Bittersweet Signatures: Bringing LWR to a Picnic for Hardware-Friendly MPC-in-the-Head – Full Version –

Brieuc Balon[1], Gianluca Brian[2,3], Sebastian Faust[2], Carmit Hazay[4], Elena Micheli[2], and François-Xavier Standaert[1]

[1] UCLouvain, ICTEAM, Crypto Group, Louvain-la-Neuve, Belgium
[2] Technical University of Darmstadt, Germany
[3] ETH Zurich, Zurich, Switzerland
[4] Bar-Ilan University, Israel

Abstract. Post-quantum signature schemes are becoming increasingly important due to the threat of quantum computers to classical cryptographic schemes. Among the approaches considered in the literature, the MPC-in-the-head paradigm introduced by Ishai et al. (STOC'07) provides an innovative solution for constructing zero-knowledge proofs by exploiting Multi-Party Computation (MPC). This technique has proven to be a versatile tool in order to design efficient cryptographic schemes, including post-quantum signatures. Building on the MPC-in-the-head paradigm, we introduce Bittersweet signatures, a new class of signature schemes based on the Learning With Rounding (LWR) assumption. Their main advantage is conceptual simplicity: by exploiting (almost) key-homomorphic pseudorandom functions (PRFs), a cryptographic object that preserves pseudorandomness while allowing linear operations on keys, we obtain a very regular design offering nice opportunities for parallel implementations. Theoretically, analyzing Bittersweet signatures requires addressing significant challenges related to the (carry) leakage that almost key-homomorphic operations lead to. Concretely, Bittersweet signatures natively lead to competitive signature sizes, trading moderate software performance overheads for hardware performance gains when compared to state-of-the-art MPC-in-the-head schemes (e.g., relying on code-based assumptions), while admittedly lagging a bit behind recent algorithms based on the VOLE-in-the-head or Threshold-Computation-in-the-head frameworks. Besides, their scalability and algebraic structure makes them promising candidates for leakage-resilient implementations. The new abstractions we introduce additionally suggest interesting research directions towards further optimization and generalization.

1 Introduction

1.1 State of the Art

For more than a decade now, designing signature schemes that can withstand quantum computers has been a major research area. These efforts culminated with the selection of three first algorithms for standardization by the NIST

in July 2022: Dilithium (https://pq-crystals.org/dilithium/) and Falcon (https://falcon-sign.info/), whose security relies on the hardness of solving lattice problems, and SPHINCS$^+$, whose security relies on cryptographic hash functions (https://sphincs.org/). Nevertheless, the design of post-quantum signature schemes remains of fundamental importance for several reasons. First, increasing the diversity of the hardness assumptions these algorithms rely on is needed to avoid "putting all eggs in the same basket". This has prompted the NIST to open a call for new proposals, leading to the submission of 50 candidates in June 2023.[1] Second, the performances and cost of these algorithms remains a topic of concern, especially if security against implementation attacks is required [65, 11, 38, 66]. Finally, the state of the art regarding signatures with advanced functionalities such as threshold and multisignatures [29], group signatures [37] and ring signatures [77], currently remains relatively sparse.

As reflected by the submissions to the 2023 NIST call, post-quantum signatures can be based on various assumptions (e.g., besides lattices and hash functions used by Dilithium, Falcon, and SPHINCS$^+$, Isogenies, code-based assumptions, and multivariate polynomials have been considered) and rely on different methodologies. Among those, one popular approach, on which we focus in this work, is to rely on zero-knowledge proofs using the MPC-in-the-head paradigm, introduced by Ishai *et al.* [62]. This paradigm refers to a technique that uses Multi-Party Computation (MPC) in the context of Zero-Knowledge (ZK) proofs, where a set of parties compute a function over their inputs while keeping those inputs private. In the proof system, the prover then simulates an MPC protocol in his mind (or "in the head") and commits to the produced views. The verifier then asks to open a subset of these views and verifies that they are consistent and that the output of the computation is correct.

While initially seen as a primarily theoretical contribution, this paradigm has been shown to be a versatile tool to construct (among others) efficient signature schemes, starting with the work of Giacomelli *et al.* [58]. On the one hand, it has motivated several designs relying on symmetric cryptographic primitives like Picnic [36, 35], BBQ [40], Banquet [18], Limbo [41] and Rainier [46]. On the other hand, it has also been exploited to leverage various other assumptions. For example, LegRoast relies on the one-wayness of the Legendre PRF [26], several schemes rely on the syndrome decoding problem [60, 52, 28, 34], Feneuil *et al.* proposed schemes based on the subset sum problem [53], Adj *et al.* proposed a scheme based on the MinRank problem [3], Benadjila *et al.* proposed a scheme based on the multivariate quadratic problem [21], and Bettaieb *et al.* proposed one based on the permuted kernel problem [24]. To a large extent, these new designs followed the first motivations mentioned above (*i.e.,* avoiding putting all eggs in the same basket and improving performances, mostly ignoring implementation attacks so far). As a result, and to the best of our knowledge, there are few works considering the combination of lattice-based assumptions and the MPC-in-the-head paradigm (exceptions are discussed in related works).

[1] https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures.

1.2 Our Contribution

The early MPC-in-the-head schemes relied on various instantiations of the underlying MPC protocol, with dishonest majority [58, 36, 67] and honest majority [8, 27], introducing different trade-offs between prover runtime and proof length. More recent schemes considered new abstractions that capture simpler cryptographic building blocks, such as game-based cryptographic primitives that support homomorphic computations [61] and puncturable PRFs [34]. In this paper, we consider yet another instantiation of the MPC-in-the-head paradigm based on (almost) key-homomorphic PRFs [31], which preserve pseudorandomness while allowing operations on the keys in an (almost) homomorphic way.

Key-homomorphic PRFs are useful in the MPC-in-the-head (signature) setting because they enable non-interactive computations, which considerably simplifies the views on which a prover has to commit. The concrete PRF we use for this purpose is based on the Learning With Rounding (LWR) assumption [13], and therefore goes against the conventional wisdom that signatures based on the MPC-in-the-head paradigm are best combined with non-lattice-based assumptions. We nevertheless show that this choice comes with various advantages, which we reflect with the Bittersweet signatures' name.[2] Among others, working with this object enables a clean formalization of the virtual MPC protocol for evaluating the PRF on a public value (as needed for our signature scheme). It also leads to a regular design facilitating parallel implementations where the (multi-party) computations can be performed on each share independently.

As a first step towards formalizing the use of key-homomorphic primitives for MPC-in-the-head signatures, we investigate a simple setting where all-but-one share are revealed at every iteration of the protocol. The main technical challenge we need to solve for this purpose is to deal with the additional (carry) leakage that the almost key-homomorphism leads to. That is, since the share-based MPC computations of the PRF we use are only approximately correct, the errors may leak additional information about the secret (via carry propagation). Like many other post-quantum signature schemes, we rely on the Fiat-Shamir with abort transform in order to mitigate this carry leakage [69]. We also discuss how heuristic instantiation choices can make this carry leakage arbitrarily small and how to capture it directly with (new) hardness assumptions. More precisely, we show in Section A that a "LWR with carry leakage" can be reduced to standard LWR (given some increase of the parameters), and that it is also connected to a form of "LWR with misuse", which may be of independent interest.

A more detailed overview of our technical contribution is given in Section 2.

We finally illustrate the potential of our results by discussing the concrete instantiation of Bittersweet signatures. This allows us to exhibit the gaps between parameters suggested by (current) security proofs and parameters enabling efficient implementations, as observed for other schemes like Picnic [36, 35]. It also allows us to exhibit how more aggressive optimizations can gradually reduce the size of Bittersweet signatures and lead to competitive instances. We

[2] Suggesting an initially strange but ultimately good-tasting mix of ingredients.

additionally confirm the interest of regular designs for hardware implementation purposes. We show that almost key-homomorphism leads to a straightforward way to increase the number of shares since the same circuit can be applied to all the shares without interaction, which is easily leveraged by hardware implementations for computationally-intensive operations (e.g., hash functions, multiplications). As a result, we can demonstrate hardware performance gains for moderate software performance overheads when compared to state-of-the-art MPC-in-the-head schemes (e.g., relying on code-based assumptions), while the performances of Bittersweet are admittedly lagging a bit behind the ones recent schemes based on the VOLE-in-the-head or Threshold-Computation-in-the-head frameworks. Incidentally, our results also lead to one of the few efficient signature schemes relying on unstructured lattice assumptions in the literature.

1.3 Future Work

The conceptual simplicity of Bittersweet signatures and the generic investigation of almost key-homomorphic primitives for MPC-in-the-head signatures that we initiate come with a nice potential for further developments towards leakage-resilient implementations, and opens interesting research avenues:

Leakage-resilience. It has already been shown that the PRF on which Bittersweet signatures rely can be efficiently masked [48]. Hence, implementations with strong leakage-resilience are a natural target for future work. The (almost linear) scaling trends of masked key-homomorphic primitives should, for example, lead to a better efficiency vs. (side-channel) security tradeoff than MPC-in-the-head signatures based on symmetric cryptographic primitives [78, 10]. The large moduli used by Bittersweet signatures could additionally be turned intro an asset if combined with advanced (e.g., inner product) masking schemes [50].

Generalizations. Our investigations focus on a PRF operating in $\mathbb{Z}_{2^q}$, additive secret sharing and a protocol opening all-but-one share per iteration. But nothing theoretically prevents generalizations to other fields, secret sharing schemes, and opening strategies. Such generalizations could be interesting for performance [54, 55], leakage-resilience [72] and possibly help to design new signatures with advanced functionalities (*e.g.,* threshold or multisignatures). Whether our results can be extended to the LWE context and whether other key-homomorphic PRFs can lead to better results are also interesting open questions.

Optimizations. As many post-quantum signature schemes based on the MPC-in-the-head paradigm, Bittersweet signatures require to commit on a lot of randomness. The parallel nature of the Bittersweet design nevertheless raises the question of whether one could get rid of (some of) these commitments by leveraging properties of LWR (*e.g.,* if the LWR function was itself sufficiently committing). Such optimizations would lead to further performance improvements.

