists (\mu, \mathbf{r}) \in \mathcal{M} \times \mathsf{RandSpace} \,|\mathsf{Dec}_{\mathsf{CPA}}(\mathsf{sk}, \mathsf{Enc}_{\mathsf{CPA}}(\mathsf{pk}, \mu; \mathbf{r})) \neq \mu] \leq \delta$$

which implies that $P_2 \leq 2\delta$. From the proof of Lemma 14, we know that decryption succeeds if $\|\mathbf{r}^t \cdot \mathbf{z}\|_\infty \leq \lfloor q/4 \rfloor$, where $\mathbf{z} = (\mathbf{e}, -\mathbf{s}, 1)$ comes from the key generation. In contrast to the correctness proof, we do not know whether $\mathbf{r}$ has been sampled correctly from $\mathsf{RandDistr}$, but we know that $\mathbf{r} \in \mathsf{RandSpace}$, hence $\|\mathbf{r}\|_2 \leq B^{(e)}$. As the key generation is assumed to be honest, we can further bound $\|(\mathbf{e}, -\mathbf{s}, 1)\|_2$ by $B^{(e)}$, except with probability $\delta = \exp(-2n(2k+1))$. As $\delta_{\mathsf{NY}} = \max(P_1, P_2) = 2\delta$, this concludes the proof. □

Note that in our scheme the correctness parameter δ_{cor} coincides with the NY-compatibility parameter δ_{NY}. This is not the case in general.

5 Non-Interference Properties of Rutile

In this section we prove non-interference properties of Rutile defined in Section 4. The non-interference of the key generation is needed to prove probing IND-CPA security in Section 6.1. The non-interference of the decryption is needed as an ingredient of the probing variant of the NY-transform from Section 3.3.

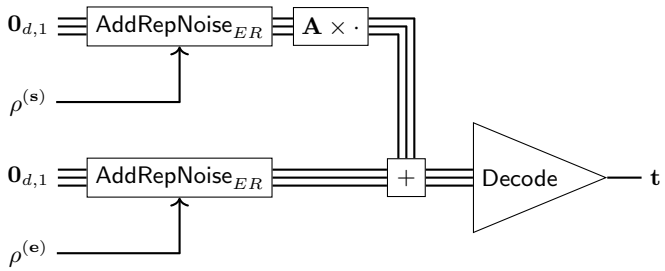

Fig. 6: DAG representing the structure of $\mathsf{KeyGen}_{\mathsf{ER}}$

Key Generation. In Section 4 we described a version of $\mathsf{KeyGen}_{\mathsf{CPA}}$ in which we call the procedure $\mathsf{AddRepNoise}$. Essentially, this procedure samples a number of small uniforms – as with Unif– adds them to a d-shared input, adds a constant off-set and outputs a d-shared variable. In this section we use a slight variation of this procedure which we call $\mathsf{AddRepNoise}_{\mathsf{ER}}$ for which small uniforms to be summed are directly given as input and not sampled during the execution of the procedure. Here, ER stands for Explicit Randomness. We thus call $\mathsf{KeyGen}_{\mathsf{ER}}$ the procedure relying on $\mathsf{AddRepNoise}_{\mathsf{ER}}$ instead of $\mathsf{AddRepNoise}$ and which takes as additional secret unmasked input the small uniforms to be input to the two $\mathsf{AddRepNoise}_{\mathsf{ER}}$. For clarity, we note $\boldsymbol{\rho}^{(s)}$ [resp. $\boldsymbol{\rho}^{(e)}$] the part of these small uniforms used to construct $\mathbf{s}$ [resp. $\mathbf{e}$]. It is straightforward that first sampling $\boldsymbol{\rho}^{(s)}$ and $\boldsymbol{\rho}^{(e)}$ and then inputting them $\mathsf{KeyGen}_{\mathsf{ER}}$ is indistinguishable from using $\mathsf{KeyGen}_{\mathsf{CPA}}$ directly. The DAG representing $\mathsf{KeyGen}_{\mathsf{ER}}$ can be found in Figure 6. A formal description of $\mathsf{AddRepNoise}_{\mathsf{ER}}$ and $\mathsf{KeyGen}_{\mathsf{ER}}$ can be found in [dPKPR24, Sec. 6], from which most of our inspiration for this section comes from. Also, $\mathsf{AddRepNoise}_{\mathsf{ER}}$ is required to be sNIu — a stronger version of NIu — by [dPKPR24, Lemma 6.2] which we invoke when proving Lemma 9. We provide a definition of sNIu for completeness in Appendix A.

Decryption. The DAG representing $\mathsf{Dec}_{\mathsf{CPA}}$ can be found in Figure 7.

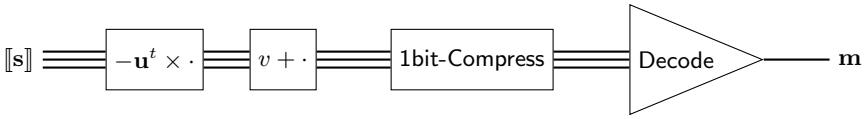

Fig. 7: DAG representing the structure of $\mathsf{Dec}_{\mathsf{CPA}}$

Lemma 9. *Assuming that* $\mathsf{AddRepNoise}_{\mathsf{ER}}$ *is* sNIu, $\mathsf{1bit\text{-}Compress}$ *is* NI *and that* Decode *is* NIo, *then we have that* $\mathsf{KeyGen}_{\mathsf{ER}}$ *is* NIuo *and that* $\mathsf{Dec}_{\mathsf{CPA}}$ *is* NIo.

Proof. Note that any (bi)linear operation is NI when applied share-by-share.

We denote by $\mathsf{KeyGen}^*_{\mathsf{ER}}$ the circuit in Fig. 6 without the Decode procedure (i.e. the circuit that outputs $[\![\mathbf{t}]\!]$). Assumptions and design of $\mathsf{KeyGen}^*_{\mathsf{ER}}$ coincides with that of $\mathsf{KeyGen}_{\mathsf{ER}}$ in [dPKPR24, Sec. 6.3], thus, we forward readers to their proof that $\mathsf{KeyGen}^*_{\mathsf{ER}}$ is NIu. By assumption Decode is NIo, thus, applying Proposition 2 to composition of $\mathsf{KeyGen}^*_{\mathsf{ER}}$ and Decode suffices.

The composition of two NI gadgets which only take as input one masked variable and only output one masked variable is NI as well [BBD+16, Prop. 4]. All base procedures of $\mathsf{Dec}_{\mathsf{CPA}}$ are NI except Decode (by assumption), they only take as input one masked variable, only output one masked variable and no public output, therefore, their chain composition (i.e $[-\mathbf{u}^t \times \cdot] \to [v + \cdot] \to [\mathsf{1bit\text{-}Compress}]$) is NI as well. As this chain composition is NI it is NIu as well, and as Decode is NIo, their composition – i.e. $\mathsf{Dec}_{\mathsf{CPA}}$ – is NIuo by Proposition 2. As $\mathsf{Dec}_{\mathsf{CPA}}$ is NIuo and has no unshared input, it is NIo. □

Remark 1. You can found a sNIu implementation of $\mathsf{AddRepNoise}_{\mathsf{ER}}$ in [dP-KPR24, Alg. 9], and a NI implementation of $\mathsf{1bit\text{-}Compress}$ in [BC22, Alg. 15]. Also, using [CGMZ23, Cor. 1] on [BCRT23, Cor. 5] yields a NIo implementation of Decode.

We implicitly rely on the reduction from probing security to noisy leakage resilience [DDF19]. Hence, we obtain the standard trade-off between (at most) quadratic increase in cost of masking and exponential increase for the attacker.

6 Probing IND-CCA Security of Topaz

In this last section, we prove the probing $\mathsf{IND\text{-}CPA}$ security of Rutile. With this at hand, we have now all the ingredients ready to instantiate the NY-transform in the probing setting from Section 3.3, upgrading Rutile to Topaz. We conclude with non-optimized concrete parameters for both encryption schemes.

6.1 Probing IND-CPA Security of Rutile

To prove that the PKE scheme described in Section 4 is probing $\mathsf{IND\text{-}CPA}$ secure, we proceed through a reduction very similar to the $\mathsf{IND\text{-}CPA}$ reduction of Kyber [BDK+18]. We need two additional hybrids to replace true leakage of the KeyGen by simulated leakage. The public key together with the simulated leakage can then be captured by the probing M-LWE hardness assumption.

Theorem 2. *Let* $\Sigma = (\mathsf{KeyGen}_{\mathsf{CPA}}, \mathsf{Enc}_{\mathsf{CPA}}, \mathsf{Dec}_{\mathsf{CPA}})$ *be the scheme described in Section 4 with masking parameter* d*. We have that* Σ *is (d-1)-probing* $\mathsf{IND\text{-}CPA}$ *and for any* PPT *adversary* $\mathcal{A}$ *against* Σ *with advantage* $\mathsf{Adv}^{\mathsf{dCPA}}_{\mathcal{A}}$ *we have:*

$$\mathsf{Adv}^{\mathsf{dCPA}}_{\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{M\text{-}LWE}}_{\mathcal{R}_q,k,d\cdot(rep-1)+1,u} + \mathsf{Adv}^{\mathsf{M\text{-}LWE}}_{\mathcal{R}_q,k+1,d\cdot rep,u}$$

where $\mathsf{Adv}^{\mathsf{M\text{-}LWE}}$ *denotes the advantage against the* M-LWE *problem.*

Proof. The reduction does a sequence of hybrids.

Hybrid_1: The starting hybrid corresponds to the *(d-1)-probing* $\mathsf{IND\text{-}CPA}$ original experiment detailed in Figure 3 with bit $b \in \{0,1\}$.

Hybrid_2: In the second hybrid, we replace the KeyGen function by the following: first we sample $2\,d\,\mathrm{rep}$ small uniforms from $\mathsf{Unif}(\mathcal{R}, k, u)$, and then feed it as input to $\mathsf{KeyGen}_{\mathsf{ER}}$, where the latter is as described in Section 5. Without loss of generality we only consider probes on $\mathsf{KeyGen}_{\mathsf{ER}}$. Indeed, all probable wires of Unif are part of the input wires of $\mathsf{KeyGen}_{\mathsf{ER}}$, thus, considering probes on $\mathsf{KeyGen}_{\mathsf{ER}}$ does not prevent from leaking values from Unif.

Hybrid_3: Lemma 9 proves that $\mathsf{KeyGen}_{\mathsf{ER}}$ is NIuo. In the third hybrid we use this property to replace the real leakage output by $\mathsf{KeyGen}^{\mathfrak{L}}_{\mathsf{ER}}$ with simulated leakage output by $\mathsf{KeyGen}_{\mathsf{ER}}$'s NIuo simulator. Let $\mathsf{SimProb}, \mathsf{SimLeak}$ be the PPT NIuo simulators of $\mathsf{KeyGen}_{\mathsf{ER}}$ (see. Section 5) and $\overline{\mathcal{P}}_{KG}$ any set of probes forwarded by the adversary. We denote by $(\boldsymbol{\rho}_i)_i$ the small uniforms carried by the wires specified by the output of $\mathsf{SimProb}(\overline{\mathcal{P}}_{KG})$ and denote by pk the public key

output by $\mathsf{KeyGen}_{\mathsf{ER}}$. We now replace real leakage $\mathcal{L}_{KG}$ output by $\mathsf{KeyGen}^{\mathfrak{L}}_{\mathsf{ER}}$ by simulated leakage $\mathcal{L}'_{KG}$ output by $\mathsf{SimLeak}((\boldsymbol{\rho}_i), \mathsf{pk}, \overline{\mathcal{P}})$.

Hybrid_4: In the fourth hybrid, we replace the pk output by $\mathsf{KeyGen}_{\mathsf{ER}}$ by a pair $(\mathbf{A}, \mathbf{u}) \leftarrow\!\!\$\ \mathcal{R}_q^{k\times k} \times \mathcal{R}_q^k$ uniformly sampled.

Hybrid_5: In the fifth hybrid, we replace the ciphertext ct^* by a vector $\mathbf{c} \leftarrow\!\!\$\ \mathcal{R}_q^{k+1}$ uniformly sampled.

We now prove the advantages for distinguishing between consecutive hybrids.