LWR with carry leakage and misuse. The formal investigation of these new variants of standard hardness assumptions that we initiate in Section A finally raises the question whether they could find applications in other contexts.

1.4 Related Works

Combining the MPC-in-the-head paradigm with lattice-based assumption was already proposed by other works like [22, 19, 25], yet in a quite different context than ours (*i.e.,* not as a building block of key-homomorphic PRFs).

Using a rounding operation as a source of non-linearity to build low-complexity cryptographic objects is reminiscent of using alternating moduli [30, 44].

A recent line of work focuses on VOLE-in-the-head signatures [17, 76, 39, 16], whose key building blocks are multi-round interactive protocols based on Vector Oblivious Linear Evaluation (VOLE) correlations. This paradigm is tailored for ZK protocols in the VOLE-hybrid model, aiming to compile them into a publicly verifiable protocol. However, it may require more than three rounds of interaction and non-black-box compilation techniques within the underlying VOLE protocol. As a result, adapting their techniques to our context would not be straightforward, and yields another interesting direction for future work.

MPC-based signatures rely on symmetric primitives (GGM/Merkle trees) in order to derive randomness for the parties. As the number of emulated parties grows, this part of the designs may lead to significant overheads. To address this, [33] uses a block cipher-based Punctured PRF (PPRF) instead of a hash-based one, which reduces the execution time, while [16] redesigns the GGM-PPRF to further shrink the signature size. We avoid these optimizations which would be less beneficial to Bittersweet signatures, for which the main performance bottleneck stems from algebraic operations. Adopting [16] would further introduce a rejection probability tied to the challenge space, complicating the design.

Finally, and besides the aforementioned thresholding approach [54, 55], the hypercube optimization has been proposed to amplify the soundness of any MPC protocol that uses additive secret sharing [74]. Such an optimization is conceptually applicable to Bittersweet, but its interest is reduced because it combines poorly with another (more significant) optimization we leverage, namely replacing the last (unopened) shares by smaller error vectors in the signature.

2 Technical Overview

A successful paradigm for designing signature schemes is based on identification schemes (which can also be viewed as a generalization of Sigma protocols). Specifically, in the first step, a challenge-response 3-round protocol is constructed for a given hard relation. Then, in the next step, a round collapsing mechanism is applied to the above protocol, using the signed message as the challenge. The MPC-in-the-head framework can be seen as a generalization of the above paradigm. In this context, the hard relation is translated into a distributed functionality, which is then realized with multiple parties running an MPC protocol. As a result, the choice of this MPC protocol will have a crucial impact on the level of concrete efficiency of the signature scheme. At a high level, the design of such a signature scheme involves the following four main steps:

1. *Selecting the hard relation* between public and secret keys;

2. *Choosing the MPC scheme* used to compute the relation;
3. *Constructing the Sigma protocol* for the hard relation;
4. *Transforming the Sigma protocol into a signature scheme.*

A well-known example of an MPC-in-the-head signature scheme that further serves as our starting point is the Picnic scheme [36] that follows the next stepping stones toward constructing their finalized (non-optimized) scheme.

1. *Selecting the hard relation:* Picnic's underlying relation is given by a pseudorandom function (PRF). Namely, the public value is the couple (x, y) of input and output of the PRF, the secret value is the key k of the PRF F, and the pairs $((x, y), k)$ in the relation are those satisfying $y = F_k(x)$. Concretely, this PRF is instantiated with the LowMC block cipher [6].
2. *Choosing the MPC scheme*: The scheme of Giacomelli *et al.*, on which Picnic relies, exploits any linear function decomposition [58]. This is a notion introduced to capture protocol virtualization as done in MPC-in-the-head. For their particular instantiation, they consider a (2,3)-decomposition, which operates by performing secret sharing of the computations carried on the gates defining the arithmetic representation of F. First, the secret key k is additively shared into 3 shares k_1, k_2, k_3. Then, linear operations are conducted on each share locally, while multiplication gates require interaction. The information involved in the computation of each output share y_1, y_2, y_3 is finally stored in the corresponding views $\mathsf{view}_1, \mathsf{view}_2, \mathsf{view}_3$.
3. *Constructing the Sigma protocol:* Picnic is based on the ZKBoo protocol of Giacomelli *et al.* [58], which works as follows. The prover first runs the MPC protocol described above on the key k, resulting in a triple of key shares k_1, k_2, k_3, output shares y_1, y_2, y_3, and views $\mathsf{view}_1, \mathsf{view}_2, \mathsf{view}_3$. Next, it commits to all the key shares and views and sends the commitments and output shares to the verifier. The verifier picks a couple of random indices i_1, i_2 and sends them to the prover. As a response, the prover reveals the key shares and views corresponding to the parties indexed by i_1, i_2. Finally, the verifier can check three properties: (1) the correctness of two output shares based on the opened key shares and views, (2) the validity of two commitments, and (3) that the output shares reconstruct to the PRF output y. This process can then be repeated t times with the same long-term key k and different random coins in order to reduce the soundness error.
4. *Transforming the Sigma protocol into a signature scheme:* In the latest versions of Picnic [35], this is done via the Fiat-Shamir heuristic, both in the classic and quantum setting. Namely, the public key is the input-output PRF pair (x, y), the signing key is the PRF key k, and a signature consists of a Sigma protocol transcript. To make it non-interactive, the challenge is computed by hashing the content of the public key, the first round of every iteration of the Sigma protocol and the message to be signed. The security proof is in the quantum random oracle model. It relies on several features of the Sigma protocol: completeness, unpredictable commitments, special soundness, zero-knowledge and quantum computationally unique responses.

Informally, the size and performances of Picnic signatures significantly depend on the number of multiplication gates within the circuit representation of the PRF F (since it affects the running time of the MPC protocol and the size of the views). This concern motivates our interest in key-homomorphic PRFs as a building block for MPC-in-the-head based signatures [31]. Key-homomorphic PRFs allow the computation on each key share independently, i.e., satisfy $F_k(x) = F_{k_1}(x) + \ldots + F_{k_d}(x)$ for every d-additive sharing $k_1, \ldots, k_d$ of the key k. In other words, they imply a "non-interactive MPC" scenario where each party contributes a single message to the MPC protocol. Consequently, key homomorphism eliminates the need for large views or interaction between the virtual parties, allowing us to increase the number of parties almost for free.

While this intuition holds regardless of the details of the key-homomorphic PRF, its specific choice affects security and efficiency. This is because each signature contains many PRF evaluations, one for each party and each iteration of the Sigma protocol. Therefore, we adopt the key-homomorphic PRF introduced by Boneh *et al.* [31], which is based on the Learning with Rounding (LWR) assumption [13]. Our primary motivation for choosing this scheme is its conceptual simplicity and, looking ahead to future work, its potential for leakage-resilient implementations via masking [48]. In this scheme, the PRF key is a random vector $\boldsymbol{k} \in \mathbb{Z}_Q^n$, the input is a random matrix $\boldsymbol{X} \in \mathbb{Z}_Q^{m \times n}$, and the output is computed as $\boldsymbol{y} = F_{\boldsymbol{k}}(\boldsymbol{X}) = \lfloor \boldsymbol{X}\boldsymbol{k} \rceil_P \in \mathbb{Z}_P^m$, where $\lfloor x \rceil_P = \left\lfloor x \cdot \frac{P}{Q} \right\rceil$.[3]

A bit more precisely, this scheme only satisfies a weaker variant of key-homomorphism, known in the literature as *almost* key homomorphism. For a function F and key k, it ensures that $|F_k(x) - (F_{k_1}(x) + \ldots + F_{k_d}(x))| \leq \delta$ for every additive sharing $k_1, \ldots, k_d$ of the key, where δ is a small error. For our LWR-based PRF, this error scales with $\log(d)$ and intuitively describes the carry information that is preserved when *summing-before-rounding* and lost when *rounding-before-summing*. Introducing a small reconstruction error translates into several technical and conceptual challenges in the analysis and design of the scheme, forming the core of our technical contribution. In the following paragraphs, we outline these challenges and explain how we address them.

A modified verification check and its impact on the security of the hard relation. To ensure completeness, we include a small error tolerance in the verification check of the output shares, therefore modifying check (3) of Picnic (cf. Item 3). In other words, instead of checking the consistency of the output shares as $y_1 + \ldots + y_d = y$, we check that $\boldsymbol{y} - (\boldsymbol{y}_1 + \ldots + \boldsymbol{y}_d) \leq \delta$. Besides, and beyond completeness, we also target special soundness, a form of extractability for Sigma protocols (see Definition 2 for a formal definition). To achieve it, we introduce a tolerance in the hard relation as well, defining it as

[3] Formally, our analysis only requires *weak* PRF and we do not need to hash the input with a random oracle as Boneh *et al.* in order to get a PRF. We nevertheless use a hash function in our concrete instances, for performance reasons.

$\mathcal{R} = \{((\boldsymbol{X}, \boldsymbol{y}), \boldsymbol{k}) : |\boldsymbol{y} - F_{\boldsymbol{k}}(\boldsymbol{X})| \leq \delta\}$. Since the distance is measured component-wise and the vector $\boldsymbol{y} - F_{\boldsymbol{k}}(\boldsymbol{X})$ has m components, this relation is $(2\delta+1)^m$-times easier to break compared to a strict equality check (see Lemma 1 for the security bound). Interestingly, we reduce this factor to 3^m when considering a setting where all-but-one shares are revealed in each protocol iteration, modifying check (3) accordingly. Intuitively, this is because having $d-1$ additive shares out of d can be seen as having 1 additive share out of 2. In practice, in check (3), the verifier therefore aggregates the opened key shares $\boldsymbol{k}_{i_1}, \ldots, \boldsymbol{k}_{i_{d-1}}$ into one, and then verifies almost key-homomorphism with the only unopened output share, *i.e.,* the verification check becomes $|\boldsymbol{y} - (F_{\boldsymbol{k}_{i_1}+\ldots+\boldsymbol{k}_{i_{d-1}}}(\boldsymbol{X}) + \boldsymbol{y}_{i_d})| \leq 1$.

Average-case completeness and zero-knowledge. The standard zero-knowledge security definition quantifies over all statement-witness pairs $((\boldsymbol{X}, \boldsymbol{y}), \boldsymbol{k})$ in the relation, which implies that privacy holds even if the distinguisher knows the witness. However, this creates an issue with our simulation due to having almost key homomorphism: while the carry is deterministic for the distinguisher, it remains probabilistic for the simulator. To address this, we restrict ourselves to pairs generated through an external distribution, denoted as $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$. This notion, referred to as average-case zero-knowledge [23], resolves the simulation issue by ensuring that the distinguisher has no longer access to the witness. An orthogonal problem that arises with completeness further requires the switch to the average case definition. The problem arises because a statement-witness pair is in the relation if the absolute value difference between $\boldsymbol{y}$ and $F_{\boldsymbol{k}}(\boldsymbol{X})$ is less than $\mathbf{1}$. However, honestly-generated proofs for pairs satisfying $|\boldsymbol{y} - F_{\boldsymbol{k}}(\boldsymbol{X})| = \mathbf{1}$ might not be accepted. This is because the verifier compares $\boldsymbol{y}$ with $\tilde{\boldsymbol{y}}$, which it computes locally based on the decommitments and the unopened output shares. Since the reconstruction of $\tilde{\boldsymbol{y}}$ can also differ from $F_{\boldsymbol{k}}(\boldsymbol{X})$ by a factor $\mathbf{1}$, by the triangle inequality the difference between $\boldsymbol{y}$ and $\tilde{\boldsymbol{y}}$ can be 2 for some components, resulting in rejection. Therefore, we restrict ourselves to the set of instances in which $\boldsymbol{y} = F_{\boldsymbol{k}}(\boldsymbol{X})$. Looking ahead, this will not pose a problem for the security of our signature scheme because it is defined over honestly generated keys, and therefore, it suffices to generate the key pairs with $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$. Furthermore, the security of Fiat-Shamir signatures is often proved for Sigma protocols with constraints on the statement-witness distribution (cast as identification schemes), and thus will adapt to our security definition. See Section 5 for the details.

The carry leakage information and the abort condition. The main technical challenge of this work concerns estimating how much information the carry leaks. Intuitively, this is because every iteration j of the Sigma protocol provides an eavesdropper with a noisy version $\boldsymbol{z}$ of $\boldsymbol{X}\boldsymbol{k}$, computable from the opened key shares $\boldsymbol{k}_{i_1}, \ldots, \boldsymbol{k}_{i_{d-1}}$ and the unopened output share $\boldsymbol{y}_{i_d}$ as

$$\boldsymbol{z} = \boldsymbol{X}(\boldsymbol{k}_{i_1} + \ldots + \boldsymbol{k}_{i_{d-1}}) + \frac{Q}{P}\boldsymbol{y}_{i_d} \in \left[\boldsymbol{X}\boldsymbol{k} - \frac{Q}{P} + 1, \boldsymbol{X}\boldsymbol{k}\right].$$

By averaging such samples over enough iterations, an eavesdropper could eventually isolate $\boldsymbol{X}\boldsymbol{k}$ and learn the secret. Although the scheme is already correct

as presented, this type of attack compromises its security. Interestingly, we show that an abort-based approach allows us to control the amount of information leaked.[4] The abort event can be triggered in any iteration by the equality of any component of two vectors (the non-rounded output $\boldsymbol{Xk}$ and the sum of the non-rounded output shares $\boldsymbol{Xk}_{i_1} + \ldots + \boldsymbol{Xk}_{i_{d-1}}$), in a fixed α-sized subset of their bits. This subset corresponds to the most significant α bits cut out by the rounding function, i.e., the ones between $r_S = \log(Q) - \log(P)$ and $r_E = \log(Q) - \log(P) + \alpha$. This condition ensures that the remaining transcripts do not reveal more than an LWR instance with α extra bits, as it intuitively guarantees that information on the (secret) LSBs of $\boldsymbol{y}$ cannot be forwarded to its (public) MSBs due to carry propagation. Since the abort condition excludes a $\approx t \cdot m \cdot 2^{-\alpha}$ fraction of the transcripts, we can increase the parameter α to make the abort probability arbitrarily small. We discuss in Section 7 that combining such an increase with reasonable heuristics can significantly reduce the size of Bittersweet signatures. We also propose a formal approach to the carry leakage in Section A, where we introduce two variants of the LWR problem and show that, under certain conditions, they can be reduced to the standard LWR.

Overall security. From the above arguments, our Sigma protocol satisfies average-case completeness and average-case honest-verifier zero-knowledge. Using similar arguments as for Picnic, we also prove that it achieves unpredictable commitments, special soundness, and quantum computationally unique responses, which are properties required for applying the Fiat-Shamir transform. At this point, the security of the signature follows from the vast literature on the post-quantum security of the Fiat-Shamir with abort transform. For correctness, we rely on the analysis of Devevey *et al.* [43], which, to our knowledge, is the only formal treatment of the subject. Regarding unforgeability, most analyses follow a modular framework: first prove unforgeability against no-message attack and then amplify to standard unforgeability. For the former, we consider the analysis of Don *et al.* [47], which, to the best of our knowledge, remains the state-of-the-art.[5] Finally, we extend unforgeability against no-message attack to the standard notion via the recent result of Barbosa *et al.* [14]. See details in Section 6.

3 Preliminaries

3.1 Notation

Given a set $\mathcal{S}$, $|\mathcal{S}|$ denotes its number of elements. If $N \in \mathbb{N}$, $\mathcal{S}^N$ denotes the N-times cartesian product, and $[N]$ denotes the set $\{1, \ldots, N\}$. Given an element $z \in \mathbb{Z}_Q$ for some positive integer Q, $|z|$ denotes $\min\{z \mod Q, -z \mod Q\}$. Given a vector $\boldsymbol{v} \in \mathbb{Z}_Q^N$, $|\boldsymbol{v}|$ denotes the maximal component of $\boldsymbol{v}$ in absolute

[4] A similar technique is used in the analysis of the NIST candidate Dilithium.

[5] Although their results are not directly defined in the abort framework, the properties involved in their proofs do not seem to worsen with the introduction of aborts. We carefully discuss the adaptation of these results we need in Section C.5.

value, and the notation $0 \in \boldsymbol{v}$ refers to the event where $\boldsymbol{v}$ as at least one zero component. Given two integers $Q > P$ and two rings $\mathcal{S}_Q, \mathcal{S}_P$ such that $|\mathcal{S}_Q| = Q$ and $|\mathcal{S}_P| = P$, the *rounding function* $\lfloor \cdot \rceil_P : \mathcal{S}_Q \to \mathcal{S}_P$ is defined as $\lfloor x \rceil_P = \left\lfloor x \frac{P}{Q} \right\rceil$. When referring to the rounding of a vector, we consider its application component-wise. Given two integers p, i such that $i < p$, and a value $x \in \mathbb{Z}_{2^p}$, we denote with $x[i]$ the i-th bit of x. In particular, $x[0]$ is the least significant bit, and $x[p-1]$ is the most significant bit. Throughout the paper, we always consider $x[i]$ as an element of $\mathbb{Z}$ rather than $\mathbb{Z}_2$ or $\mathbb{Z}_{2^p}$. For two values i, j, we denote by $i..j$ the range of indices from i (included) to j (excluded), *i.e.* $i..j = \{i, i+1, \ldots, j-1\}$. If $i = 0$ (resp., $j = p$), we omit it and simply write $..j$ (resp., $i..$). When we sum two ring elements, we always assume that the sum is the one for which the ring is defined. For example, if $x, y \in \mathbb{Z}_Q$, then we write $x + y$ for $x + y \mod Q$, while if $x, y \in \mathbb{Z}_P$, then $x + y$ denotes $x + y \mod P$.

3.2 The Computational Model

Our computational model assumes classical queries to the signing oracle as outlined in NIST's call for Additional Digital Signature Schemes.[6] However, we evaluate the security of our scheme in the Quantum Random Oracle Model (QROM), which extends the classical Random Oracle Model (ROM) to include adversaries making quantum superposition queries to the oracle.

3.3 Standard Cryptographic Primitives

We defer the definitions of signature schemes, pseudorandom generators, weak pseudo-random functions, and hash functions to Section B.

3.4 Sigma Protocols

Let $\mathcal{X}, \mathcal{W} \subseteq \{0,1\}^*$ be sets, and $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$ be an efficiently decidable relation for the language $\mathcal{L} = \{x : \exists w \text{ s.t. } (x, w) \in \mathcal{R}\}$. A *Sigma protocol* Σ for $\mathcal{R}$ is an interactive protocol between two parties, called the *prover* and the *verifier*. The prover is modeled as a couple of probabilistic polynomial-time algorithms $\Sigma.\mathsf{Prove} = (\Sigma.\mathsf{Prove}_1, \Sigma.\mathsf{Prove}_2)$, which are assumed to share state. The verifier $\Sigma.\mathsf{Verify}$ is a deterministic polynomial-time algorithm that outputs a decision bit. Sigma protocols always consist of three rounds. In the first round, the prover generates a first message $\mathit{firstmes} \in \mathcal{N}_{\mathsf{com}}$ as $\mathit{firstmes} \leftarrow\!\!\$\ \Sigma.\mathsf{Prove}_1(x, w)$. Next, in the second round, the verifier answers with a uniformly random generated challenge $ch \leftarrow\!\!\$\ \mathcal{N}_{\mathsf{ch}}$. In the third round, the prover publishes a response $resp \leftarrow\!\!\$\ \Sigma.\mathsf{Prove}_2(ch)$. Finally, the verifier outputs $\Sigma.\mathsf{Verify}(x, \mathit{firstmes}, ch, resp) \in \{0,1\}$. We call *transcript* of the protocol the triple $(\mathit{firstmes}, ch, resp)$.