Lemma 10 (Hybrid_1 to Hybrid_2). *The advantage in distinguishing* Hybrid_1 *and* Hybrid_2 *is zero.*

Proof. The only difference between the two hybrids is that small uniforms are sampled earlier in Hybrid_2, this does not induce any change in either pk, $\mathcal{L}_{KG}$, or ct^*. Hence, the adversary's advantage is the same. ■

Lemma 11 (Hybrid_2 to Hybrid_3). *The advantage in distinguishing* Hybrid_2 *and* Hybrid_3 *is zero.*

Proof Sketch. The overall idea is to use the NIuo property of $\mathsf{KeyGen}_{\mathsf{ER}}$ to prove that the joint distribution of the public key and leakage is the same in both hybrids. If so, the views of $\mathcal{A}$ are identical in both hybrids, thus their distinguishing advantage is zero. The NIuo property implies identical distribution of the leakages under the assumption that public output — i.e the public key — and all input — i.e small uniforms — are fixed. But in our hybrids those values come from certain distributions. We account for this by using Bayes formula (to condition on fixed values for the public key) and by using the law of total probability (to condition on possible small uniform values). Now, we can invoke the NIuo property and the proof ensues. We note it is important that in both hybrids this set of values for small uniforms is the same, as the sampling of small uniforms and public key computation is done the same way. A full proof can be found in App. D.

Lemma 12 (Hybrid_3 to Hybrid_4). *For any* PPT *adversary* $\mathcal{A}$*, the advantage in distinguishing* Hybrid_3 *and* Hybrid_4 *is upper bounded by* $\mathsf{Adv}^{\mathsf{M\text{-}LWE}}_{\mathcal{R}_q,k,d\cdot(rep-1)+1,u}$.

Proof. Let $\mathcal{A}$ be a PPT adversary distinguishing between Hybrid_3 and Hybrid_4 with advantage at least Adv, then we build a PPT adversary $\mathcal{B}$ that solves the *(d-1)-probing* $\text{M-LWE}_{\mathcal{R}_q,d,k,d\cdot\mathrm{rep},u}$ problem with advantage at least Adv.

We begin by describing the reduction. First $\mathcal{B}$ receives as input a matrix $\mathbf{A}$. Upon receiving $\overline{\mathcal{P}}_{KG}$ from $\mathcal{A}$, the reduction $\mathcal{B}$ sends $\overline{\mathcal{P}} := \mathsf{SimProb}(\overline{\mathcal{P}}_{KG})$ as the probes on small uniforms for the *(d-1)-probing* M-LWE instance. The reduction $\mathcal{B}$ then receives the probing M-LWE instance $(\mathbf{b}, \mathcal{L})$. They compute the leakage $\mathcal{L}_{KG} = \mathsf{SimLeak}(\mathcal{L}, \mathbf{b}, \overline{\mathcal{P}}_{KG})$ and send it to $\mathcal{A}$ along with $\mathsf{pk} = (\mathbf{A}, \mathbf{b})$. The reduction $\mathcal{B}$ outputs as their guess $b' = 0$ if $\mathcal{A}$ guesses Hybrid_3, and $b' = 1$ if $\mathcal{A}$ guesses Hybrid_4.

When $\mathcal{B}$ is input with an M-LWE instance and real leakage, the public key pk and leakage $\mathcal{L}_{KG}$ are computed exactly as in Hybrid_3. When they are input

with a uniform vector and fake leakage, the public key pk and leakage $\mathcal{L}_{KG}$ are computed exactly as in Hybrid_4.

As in both situations $\mathcal{B}$ computes either hybrid exactly as they are described, it follows that whenever $\mathcal{A}$ wins, the adversary $\mathcal{B}$ also wins. This concludes the reduction. By Proposition 1, we have that *(d-1)-probing* $\text{M-LWE}_{\mathcal{R}_q,d,k,d\cdot\text{rep},u} \Leftrightarrow \text{M-LWE}_{\mathcal{R}_q,k,d\cdot(\text{rep}-1)+1,u}$. This conclude the proof. ■

Lemma 13 (Hybrid_4 **to** Hybrid_5)**.** *The advantage in distinguishing* Hybrid_4 *and* Hybrid_5 *is upper bounded by* $\mathsf{Adv}^{\text{M-LWE}}_{\mathcal{R}_q,k+1,d\cdot rep,u}$.

Proof. As in both hybrids the leakage is now independent from pk and ct^*, we are in the same situation than in Kyber's reduction when distinguishing between games G_1 and G_2 [BDK+18, Thm. 2]. This concludes the proof. ■

□

6.2 Our NIZK Instantiation

As required in Theorem 1, to lift the probing $\mathsf{IND\text{-}CPA}$ secure encryption scheme Rutile from Section 4 to an $\mathsf{IND\text{-}CCA}$ secure encryption scheme, which we call Topaz, we need a specific NIZK for proving "joint encryption". More precisely, it needs to prove that two Rutile ciphertexts are encryptions under different public keys of a same message with overwhelming probability over the distribution of the key pair generation. To this extent we use the lattice-based Fiat-Shamir transformed NIZK argument system in the ROM from [LNP22, Sec. 5.2]. Informally, the NIZK needs to prove linear equations (for the ciphertext components $\mathbf{v}$ and v), that the encryption randomness fulfills a norm bound and that the message is a binary vector. We provide the concrete instantiation in Appendix D.2.

Zero-knowledge of the LNP NIZK is proven under the hardness assumption of M-SIS and knowledge soundness is proven under extended M-LWE a variant of M-LWE, see [LNP22, Sec. 2] for more details. Hence, our overall scheme depends on structured lattice hardness assumptions only. Note that all proofs are in the ROM. Extending it to the QROM is, as far as we are aware of, an open research question going beyond the scope of the present work.

Note that [LNP22] do not prove Π to be one-time simulation sound, which we need for the NY-transform. Recently, Lipmaa [Lip25] has shown that any Fiat-Shamir transformed proof system, which is zero-knowledge, knowledge sound, and fulfills the weak unique response property, is already one-time simulation sound. The original theorem even proves the stronger property of simulation extractability, which implies (one-time) simulation soundness. We argue in Corollary 2 in Appendix D.2 that, assuming M-SIS, [LNP22] has the weak unique response property and hence by Lipmaa [Lip25] our NIZK is one-time simulation sound. On an intuitive level, we observe that the [LNP22] simulator only programs the random oracle in the last round, hence it is enough to prove weak unique response with respect to this last round. Furthermore, the proven statement in the last round is a standard matrix-vector multiplication with a short vector. Thus, breaking the unique response property would imply a M-SIS solver.

6.3 Putting Everything Together

We have now all the pieces to put everything together for constructing our probing IND-CCA secure encryption scheme, which we call Topaz. For completeness, we provide Topaz detailed algorithms in Appendix D.3.

Recall from Theorem 1 that we need two things. First, an encryption scheme Σ which is correct and probing IND-CPA secure, whose decryption is NIo and which is NY-compatible for RandSpace and RandDistr. Second, a non-interactive proof system Π for the relation R_Σ which is complete, zero-knowledge and one-time simulation sound.

For the former, we described a lattice-based instantiation of Σ in Section 4, proved it probing IND-CPA secure under standard M-LWE in Theorem 2 and showed its decryption algorithm to be NIo in Lemma 9. Moreover, we proved it NY-compatible in Lemma 8.

For the latter, we argued in Section 6.2 that the lattice-based argument system from [LNP22] is expressive enough to prove R_Σ and is a one-time simulation sound NIZK.

6.4 Estimates for Rutile and Topaz

In this section, we provide non-optimized sets of parameters for Rutile and Topaz and put them in relation with Kyber/ML-KEM parameters [SAB+22] in Table 1. We expect all sizes to be reducible, for instance using ciphertext compression or better encoding strategies for the secret key.

Rutile *Parameters:* For the public key we consider $\mathsf{pk} = (\mathbf{t}, \mathsf{seed})$ instead of $\mathsf{pk} = (\mathbf{t}, \mathbf{A})$. Thus, $|\mathsf{pk}| = nk \cdot \lceil \log_2 q \rceil + \kappa$, $|\mathsf{sk}| = nk \cdot \lceil \log_2(d\,\mathrm{rep}) \rceil$ and $|\mathsf{ct}| = (k+1)n \cdot \lceil \log_2 q \rceil$, where $|\cdot|$ denotes the bit size.

An attractive feature of our masking-friendly encryption scheme is that the ring degree $n = 256$, the matrix dimension $k = 3$, the bound for the sum of uniforms $u = 2$ and the chosen 24-bit modulus $q = 11017217$ are independent from the masking level (number of shares). The only parameters that change are the number of shares d and the parameter rep, which are chosen such that the product $d \cdot \mathrm{rep} = 2^5$ remains constant. Note that by the correctness Lemma 14, we require $q > 4 \cdot 2^{2u} \cdot d\,\mathrm{rep} \cdot n(6k+3)$. This restriction depends on the maximal masking level d we want to support. For q, we choose the smallest which fully splits in $\mathcal{R}$, that is $q = 1 \bmod 2n$. Overall, the size of the secret/public keys and that of the cipher text remain constant as d increases.

Using these parameters, we estimate the M-LWE security level using the Lattice Estimator [APS15] with zero-centered secret and error distributions of variance given by $\sigma = 2^u\sqrt{l/6}$, where $l = d\,(\mathrm{rep} - 1) + 1$ the latter coming from Theorem 2. Note that σ slowly decreases as d increases. Therefore, the minimum security level $\kappa = 138$ corresponds to the case $d = 8$.

Topaz *Parameters:* Note that a secret key of Topaz is exactly a secret key of Rutile. The size of the public keys of Topaz is doubled as we need to store two public keys of Rutile. And finally, the resulting ciphertext size is twice the size of Rutile ciphertexts plus the NIZK proof size.

To estimate the size of the NIZK proof, we adapted the sage script for [LNP22, Sec. 6.3] accessible at https://github.com/khanhcrypto/LBZKP/blob/main/ver_enc_param_set_1.sage to our joint encryption setting. We use the formula of [LNP22, Sec. 6.3] and our norm bound for encryption randomness B to obtain the lower bound of 53 on the bit-size of the NIZK modulus p. The final security level of Topaz is the minimum of the security of Rutile and the used NIZK, the latter being set to 128. One can clearly see that ca. 82% of Topaz's ciphertext size comes from the NIZK and hence any future improvements on lattice-based NIZK sizes would help improving its efficiency.

Table 1: Communication Costs for Rutile and Topaz, compared with ML-KEM.

	$\|\mathsf{pk}\|$ (KB)	$\|\mathsf{ct}\|$ (KB)	Security
Rutile (IND-CPA)	2.32	3.07	138-bit estimated
Topaz	4.64	33.40	128-bit estimated
ML-KEM-512	0.80	0.768	NIST Cat. 1
ML-KEM-768	1.184	1.088	NIST Cat. 3

6.5 Comparison with Masked Kyber

Several approaches have been proposed for the masking of Kyber, in particular for $d > 2$ [BGR+21, BC22, CGMZ22, CGMZ23]. Yet, both Topaz and masked Kyber have (at least) an $O(d^2)$ masking overhead in the number d of shares, due to the 1-bit compression required for message decoding. The difference between the two lies in the number of secret-dependent nonlinear operations. For our $k = 3$, $n = 256$ parameters, after the public NIZK verification Topaz decryption only requires $3d$ public-by-shared ring multiplications and one 1-bit compression, i.e., 256 coefficient-wise compressions. In contrast, the FO-based decryption of Kyber768 additionally requires the masked evaluation of a hash function G, the generation of $2k + 1 = 7$ masked noise polynomials, $12d$ additional public-by-shared ring multiplications for re-encryption, mask conversions for the encoded message, and a masked comparison of $4{\cdot}256 = 1024$ ciphertext coefficients [BC22, BGR+21, CGMZ23, CGMZ22]. Thus, while the asymptotic masking complexity remains quadratic for both constructions, Topaz isolates the quadratic secret-dependent nonlinearity to the underlying message-decoding operation; its CCA-validity check itself is public and independent of d. Table 2 summarizes this comparison. Moreover, key generation is quasi-linear in the number of shares d for Topaz (due to the AddRepNoise gadget), whereas masked centered binomial distribution sampling in Kyber768 has quadratic complexity in d.

Table 2: Comparison of secret-dependent non-linear operations during decryption between Topaz and masked Kyber768.

Operation	Topaz	Kyber768 (masked)
message decoding	256 shared 1-bit compressions	256 shared 1-bit compressions
hash evaluation	none	masked hash function G
noise generation	none	7 noise polynomials
re-encryption	none	$12d$ ring products
message conversion	none	256 coefficients
CCA validity check	public NIZK verification	masked comparison of 1024 coefficients

Beyond masking costs. The comparison above only captures the *quantitative* complexity of masking the respective computations. There is also a *qualitative* difference in their side-channel attack surface. In an FO-based construction, the value obtained from decryption is subsequently processed through secret-dependent hashing, re-encryption, and ciphertext comparison, creating additional intermediate values that may become targets of physical side-channel attacks. In contrast, the CCA validity check of Topaz consists of a public NIZK verification, and only the underlying base decryption operates on secret-shared data. Thus, the NY transform structurally reduces the amount of secret-dependent computation exposed during decapsulation. We stress, however, that probing security is only an abstraction of physical leakage, and establishing whether this translates into improved resistance against concrete side-channel attacks requires an implementation and experimental evaluation.

Acknowledgments

We thank our Raccoon discussion group, especially Thomas Roche, for their helpful feedback. We further thank Akira Takahashi for helpful discussions on simulation soundness and Guilhem Niot for finding a small issue in our previous simulation extractability proof. This work was supported by the French government through the National Research Agency, under the projects ANR-21-ASTR-0016 *AMIRAL*, ANR-22-PECY-003 *SecureCompute*, and ANR-25-CE39-4214-01 *RELATE*.

References

AASA+20. G. Alagic, J. Alperin-Sheriff, D. Apon, D. Moody, et al. Status report on the second round of the NIST post-quantum cryptography standardization process. Technical Report NISTIR 8309, National Institute of Standards and Technology, July 2020. Available from the NIST website.

ABC+24. G. Alagic, M. Bros, P. Ciadoux, D. Cooper, Q. Dang, T. Dang, J. Kelsey, J. Lichtinger, C. Miller, D. Moody, R. Peralta, R. Perlner, A. Robinson, H. Silberg, D. Smith-Tone, N. Waller, and Y.-K. Liu. Status report on the first round of the additional digital signature schemes for the NIST post-quantum cryptography standardization process. Technical Report NISTIR 8528, National Institute of Standards and Technology, October 2024. "Final" version, published October 24, 2024.

AGV09. A. Akavia, S. Goldwasser, and V. Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In O. Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 474–495. Springer, Berlin, Heidelberg, March 2009.

AKSvV22. M. Azouaoui, Y. Kuzovkova, T. Schneider, and C. van Vredendaal. Post-quantum authenticated encryption against chosen-ciphertext side-channel attacks. *IACR TCHES*, 2022(4):372–396, 2022.

APS15. M. R. Albrecht, R. Player, and S. Scott. On the concrete hardness of learning with errors. *J. Math. Cryptol.*, 9(3):169–203, 2015.

BBD+16. G. Barthe, S. Belaïd, F. Dupressoir, P.-A. Fouque, B. Grégoire, P.-Y. Strub, and R. Zucchini. Strong non-interference and type-directed higher-order masking. In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, editors, *ACM CCS 2016*, pages 116–129. ACM Press, October 2016.

BBE+18. G. Barthe, S. Belaïd, T. Espitau, P.-A. Fouque, B. Grégoire, M. Rossi, and M. Tibouchi. Masking the GLP lattice-based signature scheme at any order. In J. B. Nielsen and V. Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 354–384. Springer, Cham, April / May 2018.

BBM26. J. Béguinot, G. Brian, and L. Masure. Bypassing the random-probing model in masking security proofs. Cryptology ePrint Archive, Paper 2026/288, 2026.

BC22. O. Bronchain and G. Cassiers. Bitslicing arithmetic/boolean masking conversions for fun and profit with application to lattice-based KEMs. *IACR TCHES*, 2022(4):553–588, 2022.

BCRT23. S. Belaïd, G. Cassiers, M. Rivain, and A. R. Taleb. Unifying freedom and separation for tight probing-secure composition. In H. Handschuh and A. Lysyanskaya, editors, *CRYPTO 2023, Part III*, volume 14083 of *LNCS*, pages 440–472. Springer, Cham, August 2023.

BDK+18. J. W. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé. CRYSTALS - Kyber: A CCA-secure module-lattice-based KEM. In *2018 IEEE European Symposium on Security and Privacy*, pages 353–367. IEEE Computer Society Press, April 2018.

BGR+21. J. W. Bos, M. Gourjon, J. Renes, T. Schneider, and C. van Vredendaal. Masking Kyber: First- and higher-order implementations. *IACR TCHES*, 2021(4):173–214, 2021.

BGV12. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In S. Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012.

BJRW23. K. Boudgoust, C. Jeudy, A. Roux-Langlois, and W. Wen. On the hardness of module learning with errors with short distributions. *Journal of Cryptology*, 36(1):1, January 2023.

BJTW26. K. Boudgoust, C. Jeudy, E. Tairi, and W. Wen. Hardness of M-LWE with general distributions and applications to leaky variants. *PKC*, 2026. To appear in the proceedings of PKC'26.

BRR25. S. Belaïd, M. Rivain, and M. Rossi. New techniques for random probing security and application to raccoon signature scheme. In S. Fehr and P.-A. Fouque, editors, *EUROCRYPT 2025, Part VIII*, volume 15608 of *LNCS*, pages 94–123. Springer, Cham, May 2025.

CGMZ21. J.-S. Coron, F. Gérard, S. Montoya, and R. Zeitoun. High-order table-based conversion algorithms and masking lattice-based encryption. Cryptology ePrint Archive, Report 2021/1314, 2021.

CGMZ22. J.-S. Coron, F. Gérard, S. Montoya, and R. Zeitoun. High-order table-based conversion algorithms and masking lattice-based encryption. *IACR TCHES*, 2022(2):1–40, 2022.

CGMZ23. J.-S. Coron, F. Gérard, S. Montoya, and R. Zeitoun. High-order polynomial comparison and masking lattice-based encryption. *IACR TCHES*, 2023(1):153–192, 2023.

CGTZ23. J.-S. Coron, F. Gérard, M. Trannoy, and R. Zeitoun. Improved gadgets for the high-order masking of Dilithium. *IACR TCHES*, 2023(4):110–145, 2023.

DDF14. A. Duc, S. Dziembowski, and S. Faust. Unifying leakage models: From probing attacks to noisy leakage. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 423–440. Springer, Berlin, Heidelberg, May 2014.

DDF19. A. Duc, S. Dziembowski, and S. Faust. Unifying leakage models: From probing attacks to noisy leakage. *Journal of Cryptology*, 32(1):151–177, January 2019.

dEK+23. R. del Pino, T. Espitau, S. Katsumata, M. Maller, F. Mouhartem, T. Prest, M. Rossi, and M. Saarinen. Raccoon. Technical report, National Institute of Standards and Technology, 2023. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures.

DKM+24. R. Del Pino, S. Katsumata, M. Maller, F. Mouhartem, T. Prest, and M.-J. O. Saarinen. Threshold raccoon: Practical threshold signatures from standard lattice assumptions. In M. Joye and G. Leander, editors, *EUROCRYPT 2024, Part II*, volume 14652 of *LNCS*, pages 219–248. Springer, Cham, May 2024.

dPEK+23. R. del Pino, T. Espitau, S. Katsumata, M. Maller, F. Mouhartem, T. Prest, M. Rossi, and M.-J. Saarinen. Raccoon: A side-channel secure signature scheme, 2023.

dPKPR24. R. del Pino, S. Katsumata, T. Prest, and M. Rossi. Raccoon: A masking-friendly signature proven in the probing model. In L. Reyzin and D. Stebila, editors, *CRYPTO 2024, Part I*, volume 14920 of *LNCS*, pages 409–444. Springer, Cham, August 2024.

EEN+24. M. F. Esgin, T. Espitau, G. Niot, T. Prest, A. Sakzad, and R. Steinfeld. Plover: Masking-friendly hash-and-sign lattice signatures. In M. Joye and G. Leander, editors, *EUROCRYPT 2024, Part VII*, volume 14657 of *LNCS*, pages 316–345. Springer, Cham, May 2024.

FRW25. T. Feneuil, M. Rivain, and A. Warmé-Janville. Masking-friendly post-quantum signatures in the threshold-computation-in-the-head framework. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2025(4):667–710, 2025.

HLM+23. C. Hoffmann, B. Libert, C. Momin, T. Peters, and F.-X. Standaert. POLKA: Towards leakage-resistant post-quantum CCA-secure public key encryption. In A. Boldyreva and V. Kolesnikov, editors, *PKC 2023, Part I*, volume 13940 of *LNCS*, pages 114–144. Springer, Cham, May 2023.

HPA21. J. Howe, T. Prest, and D. Apon. SoK: How (not) to design and implement post-quantum cryptography. In K. G. Paterson, editor, *CT-RSA 2021*, volume 12704 of *LNCS*, pages 444–477. Springer, Cham, May 2021.

HPS98. J. Hoffstein, J. Pipher, and J. H. Silverman. NTRU: A ring-based public key cryptosystem. In *Third Algorithmic Number Theory Symposium (ANTS)*, volume 1423 of *LNCS*, pages 267–288. Springer, June 1998.

ISW03. Y. Ishai, A. Sahai, and D. Wagner. Private circuits: Securing hardware against probing attacks. In D. Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 463–481. Springer, Berlin, Heidelberg, August 2003.

KJJ99. P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 388–397. Springer, Berlin, Heidelberg, August 1999.

Koc96. P. C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In N. Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 104–113. Springer, Berlin, Heidelberg, August 1996.

KR19. Y. T. Kalai and L. Reyzin. A survey of leakage-resilient cryptography. In *Providing Sound Foundations for Cryptography*, pages 727–794. ACM, 2019.

Lin06. Y. Lindell. A simpler construction of CCA2-secure public-key encryption under general assumptions. *Journal of Cryptology*, 19(3):359–377, July 2006.

Lip25. H. Lipmaa. Plonk is simulation extractable in ROM under falsifiable assumptions. In B. Applebaum and H. R. Lin, editors, *TCC 2025, Part IV*, volume 16271 of *LNCS*, pages 3–36. Springer, Cham, December 2025.

LNP22. V. Lyubashevsky, N. K. Nguyen, and M. Plançon. Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. In Y. Dodis and T. Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 71–101. Springer, Cham, August 2022.

LPR13. V. Lyubashevsky, C. Peikert, and O. Regev. A toolkit for ring-LWE cryptography. In T. Johansson and P. Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 35–54. Springer, Berlin, Heidelberg, May 2013.

LS15. A. Langlois and D. Stehlé. Worst-case to average-case reductions for module lattices. *DCC*, 75(3):565–599, 2015.

McE78. R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. The deep space network progress report 42-44, Jet Propulsion Laboratory, California Institute of Technology, January/February 1978. https://ipnpr.jpl.nasa.gov/progress_report2/42-44/44N.PDF.

NS09. M. Naor and G. Segev. Public-key cryptosystems resilient to key leakage. In S. Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 18–35. Springer, Berlin, Heidelberg, August 2009.

NY90. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pages 427–437. ACM Press, May 1990.

oST22. N. I. of Standards and Technology. Call for additional digital signature algorithms for the NIST post-quantum cryptography standardization process. Technical report, National Institute of Standards and Technology,

September 2022. Call for proposals for additional post-quantum signature algorithms (Round 4).

Pat96. J. Patarin. Hidden fields equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms. In U. M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 33–48. Springer, Berlin, Heidelberg, May 1996.

PPS26a. D. Pay, T. Peters, and F.-X. Standaert. In mid-stream: Removing the FO-transform helps against leakage but is not enough. Cryptology ePrint Archive, Paper 2026/157, 2026.

PPS26b. D. Pay, T. Peters, and F.-X. Standaert. Leakage-resilient post-quantum encryption from faithful and efficient physical simulation. Springer-Verlag, 2026.

SAB+22. P. Schwabe, R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, G. Seiler, D. Stehlé, and J. 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.

Sho94. P. W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th FOCS*, pages 124–134. IEEE Computer Society Press, November 1994.

UXT+22. R. Ueno, K. Xagawa, Y. Tanaka, A. Ito, J. Takahashi, and N. Homma. Curse of re-encryption: A generic power/EM analysis on post-quantum KEMs. *IACR TCHES*, 2022(1):296–322, 2022.

Appendix A Missing Proofs of Section 2

A.1 Strong Non-Interference Definition

We provide for completeness a stronger version of non-interference.