We now introduce three key properties of Sigma protocols: *completeness*, *honest-verifier zero-knowledge*, and *s-special soundness*. The first ensures that

[6] Mentioned on page 14 of https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022.pdf

honestly generated proofs for statements in the language are always verified. Honest-verifier zero-knowledge guarantees that the transcript of an execution where the verifier issues a random challenge (and thus behaves honestly) reveals no more information on the witness beyond what could be derived from the statement alone. Finally, *s-special soundness* allows the reconstruction of a witness for a given statement from s transcripts, provided they meet specific criteria. These criteria include validity of the transcript, identical first messages, and distinct challenges. Similarly to [23], we consider the *average-case* variant of completeness and zero-knowledge, meaning that we restrict to statements generated according to precise distributions.[7] This relaxes completeness to only hold for statements generated via the distribution, and therefore, not all the honestly generated proofs for statements in the language are correctly verified. Moreover, this weakens the adversaries considered in the experiment for zero-knowledge: their choices cannot depend on the internal coins used for sampling the statement, but only on the statement itself. We make these properties formal below, where the experiments refer to Fig. 1. We denote with $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$ distributions that output values in $\mathcal{X}\times\mathcal{W}$, and with $\mathcal{D}_{\mathcal{X}}$ distributions that output values in $\mathcal{X}$.

Definition 1 (Completeness). *We say that* $\Sigma = (\Sigma.\mathsf{Prove}, \Sigma.\mathsf{Verify})$ *is* $\epsilon_{\Sigma.\mathsf{Comp}}$*-average-case complete for a distribution* $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$ *if*

$$\mathbb{P}\left[\Sigma.\mathsf{Comp}_{\Sigma} = 0\right] \leq \epsilon_{\Sigma.\mathsf{Comp}}.$$

Definition 2 (Special Soundness). *Let* Σ *be a Sigma protocol for a relation* $\mathcal{R}$*. We say that* Π_{NI} *has* $(s, \epsilon_{\Sigma.\mathsf{SS}})$*-special soundness if there exists an algorithm* E *such that for every quantum-polynomial-time adversaries* A *it holds that*

$$\mathbb{P}\left[\Sigma.\mathsf{SS}(\mathsf{A},\mathsf{E}) = 0\right] \leq \epsilon_{\Sigma.\mathsf{SS}}.$$

Looking forward, we prove the unforgeability of the FS signatures *with abort* under a *computational* variant of HVZK. To this end, we merge the techniques used in [59] and [14]. The former proves unforgeability of FS signatures under a *computational* variant of HVZK named multi-HVZK, albeit only in the abort-free scenario. In the second work, unforgeability is proved in the setting *with aborts*, though under a *statistical* variant of HVZK named accepting HVZK.

Therefore, our proof requires a new variant of HVZK. As stated before, it is an *average-case* notion; this aligns with the definitions given in [59] and [14], which are formulated directly for identification schemes. To accommodate signatures with aborts, we adopt the *accepting* property from [14], requiring simulatability only for accepting transcripts. Additionally, we incorporate the *multi*-HVZK variant from [59], where HVZK holds across multiple transcripts.[8]

[7] This is not a problem for the security of the resulting Fiat-Shamir signature. In the literature, the latter is usually built on top of identification schemes, which correspond to Sigma protocols equipped with a statement-witness generation algorithm.

[8] As noted in [59], this stronger variant follows from statistical HVZK, but not from computational HVZK in the context of identification or average-case schemes.

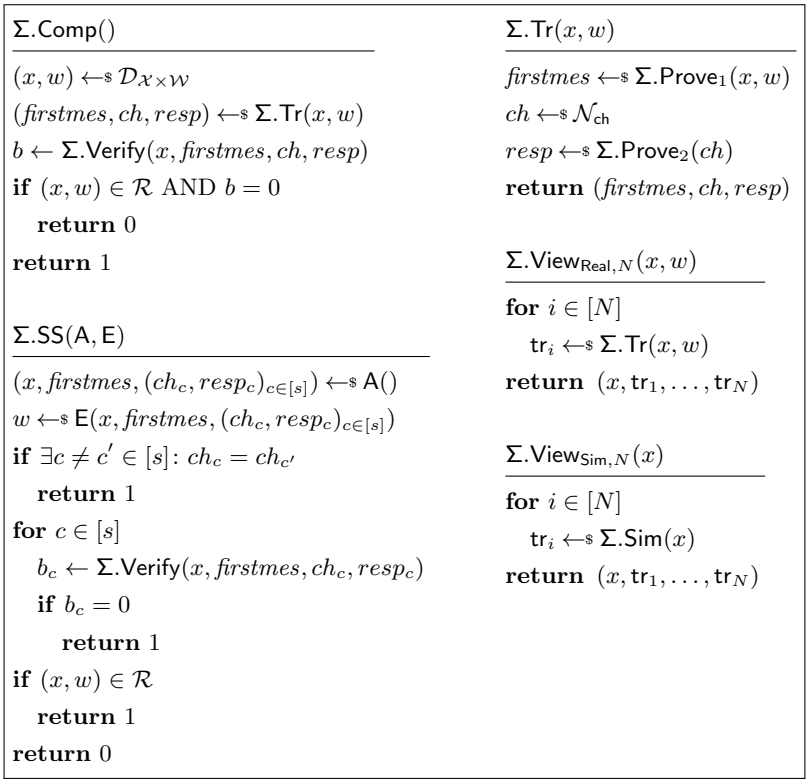

Fig. 1. The experiments relevant to the definitions in Section 3.4

Definition 3 (Average-Case Accepting Multi-HVZK). *Let $N \in \mathbb{N}$ and $\epsilon_{\Sigma.\mathsf{HVZK}} \in [0, 1]$. Let $\mathcal{X}, \mathcal{Y}$ be two sets, $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{Y}$ be a relation, and $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$ be a distribution over $\mathcal{R}$. Let $\mathcal{D}_{\mathcal{X}}$ be the distribution that samples $(x, w) \leftarrow\!\!\$\ \mathcal{D}_{\mathcal{X}\times\mathcal{W}}$ and outputs x. A Sigma protocol $\Sigma = (\Sigma.\mathsf{Prove}, \Sigma.\mathsf{Verify})$ for the relation $\mathcal{R}$ and the distribution $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$ is* average-case accepting $(N, \epsilon_{\Sigma.\mathsf{HVZK}})$-multi-HVZK *($(N, \epsilon_{\Sigma.\mathsf{HVZK}})$-acamHVZK for short) if there exists a polynomial-time simulator $\Sigma.\mathsf{Sim}$ such that, for every stateful quantum polynomial-time adversary A,*

$$\left|\mathbb{P}\left[\mathsf{A}\left(\Sigma.\mathsf{View}_{\mathsf{Real},N}(\mathcal{D}_{\mathcal{X}\times\mathcal{W}})\right) = 1 \mid \mathbf{Success}\right] - \mathbb{P}\left[\mathsf{A}\left(\Sigma.\mathsf{View}_{\mathsf{Sim},N}(\mathcal{D}_{\mathcal{X}})\right) = 1\right]\right|$$

is at most $\epsilon_{\Sigma.\mathsf{HVZK}}$, where **Success** *is the event where $resp_i \neq \perp$ for every $i \in [N]$.*

Next, we present less standard properties for Sigma protocols: *commitment unpredictability* and *quantum computationally unique responses* (QCUR) [47]. The first concerns the probability of guessing the Sigma protocol's first message. The second translates the classical notion of computationally unique responses to the quantum world, stating that it is hard to detect a measurement over a superposition of successful responses to the same first message and challenge.

Definition 4 (Commitment Unpredictability). *Let Σ be a Sigma protocol with statement space $\mathcal{X}$ and witness space $\mathcal{W}$, and let $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$ be a distribution*

that outputs values in $\mathcal{X} \times \mathcal{W}$. We say that Σ has $\epsilon_{\mathsf{ComGuess}}$-unpredictable commitments with respect to $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$ if

$$\mathbb{E}\left(\max_{firstmes_0 \in \mathcal{N}_{\mathsf{com}}} \Pr[firstmes = firstmes_0]\right) \leq \epsilon_{\mathsf{ComGuess}},$$

where the average is taken over $(x, w) \leftarrow\!\!\$\ \mathcal{D}_{\mathcal{X}\times\mathcal{W}}$, and the probability inside is taken over firstmes $\leftarrow\!\!\$\ \Sigma.\mathsf{Prove}(x, w)$.

Formalizing quantum computationally unique responses requires the introduction of an additional definition, *i.e.,* the one of *collapsing relation*. This notion concerns generic relations and is used to argue about the verification stage of the Sigma protocol. Informally, it states that given a relation between two sets $\mathcal{S}_1$ and $\mathcal{S}_2$, if an adversary produces two superpositions S_1 and S_2 that satisfy the relation, then it cannot determine whether S_1 was measured.

Definition 5 (Collapsing Relation [47]). *Let $Rel : \mathcal{S}_1 \times \mathcal{S}_1 \to \{0,1\}$ be a relation over two sets $\mathcal{S}_1$ and $\mathcal{S}_2$ and define the following two games for a two-stage adversary $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$:*

$\mathsf{Game}_1(\mathsf{A})$	$\mathsf{Game}_2(\mathsf{A})$
$(state, \mathsf{S}_1, \mathsf{S}_2) \leftarrow \mathsf{A}_1$	$(state, \mathsf{S}_1, \mathsf{S}_2) \leftarrow \mathsf{A}_1$
$r \leftarrow Rel(\mathsf{S}_1, \mathsf{S}_2)$	$r \leftarrow Rel(\mathsf{S}_1, \mathsf{S}_2)$
$\mathsf{S}_1 \leftarrow \mathcal{M}(\mathsf{S}_1)$	
$\mathsf{S}_2 \leftarrow \mathcal{M}(\mathsf{S}_2)$	$\mathsf{S}_2 \leftarrow \mathcal{M}(\mathsf{S}_2)$
$b \leftarrow \mathsf{A}_2(state, \mathsf{S}_1, \mathsf{S}_2)$	$b \leftarrow \mathsf{A}_2(state, \mathsf{S}_1, \mathsf{S}_2)$
return (b, r)	**return** (b, r)