Definition 15 (sNI/sNIu property, [BBD+16, dPKPR24]). *Let $d \in \mathbb{N}^*$ and* Prog *be a program with input wires $\overline{\mathcal{X}} = (\overline{\mathcal{X}}_{masked}, \overline{\mathcal{X}}_{unmasked})$ and output wires $\overline{\mathcal{Y}}$. We say that* Prog *is* (d-1)*-Strongly Non-Interfering with Unshared inputs (*(d-1)-sNIu *for short) if and only if it meets the same criteria than to be* NIu*, except that for any set of probes $\overline{\mathcal{P}}$, upper bound on* SimProb *outputs is $\left|\overline{\mathcal{P}} \setminus \overline{\mathcal{Y}}\right|$ instead of $\left|\overline{\mathcal{P}}\right|$. When $\overline{\mathcal{X}}_{unmasked} = \emptyset$, we simply call* Prog (d-1)*-Strongly Non-Interfering (*(d-1)-sNI *for short).*

A.2 Proof of Lemma 1

Proof. We have by triangular inequality that $\|\mathbf{a}^t\mathbf{b}\|_\infty \leq \sum_{i=1}^k \|\mathbf{a}_i \cdot \mathbf{b}_i\|_\infty$. The polynomial product $\mathbf{a}_i \cdot \mathbf{b}_i$ can be represented as a matrix-vector product and applying the Cauchy-Schwarz inequality on every row of the matrix independently gives $\|\mathbf{a}_i \cdot \mathbf{b}_i\|_\infty \leq \|\mathbf{a}_i\|_2 \cdot \|\mathbf{b}_i\|_2$. Here, the shape of power-of-two cyclotomics is important as it yields nega-cyclic matrices which do not increase the norms. Thus, we have $\|\mathbf{a}^t\mathbf{b}\|_\infty \leq \sum_{i=1}^k \|\mathbf{a}_i\|_2 \cdot \|\mathbf{b}_i\|_2$. This sum itself is an inner product. Applying Cauchy-Scwharz inequality a second time thus yields the proof. □

A.3 Proof of Proposition 1

Proof. "⇒" We start by describing the reduction.

Let $\mathcal{A}$ be an adversary in the M-LWE experiment and let $\mathcal{B}$ be an adversary in the experiment of *(d-1)-probing* M-LWE from Figure 1. Let $\mathcal{A}$ be given as input $(\mathbf{A}, \mathbf{b})$. They forward $\mathbf{A}$ as first input to $\mathcal{B}$. Upon receiving $\overline{\mathcal{P}} \subsetneq \{1, \ldots, l\}$ as the probe indices requested by $\mathcal{B}$, the adversary $\mathcal{A}$ verifies that $\left|\overline{\mathcal{P}}\right| < d$, else they abort the experiment. Let $\widetilde{\overline{\mathcal{P}}}$ denote an arbitrary subset of $\{1, \ldots, l\}$ of size exactly $d-1$ containing $\overline{\mathcal{P}}$. Then, for all $i \in \widetilde{\overline{\mathcal{P}}}$, the adversary $\mathcal{A}$ samples $\widetilde{\boldsymbol{\rho}}_i^{(s)} \leftarrow\!\!\$\ \mathsf{Unif}(\mathcal{R}, k, u)$ and $\widetilde{\boldsymbol{\rho}}_i^{(e)} \leftarrow\!\!\$\ \mathsf{Unif}(\mathcal{R}, k, u)$ and sets

$$\widetilde{\mathbf{b}} = \mathbf{b} + \mathbf{A}\left(\sum_{i \in \widetilde{\overline{\mathcal{P}}}} \widetilde{\boldsymbol{\rho}}_i^{(s)}\right) + \left(\sum_{i \in \widetilde{\overline{\mathcal{P}}}} \widetilde{\boldsymbol{\rho}}_i^{(e)}\right) \quad \text{and} \quad \mathcal{L} = \left(\widetilde{\boldsymbol{\rho}}_i^{(s)}, \widetilde{\boldsymbol{\rho}}_i^{(e)}\right)_{i \in \overline{\mathcal{P}}}$$

The reduction $\mathcal{A}$ then gives $(\widetilde{\mathbf{b}}, \mathcal{L})$ as input to $\mathcal{B}$. Upon receiving the guess b' from $\mathcal{B}$, the adversary $\mathcal{A}$ forwards b' as their own guess.

We now argue that the advantage of $\mathcal{A}$ winning the $\text{M-LWE}_{\mathcal{R}_q,k,(l-d+1),u}$ experiment is the same as that of $\mathcal{B}$ winning the *(d-1)-probing* $\text{M-LWE}_{\mathcal{R}_q,d,k,l,u}$ experiment. First, note that the matrix $\mathbf{A}$ has the same distribution in both experiments. If $b = 0$ in the M-LWE experiment, we know that $\mathbf{b}$ can be written as $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}$, where both $\mathbf{s}$ and $\mathbf{e}$ are composed of $l - d + 1$ summands. Given

that $\left|\{1,\dots,l\} \setminus \widetilde{\overline{\mathcal{P}}}\right| = l - d + 1$, we can rearrange and rename the $l - d + 1$ summands of $\mathbf{s}$ and $\mathbf{e}$ to $\left(\widetilde{\boldsymbol{\rho}}_i^{(s)}, \widetilde{\boldsymbol{\rho}}_i^{(e)}\right)_{i \in \{1,\dots,l\} \setminus \widetilde{\overline{\mathcal{P}}}}$. Now, it yields

$$\widetilde{\mathbf{b}} = \mathbf{A}\mathbf{s} + \mathbf{e} + \mathbf{A}\left(\sum_{i \in \widetilde{\overline{\mathcal{P}}}} \widetilde{\boldsymbol{\rho}}_i^{(s)}\right) + \left(\sum_{i \in \widetilde{\overline{\mathcal{P}}}} \widetilde{\boldsymbol{\rho}}_i^{(e)}\right) = \mathbf{A}\left(\sum_{i=1}^{l} \widetilde{\boldsymbol{\rho}}_i^{(s)}\right) + \left(\sum_{i=1}^{l} \widetilde{\boldsymbol{\rho}}_i^{(e)}\right)$$

Hence, $(\widetilde{\mathbf{b}}, \mathcal{L})$ has the correct distribution for the case $b = 0$ in the probing experiment. On the other hand, if $b = 1$, $\mathbf{b}$ is uniform, implying the uniformity of $\widetilde{\mathbf{b}}$. Again, $(\widetilde{\mathbf{b}}, \mathcal{L})$ has the correct distribution for the case $b = 1$ in the probing experiment. Overall, the cases $b = 0$ and the cases $b = 1$ match in both experiments, implying that the advantage is preserved.

"$\Leftarrow$" As before, we begin with describing the reduction. Let $\mathcal{B}$ be an adversary in the *(d-1)-probing* M-LWE experiment from Figure 1 and let $\mathcal{A}$ be an adversary in the experiment of M-LWE. Let $\mathcal{B}$ be given as first input $\mathbf{A}$. They output $\overline{\mathcal{P}} = \{l - d + 2, \dots, l\}$ which is a set of size $d-1$. Let $(\mathbf{b}, \mathcal{L})$ be the second input given to $\mathcal{B}$, where $\mathcal{L} = (\boldsymbol{\rho}_i^{(s)}, \boldsymbol{\rho}_i^{(e)})_{i \in \overline{\mathcal{P}}}$. They set $\widetilde{\mathbf{b}} = \mathbf{b} - \mathbf{A}\left(\sum_{i \in \overline{\mathcal{P}}} \boldsymbol{\rho}_i^{(s)}\right) - \left(\sum_{i \in \overline{\mathcal{P}}} \boldsymbol{\rho}_i^{(e)}\right)$ and provide $(\mathbf{A}, \widetilde{\mathbf{b}})$ to $\mathcal{A}$. Upon receiving the guess b', the adversary $\mathcal{B}$ forwards b' as their own guess.

We argue that the advantage of $\mathcal{B}$ winning the *(d-1)-probing* M-LWE$_{\mathcal{R}_q,d,n,l,u}$ experiment is the same as the advantage of $\mathcal{A}$ winning the corresponding experiment of M-LWE$_{\mathcal{R}_q,k,(l-d+1),u}$. First, note that the matrix $\mathbf{A}$ has the same distribution in both experiments. If $b = 0$ in the *(d-1)-probing* M-LWE experiment,

$$\widetilde{\mathbf{b}} = \mathbf{A}\mathbf{s} + \mathbf{e} - \mathbf{A}\left(\sum_{i \in \overline{\mathcal{P}}} \boldsymbol{\rho}_i^{(s)}\right) - \left(\sum_{i \in \overline{\mathcal{P}}} \boldsymbol{\rho}_i^{(e)}\right) = \mathbf{A}\left(\sum_{i=1}^{l-d+1} \boldsymbol{\rho}_i^{(s)}\right) + \left(\sum_{i=1}^{l-d+1} \boldsymbol{\rho}_i^{(e)}\right)$$

Hence, $(\mathbf{A}, \widetilde{\mathbf{b}})$ has the correct distribution for the case $b = 0$ in the M-LWE experiment. On the other hand, if $b = 1$, $\mathbf{b}$ is uniform, implying the uniformity of $\widetilde{\mathbf{b}}$. Again, $(\mathbf{A}, \widetilde{\mathbf{b}})$ has the correct distribution for the case $b = 1$ in the M-LWE experiment. Overall, the cases $b = 0$ and the cases $b = 1$ match in both experiments, implying that the advantage is preserved. □

A.4 Proof of Proposition 2

Proof. For simplicity we note $\overline{\mathcal{Z}} := \overline{\mathcal{Y}_2}$, $\overline{\mathcal{Y}} := \overline{\mathcal{Y}_1} = \overline{\mathcal{X}}_1$ and $\overline{\mathcal{X}} := \overline{\mathcal{X}}_1$, and $\mathcal{Z}$ [resp. $\mathcal{Y}$, $\mathcal{X}$] the values carried by $\overline{\mathcal{Z}}$ [resp. $\overline{\mathcal{Y}}$, $\overline{\mathcal{X}}$]. Also, we note $\overline{\mathcal{Y}_I} = \mathsf{SimProb}_2(\overline{\mathcal{P}}_2)$ [resp. $\overline{\mathcal{Y}_{\bar{I}}} = \overline{\mathcal{Y}} \setminus \mathsf{SimProb}_2(\overline{\mathcal{P}}_2)$] and $\mathcal{Y}_I$ [resp. $\mathcal{Y}_{\bar{I}}$] the values they carry.

Let $\mathsf{SimProb}_1$, $\mathsf{SimLeak}_1$ be the NIu simulators for Prog_1 on one hand and $\mathsf{SimProb}_2$, $\mathsf{SimLeak}_2$ the NIo simulators for Prog_2 on the other hand. In the following proof, we note $\overline{\mathcal{P}} = (\overline{\mathcal{P}}_1 \cup \overline{\mathcal{P}}_2)$ the set of probes queried on Prog, with $\overline{\mathcal{P}}_1$ probes on Prog_1 computations and $\overline{\mathcal{P}}_2$ probes on Prog_2 computations. By convention, we force $\overline{\mathcal{P}}_1 \cap \overline{\mathcal{P}}_2 = \emptyset$. We note $\mathcal{L}_1$ the values carried by $\overline{\mathcal{P}}_1$ wires and $\mathcal{L}_2$ the values carried by $\overline{\mathcal{P}}_2$ wires.

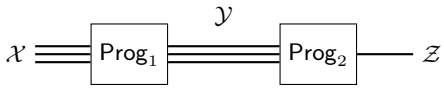

Fig. 8: DAG representing the structure of Prog

We prove the result by describing the two simulators SimProb and SimLeak and proving they are NIuo simulators for Prog.

Algorithm 7 SimProb

Input: Sets of probes $\overline{\mathcal{P}}_1, \overline{\mathcal{P}}_2$ on Prog_1 and Prog_2
Output: $\overline{\mathcal{X}}' \subset \overline{\mathcal{X}}$ respecting Def. 11 properties
1: $\overline{\mathcal{Y}_I} \leftarrow \mathsf{SimProb}_2(\overline{\mathcal{P}}_2)$
2: $\overline{\mathcal{X}}' \leftarrow \mathsf{SimProb}_1(\overline{\mathcal{P}}_1 \cup \overline{\mathcal{Y}_I})$

We show that SimProb match the criteria of the NIuo definition. Let $\overline{\mathcal{P}} = \overline{\mathcal{P}}_1 \cup \overline{\mathcal{P}}_2$ with $\overline{\mathcal{P}} < d$. We note $\overline{\mathcal{X}}'_{masked}$ [resp. $\overline{\mathcal{X}}'_{unmasked}$] wires of $\overline{\mathcal{X}}'$ carrying masked [resp. unmasked] values. We have that:

- $\overline{\mathcal{X}}'_{masked} \subset \overline{\mathcal{X}}_{masked}$ and $\overline{\mathcal{X}}'_{unmasked} \subset \overline{\mathcal{X}}_{unmasked}$ by hypothesis on $\mathsf{SimProb}_1$
- $\left|[\![\overline{x_1}]\!] \cap \overline{\mathcal{X}}'_{masked}\right| \leq \left|\overline{\mathcal{P}}_1 \cup \overline{\mathcal{Y}_I}\right| \leq \left|\overline{\mathcal{P}}_1\right| + \left|\overline{\mathcal{Y}_I}\right| \leq \left|\overline{\mathcal{P}}_1\right| + \left|\overline{\mathcal{P}}_2\right| = \left|\overline{\mathcal{P}}\right|$, where first inequality comes from hypothesis on $\mathsf{SimProb}_1$, second comes from triangular inequality and last one comes from hypothesis on $\mathsf{SimProb}_2$.
- $\left|\overline{\mathcal{X}}'_{unmasked}\right| \leq \left|\overline{\mathcal{P}}_1 \cup \overline{\mathcal{Y}_I}\right| \leq \left|\overline{\mathcal{P}}_1\right| + \left|\overline{\mathcal{Y}_I}\right| \leq \left|\overline{\mathcal{P}}_1\right| + \left|\overline{\mathcal{P}}_2\right| = \left|\overline{\mathcal{P}}\right|$, where first inequality comes from hypothesis on $\mathsf{SimProb}_1$, second comes from triangular inequality and last one comes from hypothesis on $\mathsf{SimProb}_2$.

Thus SimProb meets the NIuo criteria.

Algorithm 8 SimLeak

Input: Public output $\mathcal{Y}_{public}$, some input values $\mathcal{X}'$ and set of probes $\overline{\mathcal{P}} = (\overline{\mathcal{P}}_1 \cup \overline{\mathcal{P}}_2)$
Output: l_1, l_2 simulated values for $\overline{\mathcal{P}}_1, \overline{\mathcal{P}}_2$
1: $\overline{\mathcal{Y}_I} \leftarrow \mathsf{SimProb}_2(\overline{\mathcal{P}}_2)$
2: $l_1, y_I \leftarrow \mathsf{SimLeak}_1(\mathcal{X}', \overline{\mathcal{P}}_1 \cup \overline{\mathcal{Y}_I})$ ▷ y_I simulates values carried by $\overline{\mathcal{Y}_1}'$
3: $l_2 \leftarrow \mathsf{SimLeak}_2(y_I, \mathcal{Y}_{public}, \overline{\mathcal{P}}_2)$

We now show that SimLeak meets the criteria of the NIuo definition. We first state a claim on the $\mathsf{SimLeak}_1$ and $\mathsf{SimLeak}_2$ simulators.

Claim. Let l_1, l_2 values $\overline{\mathcal{P}}_1$ and $\overline{\mathcal{P}}_2$ could carry. For any y_I and $y_{\bar{I}}$ some values $\overline{\mathcal{Y}_I}$ and $\overline{\mathcal{Y}_{\bar{I}}}$ could carry, any input values $\mathcal{X}$ and any public output values $\mathcal{Z}$, we have the two following equalities:

$$\Pr(\mathcal{L}_1 = l_1, \mathcal{Y}_I = y_I \mid \mathcal{X}, \mathcal{Z}) = \Pr(\mathsf{SimLeak}_1() = (l_1, y_I) \mid \mathcal{X}, \mathcal{Z})$$

$$\Pr(\mathcal{L}_2 = l_2 \mid \mathcal{X}, \mathcal{Z}, y_I, y_{\bar{I}}, l_1) = \Pr(\mathsf{SimLeak}_2() = l_2 \mid \mathcal{X}, \mathcal{Z}, \mathsf{SimLeak}_1() = (l_1, y_I))$$

where we abbreviate $\mathsf{SimLeak}_2(y_I, \mathcal{Z}, \overline{\mathcal{P}}_2)$ into $\mathsf{SimLeak}_2()$ and $\mathsf{SimLeak}_1(\mathcal{X}', \overline{\mathcal{P}}_1 \cup \overline{\mathcal{Y}_I})$ into $\mathsf{SimLeak}_1()$ for clarity of notation.

This claim stems from the NIu/NIo properties of $\mathsf{SimLeak}_1$ and $\mathsf{SimLeak}_2$, and from the fact that $\mathcal{Z}$ is deterministic in $\mathcal{X}$. We highlight that if $\mathcal{Z}$ is not deterministic in $\mathcal{X}$ then first equality does not hold, but since it is, the dependence in $\mathcal{Z} := f(\mathcal{X})$ is already implicitly set by the dependence on $\mathcal{X}$ and the equality simply is the formulation of a NIu property for Prog_1.

We now need argue that SimLeak output has the same distribution than joint distribution $(\mathcal{L}_1, \mathcal{L}_2)$ for a given set of probes $\overline{\mathcal{P}} = \overline{\mathcal{P}}_1 \cup \overline{\mathcal{P}}_2$ (with $\overline{\mathcal{P}}_1 \cap \overline{\mathcal{P}}_2 = \emptyset$), a given input $\mathcal{X}$ and public output $\mathcal{Z} := f(\mathcal{X})$. Let l_1, l_2 values $\overline{\mathcal{P}}_1$ and $\overline{\mathcal{P}}_2$ could carry, we have:

$$\Pr(\mathcal{L}_1 = l_1, \mathcal{L}_2 = l_2 | \mathcal{X}, \mathcal{Z}) =$$

$$\sum_{y \in \mathcal{S}_{\overline{\mathcal{Y}_1}}} \Pr(\mathcal{L}_1 = l_1, \mathcal{L}_2 = l_2, \mathcal{Y} = y | \mathcal{X}, \mathcal{Z}) =$$

$$\sum_{y \in \mathcal{S}_{\overline{\mathcal{Y}_1}}} \Pr(\mathcal{L}_1 = l_1, \mathcal{Y} = y | \mathcal{X}, \mathcal{Z}) \cdot \Pr(\mathcal{L}_2 = l_2 | \mathcal{X}, \mathcal{Z}, \mathcal{L}_1 = l_1, \mathcal{Y} = y) =$$

$$\sum_{y_I \in \mathcal{S}_{\overline{\mathcal{Y}_I}}} \sum_{y_{\bar{I}} \in \mathcal{S}_{\overline{\mathcal{Y}_{\bar{I}}}}} \Pr(\mathcal{L}_1 = l_1, \mathcal{Y} = y | \mathcal{X}, \mathcal{Z}) \cdot \Pr(\mathcal{L}_2 = l_2 | \mathcal{X}, \mathcal{Z}, \mathcal{L}_1 = l_1, \mathcal{Y} = y) =$$

$$\sum_{y_I \in \mathcal{S}_{\overline{\mathcal{Y}_I}}} \sum_{y_{\bar{I}} \in \mathcal{S}_{\overline{\mathcal{Y}_{\bar{I}}}}} \Pr(\mathcal{L}_1 = l_1, \mathcal{Y} = y | \mathcal{X}, \mathcal{Z}) \cdot \Pr(\mathsf{SimLeak}_2() = l_2 \mid \mathcal{X}, \mathcal{Z}, \mathsf{SimLeak}_1() = (l_1, y_I)) =$$

$$\sum_{y_I \in \mathcal{S}_{\overline{\mathcal{Y}_I}}} \Pr(\mathsf{SimLeak}_2() = l_2 \mid \mathcal{X}, \mathcal{Z}, \mathsf{SimLeak}_1() = (l_1, y_I)) \cdot \Pr(\mathcal{L}_1 = l_1, \mathcal{Y}_I = y_I | \mathcal{X}, \mathcal{Z}) =$$

$$\sum_{y_I \in \mathcal{S}_{\overline{\mathcal{Y}_I}}} \Pr(\mathsf{SimLeak}_2() = l_2, \mathsf{SimLeak}_1() = (l_1, y_I) \mid \mathcal{X}, \mathcal{Z}) =$$

$$\Pr(\mathsf{SimLeak}_2() = l_2, \mathsf{SimLeak}_1() = (l_1, \cdot) \mid \mathcal{X}, \mathcal{Z}) =$$

$$\Pr(\mathsf{SimLeak}() = l_1, l_2 \mid \mathcal{X}, \mathcal{Z})$$

First equality comes from total probability formula, second one comes from Bayes theorem, third one is just a partition of $\overline{\mathcal{Y}}$, forth one comes from our claim,

fifth one comes from total probability formula, sixth one comes from our claim, seventh one comes from Bayes theorem, eighth one comes from total probability formula and last one is simply a rephrasing. This concludes our proof. □

A.5 Proof of Proposition 3

Proof. By assumption on Prog, we know that its public output $\mathcal{Y}_{public}$ is uniquely determined by the public input and the secret value x whose sharing we input to Prog. Let F denote the (deterministic) function such that $\forall x \in \mathcal{R},\ \mathcal{Y}_{public} := F(x)$. Now let $x \in \mathcal{R}$ a secret value and $[\![x]\!]$ a privacy-preserving sharing of x.

Since Prog is *(d-1)*-NIo, there exist SimProb and SimLeak which for every set of probes $\overline{\mathcal{P}}$ s.t. $\left|\overline{\mathcal{P}}\right| < d$, for any input $\mathcal{X} = (\mathcal{X}_{masked}, \emptyset)$ with $\mathcal{X}_{masked} \leftarrow \mathsf{KeyUpdate}([\![x]\!])$ and for public output $\mathcal{Y}_{public} = F(x)$ perfectly simulate $\overline{\mathcal{P}}$ for input $\mathcal{X} = (\mathcal{X}_{masked}, \emptyset)$ and $\mathcal{Y}_{public} := F(x)$, that is

$$\mathcal{L} \overset{i.d.}{=} \mathsf{SimLeak}(\mathcal{X}', \mathcal{Y}_{public}, \overline{\mathcal{P}})$$

where $(\cdot, \mathcal{L}) \leftarrow \mathsf{Prog}^{\mathfrak{L}}((\mathcal{X}_{masked}, \emptyset), \overline{\mathcal{P}})$ and $\mathcal{X}'$ are the values of the input wires $\overline{\mathcal{X}}' \leftarrow \mathsf{SimProb}(\overline{\mathcal{P}})$.

By definition of *(d-1)*-NIo, it holds $\overline{\mathcal{X}}' = (\overline{\mathcal{X}}'_{masked}, \emptyset)$, with $\left|\overline{\mathcal{X}}'_{masked}\right| < d$. Hence, $\mathcal{X}'$ contains at most $d-1$ values of the additive secret sharing $\mathcal{X}_{masked}$ output by $\mathsf{KeyUpdate}([\![x]\!])$.

By privacy of the additive secret sharing output by $\mathsf{KeyUpdate}([\![x]\!])$, we know that the joint distribution of $(x, \mathcal{X}')$ is identical to the joint distribution of x with $|\mathcal{X}'|$ independently sampled uniform variables over the finite ring $\mathcal{R}$, i.e.:

$$\mathcal{X}' \overset{i.d.}{=} (\gamma_i)_{i \in [|\overline{\mathcal{X}}'|]}, \text{ where } \gamma_i \leftarrow\!\!\$\ \mathcal{R} \tag{4}$$

By application of the probability preservation property to Equation 4 with the function $\mathsf{SimLeak}(\cdot, F(x), \overline{\mathcal{P}})$, we obtain

$$\mathsf{SimLeak}(\mathcal{X}', F(x), \overline{\mathcal{P}}) \overset{i.d.}{=} \mathsf{SimLeak}((\gamma_i)_{i \in [|\overline{\mathcal{X}}'|]}, F(x), \overline{\mathcal{P}})$$

Therefore, we define Sim as the PPT simulator that first calls SimProb on $\overline{\mathcal{P}}$, then samples a number of independent uniform variables in $\mathcal{R}$ equal to the size of SimProb's output, and then calls SimLeak on these independent variables, on $F(x) = \mathcal{Y}_{public}$[3] and on $\overline{\mathcal{P}}$, concluding the proof. □

[3] in practice, $\mathcal{Y}_{public}$ will be provided to the simulator

Appendix B Missing Details of Section 3

B.1 Missing Proofs of Theorem 1

Distinguishing between Hybrid_2 and Hybrid_3

Proof. For simplicity, we omit the dependency on the security parameter κ. We prove the following. Let $\mathcal{A}$ be a PPT adversary distinguishing between Hybrid_2 and Hybrid_3 with advantage at least ϵ_{ZK}, then we build a PPT adversary $\mathcal{B}$ that distinguishes between experiments $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{ZK\text{-}Real}}$ and $\mathsf{Expt}_{\mathcal{A},\mathsf{Sim}}^{\mathsf{ZK\text{-}Sim}}$ with advantage at least ϵ_{ZK}. Thus, $\mathcal{B}$ breaks the zero-knowledge property of the OTSS-NIZK, leading to a contraction.