In the games above, $\mathcal{M}$ denotes a measurement in the computational basis, and applying Rel to quantum registers is done by computing the relation coherently and measuring it. A relation Rel is called ϵ_{Coll}-collapsing from $\mathcal{S}_1$ to $\mathcal{S}_2$ if for every quantum-polynomial-time adversary A, it holds that

$$|\Pr[\mathsf{Game}_1(\mathsf{A}) = (1,1)] - \Pr[\mathsf{Game}_2(\mathsf{A}) = (1,1)]| \leq \epsilon_{\mathsf{Coll}}.$$

Definition 6 (Quantum Computationally Unique Responses [47]). *Let Σ be a Sigma protocol with first message space $\mathcal{N}_{\mathsf{com}}$, challenge space $\mathcal{N}_{\mathsf{ch}}$ and response space $\mathcal{N}_{\mathsf{resp}}$. We say that the protocol Σ has ϵ_{QCUR}-quantum computationally unique responses if the verification predicate $\Sigma.\mathsf{Verify}(x, \cdot, \cdot, \cdot) : \mathcal{N}_{\mathsf{com}} \times \mathcal{N}_{\mathsf{ch}} \times \mathcal{N}_{\mathsf{resp}} \to \{0,1\}$, seen as a relation between $\mathcal{N}_{\mathsf{com}} \times \mathcal{N}_{\mathsf{ch}}$ and $\mathcal{N}_{\mathsf{resp}}$, is ϵ_{QCUR}-collapsing from $\mathcal{N}_{\mathsf{resp}}$ to $\mathcal{N}_{\mathsf{com}} \times \mathcal{N}_{\mathsf{ch}}$.*

3.5 Hard-Instance Generators

Let $\lambda \in \mathbb{N}$ be a security parameter, let $\mathcal{R}$ be a relation, and let $f(1^\lambda)$ be an efficient algorithm. Intuitively, f is a *hard instance generator* if it satisfies two properties. First, it outputs pairs in $\mathcal{R}$ with overwhelming probability. Second, given a public value p generated with f, it should be hard to construct a secret value s' such that $(p, s') \in \mathcal{R}$. We next report the formal definition from [81].

Definition 7 (Hard Instance Generator). *f is called a $(\epsilon_{\mathsf{Rel}}, \epsilon_{\mathsf{HIG}})$-hard instance generator for a relation $\mathcal{R}$ if the following conditions are met:*

- $\mathbb{P}\left[(p,s) \notin \mathcal{R} \mid (p,s) \leftarrow\!\!\$\ f\right] \leq \epsilon_{\mathsf{Rel}}$*;*
- *for every quantum polynomial-time adversary A, then $\mathbb{P}\left[f.\mathsf{HIG}(\mathsf{A}) = 1\right] \leq \epsilon_{\mathsf{HIG}}$, where $f.\mathsf{HIG}(\mathsf{A})$ is the following experiment:*

$f.\mathsf{HIG}(\mathsf{A})$

$(p,s) \leftarrow\!\!\$\ f(1^\lambda)$
$s' \leftarrow\!\!\$\ \mathsf{A}(p)$
if $(p,s') \in \mathcal{R}$
  return 1
return 0

By inspection, the notion of hard instance generator is closely related to that of *one-way function*. Notably, it is also possible to instantiate hard instance generators using weak PRFs. While this result is implicitly mentioned in the literature, the version we provide below explicitly incorporates the parameters of the PRF and the associated relation. In our case, the weak PRF takes values in $\mathcal{Y} = \mathbb{Z}_P^m$. Therefore, we tailor the following lemma to this scenario. However, it is easy to extend the result to generic sets $\mathcal{Y}$ where a distance is defined.

Lemma 1. *Let $F\colon \mathcal{K} \times \mathcal{X} \to \mathbb{Z}_P^m$ be a $(N, \epsilon_{\mathsf{wPRF}})$-weak pseudorandom function for $N = 1$, and with ϵ_{wKR}-weak security against key recovery. Let f be the algorithm that samples k and x at random in $\mathcal{K}$ and $\mathcal{X}$, and outputs the pair $((x, F_k(x)), k)$. Consider the relation $\mathcal{R}_\delta = \{((x,y),k)\colon |F_k(x) - y| \leq \delta\} \subseteq (\mathcal{X} \times \mathbb{Z}_P^m) \times \mathcal{K}$. Then f is a $(\epsilon_{\mathsf{Rel}}, \epsilon_{\mathsf{HIG}})$ hard-instance generator for the relation $\mathcal{R}$, where $\epsilon_{\mathsf{Rel}} = 0$ and*

$$\epsilon_{\mathsf{HIG}} = \epsilon_{\mathsf{wPRF}} + (2\delta+1)^m \cdot \frac{|\mathcal{K}|}{P^m} \cdot \epsilon_{\mathsf{wKR}}.$$

Proof (sketch). First, we observe that f always produces pairs $((x,y),k) \in \mathcal{R}$. This proves the first part of the definition of HIG with $\epsilon_{\mathsf{Rel}} = 0$. For the second part of the definition, we consider the following hybrid experiments:

$\mathsf{Hyb}^{(1)}(\mathsf{A})$ is defined as $f.\mathsf{HIG}(\mathsf{A})$, but the PRF output y is sampled uniformly at random instead of being computed as $y = F_k(x)$.

$\mathsf{Hyb}^{(2)}(\mathsf{A})$ is defined as $\mathsf{Hyb}^{(1)}(\mathsf{A})$, but instead of checking that the key k' generated from A satisfies $|F_{k'}(x) - y| \leq \delta$, it outputs 1 only if $F_{k'}(x) = y$.

Then, we prove that for every quantum polynomial-time adversary A there exists a quantum polynomial-time adversary $\tilde{\mathsf{A}}$ such that

$$\left|\Pr[f.\mathsf{HIG}(\mathsf{A}) = 1] - \Pr[\mathsf{Hyb}^{(1)}(\mathsf{A}) = 1]\right| \leq \epsilon_{\mathsf{wPRF}} \tag{1}$$

$$\Pr[\mathsf{Hyb}^{(1)}(\mathsf{A}) = 1] = (2\delta+1)^m \cdot \Pr[\mathsf{Hyb}^{(2)}(\tilde{\mathsf{A}}) = 1] \tag{2}$$

$$\Pr[\mathsf{Hyb}^{(2)}(\mathsf{A}) = 1] \leq \frac{|\mathcal{K}|}{|\mathbb{Z}_P^m|} \cdot \epsilon_{\mathsf{wKR}}. \tag{3}$$

Notably, the combination of Equation 1, 2 and 3 proves the lemma, so it remains to explain why these equations hold. Equation 1 derives from a standard reduction over the pseudorandomness of F. Equation 2 is true because guessing a key k' yielding one value in the interval $[y-\delta^m, y+\delta^m]$ is $(2\delta+1)^m$ times easier than finding one exactly matching y. Equation 3 is true because finding a key k' such that $F_{k'}(x) = y$ is easier than finding the exact key k used for computing y from x. To quantify how easier it is, we use the fact that there are in average $\frac{|\mathcal{K}|}{|\mathbb{Z}_P^m|}$ keys k' for every possible (x, y). Therefore, finding a key such that $F_{k'}(x) = y$ is $\frac{|\mathcal{K}|}{|\mathbb{Z}_P^m|}$ times easier than finding the exact key used for generating (x, y). □

The complete proof of the above lemma can be found in Section C.1.

3.6 Signatures from Sigma Protocols and Fiat-Shamir with Aborts

Similar to other well-known signatures, our scheme is constructed from a Sigma protocol via the Fiat-Shamir transform [56] with aborts [69] framework, which we present in Fig. 2. This construction is proved correct and post-quantum secure under various combinations of assumptions on the underlying Sigma protocol. In this work, we rely on the results of Devevey *et al.* [43], Fehr *et al.* [47], and Barbosa *et al.* [14], and report them below. Looking ahead, Section 6 contains an instantiation of the theorems from this section with the parameters of the hard-instance generator and the sigma protocol derived in Section 4 and 5.

$\mathsf{S.Gen}(1^\lambda)$

$(x, w) \leftarrow\!\!\$\ \mathcal{D}_{\mathcal{X}\times\mathcal{W}}(1^\lambda)$
$(vk, sk) \leftarrow (x, (x, w))$
return (vk, sk)

$\mathsf{S.Verify}(vk, msg, \sigma)$

$(firstmes, resp) \leftarrow \sigma$
$ch \leftarrow H(firstmes \| msg)$
return $\Sigma.\mathsf{Verify}(vk, firstmes, ch, resp)$

$\mathsf{Sign}_{sk}(msg)$

while $resp = \perp$ **do**
  $(state, firstmes) \leftarrow \Sigma.\mathsf{Prove}_1(sk)$
  $ch \leftarrow\!\!\$\ H(firstmes \| msg)$
  $resp \leftarrow \Sigma.\mathsf{Prove}_2(state, ch)$
$\sigma \leftarrow (firstmes, resp)$
return σ

Fig. 2. The Signature Scheme $\Pi_{\mathsf{Sign}} = (\mathsf{S.Gen}, \mathsf{Sign}, \mathsf{S.Verify})$ constructed via Fiat-Shamir with Abort. $\Sigma = (\Sigma.\mathsf{Prove}_1, \Sigma.\mathsf{Prove}_2, \Sigma.\mathsf{Verify})$ is any Sigma protocol for some relation $\mathcal{R}$. $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$ is any distribution generating a pair of statement and witness in $\mathcal{R}$.

Analyzing correctness in the abort setting is a bit tedious [43], as many parameters come into play. The first one is the completeness parameter $\epsilon_{\Sigma.\mathsf{Comp}}$ of the sigma protocol, as the signature verification is strongly related to the one of the sigma protocol. The probability of abort in an honest execution of the sigma

protocol $\epsilon_{\mathsf{HAbort}}$, together with the number of iterations κ of the Fiat-Shamir with aborts construction, are related to the probability that Sign does not abort. Moreover, a sufficiently high min-entropy of the first message ν guarantees that the challenges generated with the random oracle are uniformly random. This intuition is made formal in Theorem 6 of [43], which we next simplify.