We start by describing the reduction. Adversary $\mathcal{B}$ first receives on input from $\mathcal{A}$ a set of probes on the key generation procedure $\overline{\mathcal{P}}_{KG}$. Both in Hybrid_2 and Hybrid_3, the key generation procedure consist of two independent calls on $\mathsf{KeyGen}_{\mathsf{CPA}}^{\mathfrak{L}}$. Therefore we can split the set of probes $\overline{\mathcal{P}}_{KG}$ in two distinct sets: $\overline{\mathcal{P}}_{KG1}$ the probes on pk_1 generation and $\overline{\mathcal{P}}_{KG2}$ the probes on pk_2 generation. The adversary $\mathcal{B}$ computes the two IND-CPA public keys pk_1 and pk_2 as described both in Hybrid_2 and Hybrid_3. Leakages for $\overline{\mathcal{P}}_{KG1}$ and $\overline{\mathcal{P}}_{KG2}$ are output by $\mathsf{KeyGen}_{\mathsf{CPA}}^{\mathfrak{L}}$. We then forward the requested leakage $\mathcal{L}_{KG}$ and the public key $\mathsf{pk} = (\mathsf{pk}_1, \mathsf{pk}_2)$ to $\mathcal{A}$. The adversary $\mathcal{B}$ computes all decryption queries as described both in Hybrid_2 and Hybrid_3. After receiving m_0 and m_1 from $\mathcal{A}$, adversary $\mathcal{B}$ computes ct_1 (resp. ct_2) as an encryption of m_0 under pk_1 (resp. pk_2). It then sets $\mathsf{x} = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1, \mathsf{ct}_2)$ and $\mathsf{w} = (m, R_1, R_2)$ where $(\mathsf{x}, \mathsf{w}) \in \mathsf{R}_{\Sigma}$. Upon outputting (x, w), $\mathcal{B}$ receives a proof π in the zero-knowledge experiment for statement x. The reduction $\mathcal{B}$ then forwards the challenge ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$ to $\mathcal{A}$ and computes all following decryption queries accordingly to description of Hybrid_2 (which is the same as in Hybrid_3). The reduction $\mathcal{B}$ outputs as their guess $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{ZK\text{-}Real}}$ if $\mathcal{A}$ guessed Hybrid_2, and $\mathsf{Expt}_{\mathcal{A},\mathsf{Sim}}^{\mathsf{ZK\text{-}Sim}}$ if $\mathcal{A}$ guessed Hybrid_3.

We have that the IND-CPA public keys, the leakage on the public key, the decryption queries and the IND-CPA encryptions of m_0 on public keys pk_1 and pk_2 are perfectly simulated as they were computed accordingly to the description of Hybrid_2 and Hybrid_3, which coincide on these parts. When $\mathcal{B}$ plays the $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{ZK\text{-}Real}}$ experiment, we have that H is a truly random oracle and that the proof π is computed using the prover P, which is exactly the description of Hybrid_2. When $\mathcal{B}$ plays the $\mathsf{Expt}_{\mathcal{A},\mathsf{Sim}}^{\mathsf{ZK\text{-}Sim}}$ experiment, H and the proof are simulated, which is exactly the description of Hybrid_3.

As in both situation $\mathcal{B}$ computes either hybrid exactly as they are described, it follows that whenever $\mathcal{A}$ wins, the adversary $\mathcal{B}$ then also wins. This concludes the reduction. ■

Distinguishing between Hybrid_3 and Hybrid_4

Proof. For simplicity, we omit the dependency on the security parameter κ. Let $\mathcal{A}$ be a PPT adversary distinguishing between Hybrid_3 and Hybrid_4 with

advantage at least $\mathsf{Adv}^{\mathsf{dCPA}}_{\mathcal{A}}$, then we build a PPT adversary $\mathcal{B}$ that breaks the *(d-1)-probing* IND-CPA security of Σ with advantage at least $\mathsf{Adv}^{\mathsf{dCPA}}_{\mathcal{A}}$.

We begin by describing the reduction. First $\mathcal{B}$ receives from $\mathcal{A}$ a set of probes $\overline{\mathcal{P}}_{KG}$ on the key generation procedure. As seen in Lemma 3's proof, we can split the set of probes $\overline{\mathcal{P}}_{KG}$ in two distinct sets of probes: $\overline{\mathcal{P}}_{KG1}$ the probes on pk_1 generation and $\overline{\mathcal{P}}_{KG2}$ the probes on pk_2 generation. The adversary $\mathcal{B}$ outputs the set of probes $\overline{\mathcal{P}}_{KG2}$ as their query and receives $(\mathsf{pk}_2, \mathcal{L}_{KG2})$, an IND-CPA public key and the leakage associated to $\overline{\mathcal{P}}_{KG2}$. Adversary $\mathcal{B}$ then computes the first IND-CPA key pair $(\mathsf{pk}_1, \mathsf{sk}_1)$, and the associated leakage $\mathcal{L}_{KG1}$ using $\mathsf{KeyGen}^{\mathfrak{L}}_{\mathsf{CPA}}$. They simulate the random oracle H as described in both Hybrid_3 and Hybrid_4. They then forward the public key $\mathsf{pk} = (\mathsf{pk}_1, \mathsf{pk}_2, r)$ and the requested leakage $\mathcal{L}_{KG} = (\mathcal{L}_{KG1}, \mathcal{L}_{KG2})$ to $\mathcal{A}$. When $\mathcal{B}$ receives m_0 and m_1 as input by $\mathcal{A}$, they forward the messages as output and get a *(d-1)-probing* IND-CPA challenge ct_2 as input. They then compute ct_1 as an encryption under pk_1 of m_0, simulate a proof π for the statement $\mathsf{x} = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{ct}_1, \mathsf{ct}_2)$ and output the challenge ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$ to $\mathcal{A}$. As they know the secret key sk_1, $\mathcal{B}$ can answer all decryption queries by proceeding as described both in Hybrid_3 and Hybrid_4. The reduction $\mathcal{B}$ outputs as their guess $b^* = 0$ if $\mathcal{A}$ guesses Hybrid_3, and $b^* = 1$ if $\mathcal{A}$ guesses Hybrid_4.

All decryption queries, the random oracle H and the challenge proof are perfectly simulated as they are computed exactly as described both in Hybrid_3 and Hybrid_4. Both IND-CPA public keys are output by the same probabilistic function $\mathsf{KeyGen}^{\mathfrak{L}}_{\mathsf{CPA}}$ even though only one of those is computed by $\mathcal{B}$ and the same goes for the challenge ciphertext IND-CPA encryptions. Let b be the secret input bit to the *(d-1)-probing* IND-CPA game. When b equals 0, the challenge ciphertext output by $\mathcal{B}$ follows the exact same distribution than in Hybrid_3, and when b is equal to 1, the challenge ciphertext follows the exact same distribution than in Hybrid_4. Thus, we perfectly simulate the views of $\mathcal{A}$.

When $\mathcal{A}$ correctly guessed Hybrid_3, this means that the challenge ciphertext $\mathcal{B}$ sent was twice an encryption of m_0. Thus, $b^* = 0$ was a correct guess for $\mathcal{B}$. If $\mathcal{A}$ correctly guessed Hybrid_4, this means that the challenge ciphertext's second IND-CPA encryption ct_2 was an encryption of m_1. Thus, $b^* = 1$ was a correct guess for $\mathcal{B}$. Therefore, whenever $\mathcal{A}$ wins, $\mathcal{B}$ also does, which completes the reduction. ■

Appendix C Missing Details of Section 4

C.1 Correctness of Rutile

In this section, we prove the correctness of Rutile. Let us first recall the definition of correctness of a public-key encryption scheme.

Definition 16 (Correctness). *Let $\delta_{\mathsf{cor}}(\kappa)$ be a function parameterized by the security level κ taking real values in $[0, 1]$. A $\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ over message space $\mathcal{M}$ with randomness distribution $\mathsf{RandDistr}$ is called $\delta_{\mathsf{cor}}(\kappa)$-correct, if*

for all $\kappa \in \mathbb{N}^*$, for all $m \in \mathcal{M}$ it holds

$$\Pr_{\substack{R \leftarrow \mathsf{RandDistr} \\ (\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa)}} [\mathsf{Dec}(\mathsf{pk}, \mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m; R)) \neq m] \leq \delta_{\mathsf{cor}}(\kappa)$$

Lemma 14. *Let $\mathsf{PKE} = (\mathsf{KeyGen}_{\mathsf{CPA}}, \mathsf{Enc}_{\mathsf{CPA}}, \mathsf{Dec}_{\mathsf{CPA}})$ be the encryption scheme described in Algorithms 4-6. If $2^{2u} \cdot d\,\mathrm{rep} \cdot n(6k+3) \leq \lfloor q/4 \rfloor$, then it is δ_{cor}-correct, with $\delta_{\mathsf{cor}} = 2\exp(-2n(2k+1))$.*

Proof. As masking gadgets do not impact correctness, we can directly look at the equations in the decoded version. It holds

$$\begin{aligned}
\mu' &= v - \mathbf{v}^t \cdot \mathbf{s} \\
&= \mathbf{u}^t \cdot (\mathbf{A}\mathbf{s} + \mathbf{e}) + e_2 + \lfloor q/2 \rceil \mu - (\mathbf{u}^t \mathbf{A} + \mathbf{e}_1^t)\mathbf{s} \\
&= \lfloor q/2 \rceil \mu + \mathbf{u}^t \mathbf{e} - \mathbf{e}_1^t \mathbf{s} + e_2
\end{aligned}$$

Hence, decryption succeeds if $\|\mathbf{u}^t\mathbf{e} - \mathbf{e}_1^t\mathbf{s} + e_2\|_\infty \leq \lfloor q/4 \rfloor$. We set $\mathbf{r} = (\mathbf{u}, \mathbf{e}_1, e_2)$ which is sampled from $\mathsf{SUnif}(\mathcal{R}, 2k+1, u, d \cdot \mathrm{rep})$ and set $\mathbf{z} = (\mathbf{e}, -\mathbf{s}, 1)$, with $(\mathbf{e}, \mathbf{s})$ sampled from $\mathsf{SUnif}(\mathcal{R}, 2k, u, d \cdot \mathrm{rep})$. By Equation 1 with probability at most $\exp(-2n(2k+1))$ it holds $\|\mathbf{r}\|_2 \leq B := 2^u\sqrt{3d\,\mathrm{rep} \cdot n(2k+1)}$. Similarly, we can upper bound $\|\mathbf{z}\|_2 \leq B$. Hence, by Lemma 1 and a union bound, it holds with probability at most $2\exp(-2n(2k+1))$

$$\begin{aligned}
\left\|\mathbf{u}^t\mathbf{e} - \mathbf{e}_1^t\mathbf{s} + e_2\right\|_\infty &= \left\|\mathbf{r}^t\mathbf{z}\right\|_\infty \\
&\leq \|\mathbf{r}\|_2 \cdot \|\mathbf{z}\|_2 \\
&\leq B^2 \\
&= 2^{2u} \cdot d\,\mathrm{rep} \cdot n(6k+3)
\end{aligned}$$

□

Appendix D Missing Details of Section 6

D.1 Proof of Lemma 11

Proof. We note $\mathcal{U}$ [resp. $\mathcal{U}'$] the $d \cdot 2\mathrm{rep}$ small uniforms input to $\mathsf{KeyGen}_{\mathsf{ER}}^{\mathfrak{L}}$ [resp. $\mathsf{KeyGen}_{\mathsf{ER}}$] and $(\mathsf{pk}, \mathcal{L}_{KG})$ [resp. $(\mathsf{pk}', \mathcal{L}'_{KG})$] the answer to $\mathcal{A}$'s first query in Hybrid_2 [resp. Hybrid_3]. Let $(\mathbf{A}, \mathbf{t}) \in \mathcal{R}_q^{k \times k} \times \mathcal{R}_q^k$ be a possible public key pair and l a possible leakage value for $\mathcal{L}_{KG}$. We argue that $\Pr[\mathsf{pk} = (\mathbf{A}, \mathbf{t}), \mathcal{L}_{KG} = l] = \Pr[\mathsf{pk}' = (\mathbf{A}, \mathbf{t}), \mathcal{L}'_{KG} = l]$, where the probability is defined over the random coins of $\mathsf{SimLeak}$, $\mathsf{KeyGen}_{\mathsf{ER}}$ and the sampling of all $d \cdot 2\mathrm{rep}$ small uniforms $(\boldsymbol{\rho}_i)$.

We note $S := \{(\boldsymbol{\rho}_i) \leftarrow \mathsf{Unif}(\mathcal{R}, k, u)^{d \cdot 2\mathrm{rep}} \mid \Pr[\mathcal{U} = (\boldsymbol{\rho}_i), \mathsf{pk} = (\mathbf{A}, \mathbf{t})] \neq 0\}$, the set of all values $(\boldsymbol{\rho}_i)$ compatible with $(\mathbf{A}, \mathbf{t})$. We have that contents of the set do not change when considering pk' and $\mathcal{U}'$ instead of pk and $\mathcal{U}$, by similarities between the two hybrids.

We consider $\Pr[\mathsf{pk} = (\mathbf{A}, \mathbf{t}), \mathcal{L}_{KG} = l]$. By first applying Bayes formula on "$\mathsf{pk} = (\mathbf{A}, \mathbf{t})$" and then applying law of total probability for all values in S for $\mathcal{U}$ we have the following:

$$\Pr[\mathsf{pk} = (\mathbf{A}, \mathbf{t}), \mathcal{L}_{KG} = l]$$

$$= \Pr[\mathsf{pk} = (\mathbf{A}, \mathbf{t})] \cdot \Pr[\mathcal{L}_{KG} = l | \mathsf{pk} = (\mathbf{A}, \mathbf{t})]$$

$$= \Pr[\mathsf{pk} = (\mathbf{A}, \mathbf{t})] \cdot$$

$$\sum_{(\boldsymbol{\rho}_i) \in S} \Pr[\mathcal{U} = (\boldsymbol{\rho}_i) | \mathsf{pk} = (\mathbf{A}, \mathbf{t})] \cdot \Pr[\mathcal{L}_{KG} = l | \mathsf{pk} = (\mathbf{A}, \mathbf{t}), \mathcal{U} = (\boldsymbol{\rho}_i)]$$

Applying the same arguments and reasoning on $\Pr[\mathsf{pk}' = (\mathbf{A}, \mathbf{t}), \mathcal{L}'_{KG} = l]$ yields:

$$\Pr[\mathsf{pk}' = (\mathbf{A}, \mathbf{t}), \mathcal{L}'_{KG} = l]$$

$$= \Pr[\mathsf{pk}' = (\mathbf{A}, \mathbf{t})] \cdot$$

$$\sum_{(\boldsymbol{\rho}_i) \in S} \Pr[\mathcal{U}' = (\boldsymbol{\rho}_i) | \mathsf{pk}' = (\mathbf{A}, \mathbf{t})] \cdot \Pr[\mathcal{L}'_{KG} = l | \mathsf{pk}' = (\mathbf{A}, \mathbf{t}), \mathcal{U}' = (\boldsymbol{\rho}_i)]$$

By application of the NIuo property for the following first equality and by similarities between the two hybrids for the latter, we have the following:

$$\Pr[\mathcal{L}_{KG} = l | \mathsf{pk} = (\mathbf{A}, \mathbf{t}), \mathcal{U} = (\boldsymbol{\rho}_i)] = \Pr[\mathcal{L}'_{KG} = l | \mathsf{pk}' = (\mathbf{A}, \mathbf{t}), \mathcal{U}' = (\boldsymbol{\rho}_i)]$$

$$\Pr[\mathsf{pk} = (\mathbf{A}, \mathbf{t})] = \Pr[\mathsf{pk}' = (\mathbf{A}, \mathbf{t})]$$

$$\Pr[\mathcal{U} = (\boldsymbol{\rho}_i) | \mathsf{pk} = (\mathbf{A}, \mathbf{t})] = \Pr[\mathcal{U}' = (\boldsymbol{\rho}_i) | \mathsf{pk}' = (\mathbf{A}, \mathbf{t})]$$

Thus, $\Pr[\mathsf{pk} = (\mathbf{A}, \mathbf{t}), \mathcal{L}_{KG} = l] = \Pr[\mathsf{pk}' = (\mathbf{A}, \mathbf{t}), \mathcal{L}'_{KG} = l]$ and the views of $\mathcal{A}$ are the same in both hybrids, which concludes the proof. □

D.2 Our Instantiation of the [LNP22] NIZK

To prove that two ciphertexts are encryptions under different public keys of a same message, we use the lattice-based NIZK proof system in the ROM from [LNP22]. More precisely, we use the interactive protocol described in [LNP22, Sec. 5.2] transformed with Fiat-Shamir to an non-interactive one, as detailed in [LNP22, Sec. 6 & App. B]. Thus, when referring to this protocol we actually refer to its transformed NIZK version. The LNP NIZK is more expressive than what we need, thus we will only be introducing the relations of interest for this paper. For a modulus p and a statement composed of

- For $i \in [v^{(d)}]$, $\mathbf{E}_i^{(d)} \in \mathcal{R}_p^{k_i^{(d)} \times 2k'+1}$, $\mathbf{v}_i^{(d)} \in \mathcal{R}_p^{k_i^{(d)}}$
- For $i \in [v^{(e)}]$, $\mathbf{E}_i^{(e)} \in \mathcal{R}_p^{k_i^{(e)} \times 2k'+1}$, $\mathbf{v}_i^{(e)} \in \mathcal{R}_p^{k_i^{(e)}}$
- Bounds $(\beta_i^{(d)})_{i \in [v^{(d)}]}$, $(\beta_i^{(e)})_{i \in [v^{(e)}]}$
- Matrix $\mathbf{E}_{bin} \in \mathcal{R}_p^{k_{bin} \times 2k'+1}$, $\mathbf{v}_{bin} \in \mathcal{R}_p^{k_{bin}}$

the LNP NIZK proves that there exists a witness vector $\mathbf{r} \in \mathcal{R}_p^{2k'+1}$ satisfying the following relationships with overwhelming probability

$$\forall i \in [v^{(d)}], \left\| \mathbf{E}_i^{(d)}\mathbf{r} - \mathbf{v}_i^{(d)} \right\|_\infty \leq \beta_i^{(d)} \tag{5}$$

$$\forall i \in [v^{(e)}], \left\| \mathbf{E}_i^{(e)}\mathbf{r} - \mathbf{v}_i^{(e)} \right\|_2 \leq \beta_i^{(e)} \tag{6}$$

$$\mathbf{E}_{bin}\mathbf{r} - \mathbf{v}_{bin} \text{ is binary}^4 \tag{7}$$

We now explain how to use the LNP NIZK to prove joint encryption of the scheme Σ described in Section 4, where we use the formulation of encryption from Equation 3. Essentially our idea is to reproduce the instantiations of Equations 5 to 7 in the context of verifiable encryption as described in [LNP22, Sec. 6.3] twice, while forcing the use of a same message for both ciphertexts. Note that for technical reasons the modulus q used in the encryption scheme Σ is not the same as the modulus p of the relationships proven in the NIZK.

Corollary 1 (Joint Encryption NIZK [LNP22, Theorem 5.3]). *Let k and q be the public matrix dimension and modulus of Σ, $k' = 2k+1$ the size of an encryption randomness for Σ in Eq. 3 notations and $B^{(e)}$ the norm bound on* RandSpace *elements. Let $\psi > 0$ be the approximate range gap and $B^{(d)} > 0$ as both defined in [LNP22, Theorem 5.3], and Π the non-interactive protocol from [LNP22, Theorem 5.3], with p its ring modulus and the constraint $q(B^{(e)}\sqrt{nk'}/2 + 1 + B^{(d)}\psi) < p$. Then Π is a NIZK in the ROM for the relation R_Σ, when for $(\mathsf{x} = (\mathbf{c}_1, \mathbf{c}_2, \mathbf{B}_1, \mathbf{B}_2), \mathsf{w} = (\mu, \mathbf{r}_1, \mathbf{r}_2)) \in \mathsf{R}_\Sigma$ we input Π with witness $(\mathbf{r}_1, \mathbf{r}_2, \mu)^t$ and the following statement[5]:*

$$v^{(d)} = v^{(e)} = 2$$

$$\mathbf{E}_1^{(d)} = \frac{1}{q}\begin{pmatrix} \mathbf{B}_1 & \mathbf{0}_{k+1,k'} & \begin{matrix} \mathbf{0}_{k,1} \\ \lfloor \frac{q}{2} \rceil \end{matrix} \end{pmatrix}, \quad \mathbf{v}_1^{(d)} = \frac{1}{q}\mathbf{c}_1, \quad \beta_1^{(d)} = B^{(d)}$$

$$\mathbf{E}_2^{(d)} = \frac{1}{q}\begin{pmatrix} \mathbf{0}_{k+1,k'} & \mathbf{B}_2 & \begin{matrix} \mathbf{0}_{k,1} \\ \lfloor \frac{q}{2} \rceil \end{matrix} \end{pmatrix}, \quad \mathbf{v}_2^{(d)} = \frac{1}{q}\mathbf{c}_2, \quad \beta_2^{(d)} = B^{(d)}$$

$$\mathbf{E}_1^{(e)} = \begin{pmatrix} \mathbf{I}_{k'} & \mathbf{0}_{k',k'} & 0 \end{pmatrix}, \quad \mathbf{v}_1^{(e)} = \mathbf{0}_{k'}, \quad \beta_1^{(e)} = B^{(e)}$$

$$\mathbf{E}_2^{(e)} = \begin{pmatrix} \mathbf{0}_{k',k'} & \mathbf{I}_{k'} & 0 \end{pmatrix}, \quad \mathbf{v}_2^{(e)} = \mathbf{0}_{k'}, \quad \beta_2^{(e)} = B^{(e)}$$

$$\mathbf{E}_{bin} = \begin{pmatrix} \mathbf{0}_{1,k'} & \mathbf{0}_{1,k'} & 1 \end{pmatrix}, \quad \mathbf{v}_{bin} = 0.$$

Equations 6 are instantiated so to check that $\mathbf{r}_1$ and $\mathbf{r}_2$ are in RandSpace, while equation 7 is so to check that μ is binary, and thus a proper message. Equations 5 instantiations are a tad more technical, but essentially these aim to ensure the encryption relations hold modulo q, while the NIZK only ensures that the relations above hold modulo p. We defer readers to [LNP22, Sec. 6.3] for full explanations.

[4] Binary means all the vector's polynomials' coefficients are either 0 or 1.

[5] Since q is prime, its inverse is well-defined modulo p as long as q does not divide p.

One-Time Simulation Soundness We now argue that [LNP22] fulfills one-time simulation soundness, following the framework of [Lip25]. To fix notation, we denote a proof transcript coming from a Fiat-Shamir $(2\mu+1)$-move argument system by $\pi = (p_1, v_1, \ldots, p_\mu, v_\mu, p_{\mu+1})$. Here, the p_i's correspond to prover messages and the v_i's to challenge messages in the interactive argument system.

An important characteristic of a NIZK in the framework of Lipmaa [Lip25] is at which rounds the (canonical) simulator must explicitly program the random oracle. More precisely, for which rounds does the adversary first sample v_i, then computes p_i as a function of v_i and later programs H_i to be consistent. We focus on simulators where programming happens only at the last round.

Definition 17 (Last Round Canonical Simulator). *Consider a* NIZK $\Pi^{\mathsf{H}}_{\mathsf{FS}} = (\mathsf{P}^{\mathsf{H}}_{\mathsf{FS}}, \mathsf{V}^{\mathsf{H}}_{\mathsf{FS}})$*, resulting from a Fiat-Shamir transformed* $(2\mu+1)$*-move argument system, with access to the random oracle* H *and with simulator* $(\mathsf{Sim}_{\mathsf{H}}, \mathsf{Sim}_{\pi})$*. We say that the simulator is* last round canonical *if*

- *it computes* $(p_1, v_1, \ldots, p_{\mu-1}, v_{\mu-1})$ *without programming the random oracle,*
- *it samples* v_μ *uniformly at random from the challenge space,*
- *it computes* p_μ *and* $p_{\mu+1}$ *as a function of* v_μ*,*
- *it programs the random oracle* H *so that* $v_\mu = \mathsf{H}(\mathsf{pp}, \mathsf{x}, (p_j)_{j\in[\mu]})$*.*

We introduce the property of last round weak unique responses.

$\mathsf{Expt}^{\mathsf{LUR}}_{\mathcal{A}}(1^\kappa)$

1 : $\mathsf{x} \leftarrow \mathcal{A}^{\mathsf{Sim}_{\mathsf{H}}}(1^\kappa)$
2 : $\pi \leftarrow \mathsf{Sim}_\pi(x)$
3 : $\pi' \leftarrow \mathcal{A}^{\mathsf{Sim}_{\mathsf{H}}}(\pi)$
4 : **parse** $\pi = (p_1, v_1, \ldots, p_\mu, v_\mu, p_{\mu+1})$
5 : **parse** $\pi' = (p'_1, v'_1, \ldots, p'_\mu, v'_\mu, p'_{\mu+1})$
6 : $\pi_| := (p_1, v_1, \ldots, p_\mu, v_\mu)$
7 : $\pi'_| := (p'_1, v'_1, \ldots, p'_\mu, v'_\mu)$
8 : $b = \mathsf{Vf}(\mathsf{x}, \pi') \wedge (\pi_| = \pi'_|) \wedge (p_{\mu+1} \neq p_{\mu+1})$
9 : **return** b

Fig. 9: Last Round Weak Unique Response

Definition 18 (Last Round Weak Unique Response). *Consider a* $(2\mu+1)$*-move public-coin interactive argument system* $\Pi = (\mathsf{P}, \mathsf{Vf})$ *and its Fiat-Shamir transformed argument system* $\Pi_{\mathsf{FS}} = (\mathsf{P}^{\mathsf{H}}_{\mathsf{FS}}, \mathsf{Vf}^{\mathsf{H}}_{\mathsf{FS}})$*, with access to the random oracle* H *and with simulator* $(\mathsf{Sim}_{\mathsf{H}}, \mathsf{Sim}_{\pi})$*. The* Π_{FS} *has computational last round*

weak unique responses (LUR*) in the random oracle if for any* $\kappa \in \mathbb{N}$ *and any PPT adversary* $\mathcal{A}$ *it holds*

$$\mathsf{Adv}^{\mathsf{LUR}}_{\mathcal{A}}(1^\kappa) := \Pr\left[\mathsf{Expt}^{\mathsf{LUR}}_{\mathcal{A}}(1^\kappa)\right] \leq \epsilon_{\mathsf{LUR}}(\kappa)$$

where the probability is taken over the random coins of the adversary and the experiment defined in Figure 9.