Theorem 1 (Correctness, [43], restated). *Let H be a hash function modeled as a random oracle. Let Σ be a sigma protocol that meets the following properties. First, the probability of aborting in an honest execution should be at most $\epsilon_{\mathsf{HAbort}}$. Moreover, every non-aborting honest execution should be accepted, i.e. $\epsilon_{\Sigma.\mathsf{Comp}}$-average-case completeness holds with $\epsilon_{\Sigma.\mathsf{Comp}} = \epsilon_{\mathsf{HAbort}}$.*[9] *Finally, for every statement and witness couple generated with $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$, the min entropy of the first message should be at least ν. Let Π_{Sign} be the signature scheme constructed from $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$ and Σ via the Fiat-Shamir with Abort transform (cf. Fig. 2). Then, for any (vk, sk) generated with $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$ and any message msg, Π_{Sign} satisfies ϵ_{Corr}-correctness with*

$$\epsilon_{\mathsf{Corr}} = \frac{2^{-\nu}}{(1-\epsilon_{\mathsf{HAbort}})^3}.$$

The next result concerns the security of the signature scheme against no-message attacks. As the presence of aborts does not play a role here, the main challenges in the security analysis arise from the increased capabilities of the adversary in the QROM. In the classical ROM, unforgeability is typically reduced to the soundness of the underlying Sigma protocol by measuring the adversary's oracle queries. However, extending this approach to the quantum world presents additional hurdles, as it is impossible to measure superposition queries without disturbing them. The work of [47] provides a comprehensive treatment of this scenario, and we condense their Theorem 21 and 25 in Theorem 2 below.

The first requirement for unforgeability against no-message attacks (UF-NMA) is that the key pair is generated with an hard-instance generator. This guarantees, intuitively, that an adversary A can hardly recover a valid signing key from the public key only. UF-NMA is achieved in combination with another important property: any adversary capable of forging a signature given only the public key must, with high probability, also know the underlying signing key. In the quantum setting, this property can be reduced to the sigma protocol level up to a multiplicative factor $O(N_H^2)$ and an additive factor $\frac{1}{N_H C}$ (see Theorem 8 in [47]). This results in the sigma protocol property known as *proof of knowledge*. Theorem 21 in [47] formalizes the intuitions given up to this point. Namely, a sigma protocol with the proof of knowledge property and superpolynomial challenge space, combined with a hard-instance generator in the Fiat-Shamir heuristic, guarantees unforgeability against no-message attacks.

In Theorem 25 in [47], the authors investigate the assumptions implying that a sigma protocol is a proof of knowledge, made formal by considering an algorithm E extracting a witness from an interaction with the adversary. In the

[9] Theorem 6 of [43] also holds for the more general case where $\epsilon_{\Sigma.\mathsf{Comp}} \geq \epsilon_{\mathsf{HAbort}}$.

proof of Theorem 25 in [47], E runs the malicious prover A to the point where it outputs the first message. Then, it samples a random challenge and sends it to A, obtaining a response by measuring A's corresponding register. Next, E rewinds A on the measured state, and chooses and sends to A a fresh random challenge, resulting in a response. If both transcripts verify, then E can compute the witness w using the extractor E_{SS} corresponding to the $(s, \epsilon_{\Sigma.\mathsf{SS}})$-special soundness property with $s = 2$. This strategy fails if the fresh challenge equals the first one and if any of the measurements is detected. This motivates the factor $1/C$ and the assumption that the scheme has ϵ_{QCUR}-quantum computationally unique responses. We refer to Theorem 21 and 25 in [47] for a formal treatment and to Section C.5 for a more detailed explanation of how Theorem 2 follows.

Theorem 2 (Unforgeability against no-message attack, [47], restated). *Let Σ be a Sigma protocol for a relation $\mathcal{R}$ with challenge space size C, $(s, \epsilon_{\Sigma.\mathsf{SS}})$-special soundness with $s = 2$ and $\epsilon_{\Sigma.\mathsf{SS}} = 0$ (cf. Definition 2), and ϵ_{QCUR}-quantum computationally unique responses (cf. Definition 6).*[10] *Let $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$ be a ϵ_{HIG}-hard instance generator according to Definition 7, yielding a pair of statements and witnesses in $\mathcal{R}$. Let Π_{Sign} be the signature scheme constructed from $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$ and Σ via the Fiat-Shamir with Abort transform (cf. Fig. 2). Then Π_{Sign} is $(\epsilon_{\mathsf{Sign},0}, N_H)$-unforgeable against no-message attack, with*

$$\epsilon_{\mathsf{Sign},0} \leq \left(\epsilon_{\mathsf{HIG}} \cdot N_H^6 + \frac{1}{C} + 2\epsilon_{\mathsf{QCUR}} + \frac{1}{O(N_H^5)\cdot C} + \frac{1}{8N_H^3 C^3}\right)^{\frac{1}{3}}.$$

Proof (sketch). Theorem 2 is a direct application of two slightly adapted variants of Theorem 25 and Theorem 21 of [47]. Below, we detail the modification to each theorem and motivate why the resulting statements stay valid.

Theorem 25 assumes generic t-soundness (*cf.* [47], section 5), while we assume $(s, \epsilon_{\Sigma.\mathsf{SS}})$-special soundness. Since special soundness implies t-soundness, the theorem continues to hold. Looking ahead, we directly instantiate $(s, \epsilon_{\Sigma.\mathsf{SS}}) = (2, 0)$, as our sigma protocol satisfies special soundness with these particular parameters. Moreover, we make explicit the parameters defining the proof of knowledge (cf. [47], definition 14), yielding $p = 1, d = 3$, and $\kappa = \frac{1}{C} + 2\epsilon_{\mathsf{QCUR}}$. These values are derived from a step-by-step revision of the proof of Theorem 25 and of the underlying technical lemma ([47], Lemma 15). The latter proves that proofs of knowledge for static adversaries extend to adaptive adversaries, and through our revision we prove that this extension leaves the parameters p, d, κ unchanged.

Theorem 21 proves unforgeability against no message attacks for standard Fiat-Shamir signatures, while our variant refers to the Fiat-Shamir *with abort* transform (*cf.* Fig. 2). However, the possibility of abort only reduces the probability that an adversary can generate an accepting signature, therefore we can keep the same bounds. Since we apply Theorem 21 to the proof of knowledge

[10] The authors prove this result for generic $(s, \epsilon_{\Sigma.\mathsf{SS}})$-special soundness. For simplicity, we only report the concrete case of $s = 2$ and $\epsilon_{\Sigma.\mathsf{SS}} = 0$.

from Theorem 25, where $d = 3$, we compute the parameters for Theorem 21 under this assumption, yielding unforgeability with

$$\epsilon_{\mathsf{Sign},0} \leq \left(\epsilon_{\mathsf{HIG}} N_H^6 p(\eta) + \kappa(\eta) + \frac{1}{p(\eta)} \cdot \left(\frac{1}{8N_H^3 C^3} + \frac{1}{\mathcal{O}(N_H^5)C}\right)\right)^{1/3}.$$

The parameter computation derives again from a detailed revision of the proof of Theorem 21 and the underlying Corollary (Corollary 16 in [47]). For the latter, we observe that the proof of knowledge property of $\mathsf{FS}(\Sigma)$ holds with the same parameters p, d as in Σ, and with $e = 6, \mu(\eta) = \kappa(\eta) + \frac{1}{p(\eta)} \cdot \left(\frac{1}{8N_H^3 C^3} + \frac{1}{\mathcal{O}(N_H^5)C}\right)$, where N_H is the number of queries to the random oracles made by the attacker to $\mathsf{FS}(\Sigma)$, under the assumption that $d = 3$. □

Lifting unforgeability against no-message attacks to standard unforgeability requires, first of all, an underlying zero-knowledge sigma protocol. This holds because the adversary can try to forge based on each of the N_{Sign} previous signatures, which in this case are sigma protocol transcripts. In turn, relating the signatures to sigma protocol conversations requires arguing about the randomness of the challenges produced by the RO. This is strictly related to the entropy of the input to the random oracle, which includes the first message, motivating the assumption of $\epsilon_{\mathsf{ComGuess}}$-unpredictable commitments.

At the proof level, using the ZK simulator implies reprogramming the random oracle accordingly, which poses several technical challenges in the quantum setting. The presence of aborts further complicates the analysis, since reprogramming introduces a bias toward first messages corresponding to non-aborting transcripts. Barbosa *et al.* [14] formalize this intuition in their Theorem 2, and we report a slightly simplified version of it below.[11]

Theorem 3 (Unforgeability, [14], restated for Multi-HVZK). *Let N_{Sign}, $N_H \in \mathbb{N}$. Let $\mathcal{R}$ be a relation, and $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$ be a distribution yielding a pair of statements and witnesses in $\mathcal{R}$. Let Σ be an* average-case accepting $(N_{\mathsf{Sign}}, \epsilon_{\Sigma.\mathsf{HVZK}})$-multi-HVZK *Sigma protocol for the relation $\mathcal{R}$ and the distribution $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$. Denote with $\epsilon_{\mathsf{HAbort}}$ the probability of abort in an honest execution of Σ, and with $\epsilon_{\mathsf{ComGuess}}$ the commitment-guessing probability, as defined in Definition 4. Let Π_{Sign} be the signature scheme constructed from $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$ and Σ via the Fiat-Shamir transform with Abort (cf. Fig. 2). Assume that the attacker A to unforgeability makes N_{Sign} classical queries to the signing oracle and N_H quantum queries to the random oracle H. Further, assume that Π_{Sign} is $(\epsilon_{\mathsf{Sign},0}, N_H)$-unforgeable against no-message attacks. Then Π_{Sign} is also $(\epsilon_{\mathsf{Sign},N_{\mathsf{Sign}}}, N_H, N_{\mathsf{Sign}})$-unforgeable, with*

$$\begin{aligned}\epsilon_{\mathsf{Sign},N_{\mathsf{Sign}}} \leq & \epsilon_{\mathsf{Sign},0} + \frac{2N_{\mathsf{Sign}}\sqrt{\epsilon_{\mathsf{ComGuess}}}}{1-\epsilon_{\mathsf{HAbort}}}\sqrt{N_H + 1 + \frac{N_{\mathsf{Sign}}}{1-\epsilon_{\mathsf{HAbort}}}} \\ & + 2(N_H+1)\sqrt{\frac{N_{\mathsf{Sign}}\epsilon_{\mathsf{ComGuess}}}{1-\epsilon_{\mathsf{HAbort}}}} + \epsilon_{\Sigma.\mathsf{HVZK}}.\end{aligned}$$

[11] The main difference is that their result is defined over generic events Γ, while in the following, we restrict to Γ being the event that is always true.