We are now ready to state the composition theorem of Lipmaa [Lip25, Thm. 1]. Informally, it proves that a Fiat-Shamir transformed interactive public-coin argument system, which is zero-konwledge, knowledge sound, whose simulator only explicitly programs the random oracle at the last round and has last round unique response is simulation extractable.

Theorem 3 ([Lip25, Thm. 1]). *Let* Π_{FS} *be a Fiat-Shamir transformed non-interactive argument system with a last round canonical simulator* Sim*. If* Π_{FS} *is zero-knowledge, knowledge sound and has last round weak unique responses, then it is simulation extractable with respect to* Sim*. More concretely, for any* $\kappa \in \mathbb{N}$*, PPT adversary* $\mathcal{A}$ *and knowledge extractor* $\mathcal{E}_{\mathsf{ks}}$*, there exists PPT algorithms* $\mathcal{B}$*, simulation extractor* $\mathcal{E}_{\mathsf{se}}$ *and* $\mathcal{C}$ *such that*

$$\mathsf{Adv}^{\mathsf{SE}}_{\mathcal{E}_{\mathsf{se}},\mathcal{A},\mathsf{Sim}}(1^\kappa) \leq \mathsf{Adv}^{\mathsf{KS}}_{\mathcal{E}_{\mathsf{ks}},\mathcal{B}}(1^\kappa) + q \cdot \mathsf{Adv}^{\mathsf{LUR}}_{\mathcal{C}}(1^\kappa)$$

where q *is the number of* $\mathcal{A}$*'s queries to* Sim*.*

The original result is stated for a more general notion of weak unique response, but as the NIZK by [LNP22] only needs the unique response property for the last round, we adapted the statement for simplicity.

We start by observing that the simulator described in [LNP22] is last round canonical (cf. Def. 17), as it only programs the random oracle at the very last round (when invoking the sub-protocol $\Pi^{(2)}$ described in [LNP22, Fig. 6]). More precisely, the simulator will first sample a random challenge c, as well as responses $\mathbf{z}_1, \mathbf{z}_2$ following a Gaussian distribution. The simulator then computes a matching previous prover message $\mathbf{w} = \mathbf{A}_1\mathbf{z}_1 + \mathbf{A}_2\mathbf{z}_2 - c\mathbf{t}_A$, where $\mathbf{A}_1, \mathbf{A}_2$ and $\mathbf{t}_A$ are part of the public parameters and statement. Note that both $\mathbf{A}_1 \in \mathcal{R}_q^{n \times m_1}$ and $\mathbf{A}_2 \in \mathcal{R}_q^{n \times m_2}$ are uniform random matrices.

We are now ready to prove that the [LNP22] argument system has last round weak unique response.

Lemma 15. *Assuming the hardness of* M-SIS*, the [LNP22] argument system has last round weak unique responses.*

Proof. The interactive [LNP22] argument system is 9-move, hence $\mu = 4$ and a Fiat-Shamir transformed transcript looks like $\pi = (p_1, v_1, \ldots, p_4, v_4, p_5)$, where the p_i's are the prover's messages and the v_i's are the challenges. Without fully describing the transcript, we observe that $\mathbf{w}$ mentioned above is part of p_4, that v_4 equals the challenge c and the last prover message p_5 corresponds to the response $\mathbf{z}_1, \mathbf{z}_2$. Verification checks (among other things) if $\mathbf{A}_1\mathbf{z}_1 + \mathbf{A}_2\mathbf{z}_2 =$

$\mathbf{w} + c \cdot \mathbf{t}_A$ and if $\|\mathbf{z}_1\| \leq B_1$ and $\|\mathbf{z}_2\| \leq B_2$, where we defer for the concrete formula of B_1 and B_2 to [LNP22]. Now, assume there exists a PPT adversary $\mathcal{A}$ who wins the LUR game from Figure 9. That is, on a statement x defined by the adversary $\mathcal{A}$ and a simulated proof π, the adversary provides a second proof π'. Both proofs π and π' are using the same $\mathbf{A}_1, \mathbf{A}_2$, t_A, $\mathbf{w}$ and challenge c. However, they contain two distinct pairs $(\mathbf{z}_1, \mathbf{z}_2) \neq (\mathbf{z}'_1, \mathbf{z}'_2)$. By the winning condition of the LUR experiment, π' has to verify, hence $\mathbf{w}+c\cdot\mathbf{t}_A = \mathbf{A}_1\mathbf{z}'_1+\mathbf{A}_2\mathbf{z}'_2$. Moreover, by the construction of the simulator, it holds $\mathbf{A}_1\mathbf{z}_1 + \mathbf{A}_2\mathbf{z}_2 = \mathbf{w} + c$. Thus $\begin{bmatrix}\mathbf{A}_1 & \mathbf{A}_2\end{bmatrix} \cdot \begin{bmatrix}\mathbf{z}_1 - \mathbf{z}'_1 \\ \mathbf{z}_2 - \mathbf{z}'_2\end{bmatrix} = 0$, where the norm of $\mathbf{z} = \begin{bmatrix}\mathbf{z}_1 - \mathbf{z}'_1 \\ \mathbf{z}_2 - \mathbf{z}'_2\end{bmatrix}$ is bounded by $2(B_1 + B_2)$ except with negligible probability. Note that for the norm bound of the simulated proof's $\mathbf{z}_1, \mathbf{z}_2$ we apply a Gaussian tail bound. This solves M-SIS for matrices of dimension $n \times (m_1 + m_2)$ over $\mathcal{R}_q$ with norm bound $B = 2(B_1 + B_2)$. □

As simulation extractability implies one-time simulation soundness, we state the following corollary.

Corollary 2. *The [LNP22] argument system is one-time simulation sound.*

For the proof, instantiate Theorem 3 with [LNP22], using that its simulator programs the random oracle only in the last round, that it is proven to be zero-knowledge, knowledge sound and has last round weak unique responses (as proven in Lemma 15).

D.3 Topaz Algorithms

For completeness, we provide in this section the full description of Topaz, as discussed in Section 6.

Let $\mathcal{M}$ be Rutile's message space and RandDistr its randomness distribution. Let (P, V) be the one-time simulation-sound NIZK described in Section 6.2. Concretely, we set $\mathcal{M} = \{0,1\}^n$ and $\mathsf{RandDistr} = \mathsf{SUnif}(\mathcal{R}, 2k+1, u, d\cdot\mathrm{rep})$. Then, let $B^{(e)} = 2^u\sqrt{3n\,(2k+1)\,d\,\mathrm{rep}}$ and define $\mathsf{RandSpace} = \left\{\mathbf{r} \in \mathcal{R}^{2k+1} \colon \|\mathbf{r}\|_2 \leq B^{(e)}\right\}$.

Algorithm 9 $\mathsf{KeyGen}_{\mathsf{Topaz}}$

Input: level of security κ
Output: public key $\mathsf{pk} := (\mathbf{A}_1, \mathbf{t}_1, \mathbf{A}_2, \mathbf{t}_2)$ | private key $\mathsf{sk} = [\![\mathbf{s}_1]\!]$

1: $\mathbf{A}_1, \mathbf{A}_2 \leftarrow\!\$\ \mathcal{R}_q^{k\times k}$
2: $[\![\mathbf{s}_1]\!], [\![\mathbf{s}_2]\!] \leftarrow \mathsf{AddRepNoise}(\mathbf{0}_{d,1}, u, \mathrm{rep})$ ▷ NIuo gadget for SUnif
3: $[\![\mathbf{e}_1]\!], [\![\mathbf{e}_2]\!] \leftarrow \mathsf{AddRepNoise}(\mathbf{0}_{d,1}, u, \mathrm{rep})$ ▷ NIuo gadget for SUnif
4: $[\![\mathbf{t}_1]\!] \leftarrow \mathbf{A}_1 \cdot [\![\mathbf{s}_1]\!] + [\![\mathbf{e}_1]\!]$
5: $[\![\mathbf{t}_2]\!] \leftarrow \mathbf{A}_2 \cdot [\![\mathbf{s}_2]\!] + [\![\mathbf{e}_2]\!]$
6: $\mathbf{t}_1 := \mathsf{Decode}([\![\mathbf{t}_1]\!])$ ▷ NIo gadget for $\mathbf{t}_1 := \sum_i [\![\mathbf{t}_1]\!]_i$
7: $\mathbf{t}_2 := \mathsf{Decode}([\![\mathbf{t}_2]\!])$ ▷ NIo gadget for $\mathbf{t}_2 := \sum_i [\![\mathbf{t}_2]\!]_i$
8: $\mathsf{pk} := (\mathbf{A}_1, \mathbf{t}_1, \mathbf{A}_2, \mathbf{t}_2)$
9: $\mathsf{sk} := [\![\mathbf{s}_1]\!]$ ▷ $[\![\mathbf{s}_2]\!]$ is discarded

Algorithm 10 $\mathsf{Enc}_{\mathsf{Topaz}}$

Input: message $m \in \mathcal{M}$ | the public key $\mathsf{pk} = (\mathsf{pk}_1, \mathsf{pk}_2) = (\mathbf{A}_1, \mathbf{t}_1, \mathbf{A}_2, \mathbf{t}_2)$
Output: ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$

$\mathsf{flag} = \mathsf{false}$
while $\mathsf{flag} = \mathsf{false}$ **do**
  $R_1 := (\mathbf{u}_1, \mathbf{e}_1, f_1) \leftarrow \mathsf{RandDistr}(1^\kappa)$
  $R_2 := (\mathbf{u}_2, \mathbf{e}_2, f_2) \leftarrow \mathsf{RandDistr}(1^\kappa)$
  if $R_1, R_2 \in \mathsf{RandSpace}$ **then**
    $\mathsf{flag} = \mathsf{true}$
  end if
end while
$\mathbf{v}_1^t = \mathbf{u}_1^t \cdot \mathbf{A}_1 + \mathbf{e}_1^t$
$v_1 = \mathbf{u}_1^t \cdot \mathbf{t}_1 + f_1 + \lfloor q/2 \rceil m$
$\mathsf{ct}_1 := (\mathbf{v}_1^t, v_1)$
$\mathbf{v}_2^t = \mathbf{u}_2^t \cdot \mathbf{A}_2 + \mathbf{e}_2^t$
$v_2 = \mathbf{u}_2^t \cdot \mathbf{t}_2 + f_2 + \lfloor q/2 \rceil m$
$\mathsf{ct}_2 := (\mathbf{v}_2^t, v_2)$
$\pi \leftarrow \mathsf{P}(1^\kappa, (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{pk}_1, \mathsf{pk}_2), (m, R_1, R_2))$
$\mathsf{ct} := (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$

Algorithm 11 $\mathsf{Dec}_{\mathsf{Topaz}}$

Input: ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$ | private key $\mathsf{sk} = [\![\mathbf{s}_1]\!]$ | pub key $\mathsf{pk} := (\mathsf{pk}_1, \mathsf{pk}_2)$
Output: message $m \in \mathcal{M} \cup \{\perp\}$

if $\mathsf{V}(\pi, (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{pk}_1, \mathsf{pk}_2))$ **then**
  parse $\mathsf{ct}_1 := (\mathbf{v}_1^t, v_1)$
  $[\![m']\!] \leftarrow v_1 - \mathbf{v}_1^t \cdot [\![\mathbf{s}_1]\!]$
  $[\![m]\!] \leftarrow \mathsf{1bit\text{-}Compress}([\![m']\!])$ ▷ NI gadget for a d-sharing of $\left\lfloor \|m'\|_\infty / \lfloor q/2 \rceil \right\rceil$
  $m := \mathsf{Decode}([\![m]\!])$ ▷ NIo gadget for $m := \sum_i [\![m]\!]_i$
else
  $m := \perp$
end if