Proof (sketch). The proof is similar to the one in [14]. The only difference is the need to replace the real proofs with simulated ones all at once, which is possible thanks to the new definition of Average-Case Accepting Multi-HVZK. □

We refer the reader to Section C.6 for a more detailed explanation of the differences between the work of [14] and the proof of Theorem 3.

4 Learning with Rounding

The *learning with rounding* (LWR) assumption is a deterministic variant of the well-known *learning with errors* (LWE) assumption, differing in how noise is added to the result. In the LWE case, a small error is added to the product, whereas in LWR, the product is instead *rounded*. Given two integers Q, P such that $Q > P$, the *rounding function* $\lfloor \cdot \rceil_P : \mathbb{Z}_Q \to \mathbb{Z}_P$ is defined as $\lfloor x \rceil_P := \left\lfloor x \cdot \frac{P}{Q} \right\rfloor$, where $\lfloor . \rfloor$ denotes the floor function over rational numbers. We can extend the rounding function $\lfloor \cdot \rceil_P$ to operate on vectors: if $\boldsymbol{x} \in \mathbb{Z}_Q^m$ is a vector, $\lfloor \boldsymbol{x} \rceil_P$ simply applies the $\lfloor \cdot \rceil_P$ to every component, yielding a vector in $\mathbb{Z}_P^m$. The inherent properties of the rounding function, including almost key-homomorphism, form the base for the technical results of this work. They are detailed in Section D.

For parameters $Q, P, n, m \in \mathbb{N}$ such that $Q > P$, and a vector $\boldsymbol{k} \in \mathbb{Z}_Q^n$, we define the LWR distribution (and the corresponding random distribution) as:

$\mathsf{LWR}(Q,P,n,m).\mathsf{Real}(\boldsymbol{k})$	$\mathsf{LWR}(Q,P,n,m).\mathsf{Rand}()$
$1:\ \boldsymbol{X} \leftarrow\!\!\$\ \mathbb{Z}_Q^{m\times n},\ \boldsymbol{y} \leftarrow \lfloor \boldsymbol{X}\boldsymbol{k} \rceil_P$	$\boldsymbol{X} \leftarrow\!\!\$\ \mathbb{Z}_Q^{m\times n},\ \boldsymbol{y} \leftarrow\!\!\$\ \mathbb{Z}_P^m$
$2:\ \textbf{return}\ (\boldsymbol{X}, \boldsymbol{y})$	$\textbf{return}\ (\boldsymbol{X}, \boldsymbol{y})$

When it is clear from the context, we just omit the parameters and write $\mathsf{LWR.Real}(\boldsymbol{k})$ (*resp.*, $\mathsf{LWR.Rand}()$) or just $\mathsf{Real}(\boldsymbol{k})$ (*resp.*, $\mathsf{Rand}()$). The LWR assumption is defined from these experiments, and states that it is hard to distinguish the rounded matrix product from a fresh random vector.

Definition 8 (LWR [13]). *Let $Q, P, n, m \in \mathbb{N}$ such that $Q > P$. The decisional LWR assumption holds with advantage ϵ_{LWR} if, $\forall$ quantum PPT adversary A,*

$$\left| \mathbb{P}\left[\mathsf{A}\left(\mathsf{LWR.Real}(\boldsymbol{k})\right) = 1 \mid \boldsymbol{k} \leftarrow\!\!\$\ \mathbb{Z}_Q^n\right] - \mathbb{P}\left[\mathsf{A}\left(\mathsf{LWR.Rand}()\right) = 1\right] \right| \leq \epsilon_{\mathsf{LWR}}.$$

4.1 A LWR-Based Weak Pseudorandom Function

We now introduce the weak pseudorandom function (wPRF) used in our construction and analyze two key properties: its security as a weak pseudorandom function (wPRF) and as a hard-instance generator. We consider a natural LWR-based wPRF, $F_{\boldsymbol{k}} \colon \mathbb{Z}_Q^n \times \mathbb{Z}_Q^{m\times n} \to \mathbb{Z}_P^m(\boldsymbol{X})$ such that $\boldsymbol{y} \leftarrow \lfloor \boldsymbol{X}\boldsymbol{k} \rceil_P$.

LWR implies that F is a weak PRF, with the remark that an increased number of wPRF queries corresponds to a matrix with more rows.

Lemma 2. *Consider $F_{\boldsymbol{k}}$ with parameters $P, Q, n, m \in \mathbb{N}$ such that $P < Q$. Let $N \in \mathbb{N}$, and assume that the LWR assumption holds with probability ϵ_{LWR} over the same parameters, apart from the number of rows of the random matrix being $\tilde{m} = Nm$. Then F is a $(N, \epsilon_{\mathsf{wPRF}})$-weak PRF, with*

$$\epsilon_{\mathsf{wPRF}} \leq \epsilon_{\mathsf{LWR}}.$$

Proof (sketch). To show that F is a weak PRF, we reduce its security to the LWR assumption. In wPRF.Real and wPRF.Rand the adversary receives N random matrices with either rounded products $\lfloor \boldsymbol{X}_i \boldsymbol{k} \rfloor$ or random outputs. By stacking these matrices (and outputs) row-wise, the distribution is identical to sampling a single large matrix and vector in the LWR experiments. □

The full proof of Lemma 2 is deferred to Section C.2.

As a result, LWR implies that F is a hard instance generator.

Lemma 3. *Consider $F_{\boldsymbol{k}}$ with parameters $P, Q, n, m \in \mathbb{N}$ such that $P < Q$. Assume that the LWR assumption holds with probability ϵ_{LWR} over the same parameters, apart from the number of rows of the random matrix being $\tilde{m} = Nm$, with $N \geq 2$. Let $\mathcal{D}_{\mathcal{X} \times \mathcal{W}}$ be the algorithm that samples $\boldsymbol{k}, \boldsymbol{X}$ at random in $\mathbb{Z}_Q^n \times \mathbb{Z}_Q^{m \times n}$, and outputs the pair $((\boldsymbol{X}, F_{\boldsymbol{k}}(\boldsymbol{X})), \boldsymbol{k})$. Consider the relation $\mathcal{R} = \{((\boldsymbol{X}, \boldsymbol{y}), \boldsymbol{k}) \colon |F_{\boldsymbol{k}}(\boldsymbol{X}) - \boldsymbol{y}| \leq 1\} \subseteq (\mathbb{Z}_Q^{m \times n} \times \mathbb{Z}_P^m) \times \mathbb{Z}_Q^n$. Then f is a $(\epsilon_{\mathsf{Rel}}, \epsilon_{\mathsf{HIG}})$ hard-instance generator for the relation $\mathcal{R}$, where $\epsilon_{\mathsf{Rel}} = 0$ and*

$$\epsilon_{\mathsf{HIG}} \leq \epsilon_{\mathsf{LWR}} \left(1 + 3^m \frac{Q^n}{P^m}\right) + 3^m \frac{Q^n}{P^{2mN}}.$$

Proof (sketch). This is a direct application of Lemma 1, where ϵ_{wPRF} is bounded according to Lemma 2, and ϵ_{wKR} is upper bounded in terms of ϵ_{wPRF} via Lemma 7. The full proof of Lemma 3 appears in Section C.3.

5 A Sigma Protocol Based on LWR

In this section, we introduce our Sigma protocol and prove its completeness, honest-verifier zero-knowledge, special soundness, commitment unpredictability, and quantum computationally unique responses. Our protocol follows a similar approach as previous MPC-in-the-head-based schemes like ZKBoo [58, 36] and ZKB++ [36], but is tailored to the almost key-homomorphic functionality introduced in Section 4.1. Its full description is given in Fig. 3.

We start with a high-level overview on its main features, including the prover's computation of the first message and response, the verification algorithm, and the abort condition. The first message is computed by creating key shares $\boldsymbol{k}_i$'s, and then returning their hash-based commitments h_i's together with the local PRF evaluations $\boldsymbol{y}_i$, for every iteration $j \in [t]$. Upon receiving a challenge, the prover checks whether the abort event holds and opens the committed shares if not. The verifier then checks that the openings are computed correctly, that the

$\Sigma.\mathsf{Prove}_1((\boldsymbol{X},\boldsymbol{y}),\boldsymbol{k})$

for $j \in [t]$
  $(\boldsymbol{k}_1^{(j)},\ldots,\boldsymbol{k}_d^{(j)}) \leftarrow\!\!\$\ \mathsf{Share}(\boldsymbol{k})$
  for $i \in [d]$
    $\boldsymbol{z}_i^{(j)} \leftarrow \mathsf{Output}(\boldsymbol{X},\boldsymbol{k}_i^{(j)},i)$
    $\boldsymbol{y}_i^{(j)} \leftarrow \left\lfloor \boldsymbol{z}_i^{(j)} \right\rfloor_{2^{p-\alpha}}$
    $\rho_i^{(j)} \leftarrow\!\!\$\ \mathcal{P}_{\mathsf{H}}$
    $h_i^{(j)} \leftarrow \mathsf{H}(\boldsymbol{k}_i^{(j)},\rho_i^{(j)})$
$\boldsymbol{y}_{ext} \leftarrow \lfloor \boldsymbol{X}\boldsymbol{k} \rfloor_{2^p}$
$state \leftarrow (\boldsymbol{X},\boldsymbol{y}_{ext},(\boldsymbol{k}_i^{(j)},\rho_i^{(j)})_{i\in[d],j\in[t]})$
$firstmes \leftarrow (\boldsymbol{y}_i^{(j)},h_i^{(j)})_{i\in[d],j\in[t]}$
return $firstmes, state$

$\Sigma.\mathsf{Prove}_2(state,ch)$

$(\boldsymbol{X},\boldsymbol{y}_{ext},(\boldsymbol{k}_i^{(j)},\rho_i^{(j)})_{i\in[d],j\in[t]}) \leftarrow state$
$(\mathcal{I}^{(j)})_{j\in[t]} \leftarrow ch$
if $\mathsf{Abort}(state,ch)=1$
  $resp \leftarrow \perp$
else
  $resp \leftarrow (\boldsymbol{k}_i^{(j)},\rho_i^{(j)})_{i\in\mathcal{I}^{(j)},j\in[t]}$
return $resp$

$\mathsf{Share}(\boldsymbol{k})$

$(\boldsymbol{k}_1,\ldots,\boldsymbol{k}_{d-1}) \leftarrow\!\!\$\ \mathcal{P}_{\mathcal{K}}$
$\boldsymbol{k}_d = \boldsymbol{k} - \sum_{i\in[d-1]} G_{\mathcal{K}}(\boldsymbol{k}_i)$
return $(\boldsymbol{k}_i)_{i\in[d]}$

$\mathsf{Output}(\boldsymbol{X},\tilde{\boldsymbol{k}},i)$

if $i \in [d-1]$
  $\boldsymbol{k} \leftarrow G_{\mathcal{K}}(\tilde{\boldsymbol{k}})$
else
  $\boldsymbol{k} \leftarrow \tilde{\boldsymbol{k}}$
$\boldsymbol{z} \leftarrow \boldsymbol{X}\boldsymbol{k}$
return $\boldsymbol{z}$

$\Sigma.\mathsf{Verify}((\boldsymbol{X},\boldsymbol{y}),firstmes,ch,resp)$

$(\boldsymbol{y}_i^{(j)},h_i^{(j)})_{i\in[d],j\in[t]} \leftarrow firstmes$
$(\mathcal{I}^{(j)})_{j\in[t]} \leftarrow ch$
if $resp = \perp$
  return 0
$(\boldsymbol{k}_i^{(j)},\rho_i^{(j)})_{i\in\mathcal{I}^{(j)},j\in[t]} \leftarrow resp$
for $j \in [t]$
  for $i \in \mathcal{I}^{(j)}$
    if $\mathsf{H}(\boldsymbol{k}_i^{(j)},\rho_i^{(j)}) \neq h_i^{(j)}$
      return 0
    $\boldsymbol{z}_i^{(j)} \leftarrow \mathsf{Output}(\boldsymbol{X},\boldsymbol{k}_i^{(j)},i)$
    if $\left\lfloor \boldsymbol{z}_i^{(j)} \right\rfloor_{2^{p-\alpha}} \neq \boldsymbol{y}_i^{(j)}$
      return 0
  $sum^{(j)} \leftarrow \sum_{i\in\mathcal{I}^{(j)}} \boldsymbol{z}_i^{(j)}$
  $i^* \leftarrow [d] \setminus \mathcal{I}^{(j)}$
  $\tilde{\boldsymbol{y}}^{(j)} = \left\lfloor sum^{(j)} \right\rfloor_{2^{p-\alpha}} + \boldsymbol{y}_{i^*}^{(j)}$
  if $\boldsymbol{y} - \tilde{\boldsymbol{y}}^{(j)} \notin \{0,1\}$
    return 0
return 1

$\mathsf{Abort}(state,ch)$

$(\boldsymbol{X},\boldsymbol{y}_{ext},(\boldsymbol{k}_i^{(j)},\rho_i^{(j)})_{i\in[d],j\in[t]}) \leftarrow state$
$(\mathcal{I}^{(j)})_{j\in[t]} \leftarrow ch$
for $j \in [t]$
  for $i \in \mathcal{I}^{(j)}$
    $\boldsymbol{z}_i^{(j)} \leftarrow \mathsf{Output}(\boldsymbol{X},\boldsymbol{k}_i^{(j)},i)$
  $sum^{(j)} \leftarrow \sum_{i\in\mathcal{I}^{(j)}} \boldsymbol{z}_i^{(j)}$
  if $0 \in sum^{(j)}[r_S..r_E] - \boldsymbol{y}_{ext}[0..\alpha]$
    return 1
return 0

Fig. 3. The Sigma protocol $\Sigma = (\Sigma.\mathsf{Prove}_1, \Sigma.\mathsf{Prove}_2, \Sigma.\mathsf{Verify})$ and its subprocedures $\mathsf{Share}, \mathsf{Output}$ and Abort. The statement is $(\boldsymbol{X},\boldsymbol{y})$ and the witness is $\boldsymbol{k}$. The challenge $ch = (\mathcal{I}^{(j)})_{j\in[t]}$ is a tuple of t subsets of $[d]$ with $d-1$ elements each. In the subprocedures Share and Output, $G_{\mathcal{K}}\colon \mathcal{P}_{\mathcal{K}} \to \mathbb{Z}_{2^q}^n$ is a pseudorandom generator.

$\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$

$\boldsymbol{k} \leftarrow\!\!\$\ \mathbb{Z}_{2^q}^n$

$\boldsymbol{X} \leftarrow\!\!\$\ \mathbb{Z}_Q^{m\times n}$

$\boldsymbol{y} = \lfloor \boldsymbol{X}\boldsymbol{k} \rfloor_{2^{p-\alpha}}$

return $((\boldsymbol{X}, \boldsymbol{y}), \boldsymbol{k})$

Fig. 4. The distribution $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$ generating statements and witnesses for the Σ protocol.

key shares correspond to the output shares, and that the output shares (almost) sum up to $\boldsymbol{y}$, up to a difference $\mathbf{1}$. The abort event can be triggered in any iteration by the equality of any component of two vectors, namely the non-rounded output and the sum of the non-rounded output shares, in a fixed α-sized subset of their bits. This subset corresponds to the most significant α bits cut out by the rounding function, i.e., the ones between $r_S = q - p$ and $r_E = q - p + \alpha$.

Theorem 4 (Completeness). *Let Σ be the sigma protocol described in Fig. 3, $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$ be the distribution described in Fig. 4, and consider the relation $\mathcal{R} = \{((\boldsymbol{X}, \boldsymbol{y}), \boldsymbol{k}) \colon |\boldsymbol{y} - \lfloor \boldsymbol{X}\boldsymbol{k} \rfloor_{2^{p-\alpha}}| \leq 1\}$. Assume that decisional LWR holds with parameters $(2^q, 2^p, n, m)$ and advantage ϵ_{LWR}, and that $G_{\mathcal{K}}$ is a $\epsilon_{\mathsf{PRG},\mathcal{K}}$-secure PRG. Then Σ is a $\epsilon_{\Sigma.\mathsf{Comp}}$-average-case complete sigma protocol for the relation $\mathcal{R}$ and for the distribution $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$, where*

$$\epsilon_{\Sigma.\mathsf{Comp}} = \epsilon_{\mathsf{HAbort}} = tm \cdot 2^{-\alpha} + t \cdot \epsilon_{\mathsf{PRG},\mathcal{K}} + \epsilon_{\mathsf{LWR}},$$

where $\epsilon_{\mathsf{HAbort}}$ denotes the probability of aborting in an honest execution.

Proof (sketch). As long as the abort condition is not triggered and the statement-witness pair is generated with $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$, an honestly-generated proof always passes the verification checks. This makes completeness only depend on the probability of aborts. Recall that the abort condition occurs when, in any of the t iterations and for any of the m components, α bits of two vectors match. Assuming that G is a PRG and that LWR holds with parameters $(2^q, 2^p, n, m)$ ensures that one of these vectors, namely $\boldsymbol{y}_{ext}$, is pseudorandom. This reduces the problem to upper bounding the probability that $t \cdot m$ α-bit strings equal a random one, which is at most $t \cdot m \cdot 2^{-\alpha}$ by union bound. The full proof is deferred to Section E.1.

Proving average-case accepting multi-honest-verifier zero-knowledge poses significant challenges, which we tackle in Section E.2 and summarize in the next lines. In this setting, we wish to prove that an adversary observing a non-aborting transcript hardly gets information about the secret key. The main difficulty comes from the use of an almost-key homomorphic PRF, as potential reconstruction errors could leak information about the secret. Therefore, we need to carefully quantify the possible leakage. As part of the proof we present an algorithm, named Sim^*, able to simulate a full transcript without knowledge of the secret key. However, this comes with certain constraints: Sim^* needs α additional bits of the public key to function properly, and has a rather high probability of

aborting. Due to the first constraint, zero-knowledge reduces to LWR with parameters $(2^q, 2^p, n, m)$, even though the public key itself is only in $\mathbb{Z}_{2^{p-\alpha}}^n$. To deal with the second constraint, we use a common technique from the literature: we define an algorithm Sim that runs Sim^* ω times, drastically reducing the probability of aborts. This motivates the presence of ω in the statement of Theorem 5. The factor $t\epsilon_{\mathsf{Hide}}$ corresponds to the attack where any of the t unopened shares can be recomputed from the first message, leading to complete key leakage. The remaining terms correspond, as in Theorem 4, to the probability of aborts, and the way they appear is influenced by how this probability changes across the hybrid experiments in the proof. Finally, all of the above only holds for every transcript considered separately; when joining all the N transcripts together, we have to take into account a factor N that multiplies everything else.

Theorem 5 (Average-Case Accepting Multi-HVZK). *Let $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$ be the distribution described in Fig. 4, and let $\omega \in \mathbb{N}$. Assume that*

- *decisional LWR holds with parameters $(2^q, 2^p, n, m)$ and advantage ϵ_{LWR},*
- *$G_{\mathcal{K}}$ is a ϵ_{PRG}-secure PRG, and*
- *the canonical commitment scheme for H is ϵ_{Hide}-hiding.*

Consider the relation $\mathcal{R} = \{((\boldsymbol{X}, \boldsymbol{y}), \boldsymbol{k})\colon |F_{\boldsymbol{k}}(\boldsymbol{X}) - \boldsymbol{y}| \leq 1\}$, where F is the weak PRF defined in Section 4.1. Then Σ satisfies $(N, \epsilon_{\Sigma.\mathsf{HVZK}})$-acamHVZK for the relation $\mathcal{R}$ and the distribution $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$, where $N \in \mathbb{N}$ and

$$\epsilon_{\Sigma.\mathsf{HVZK}} = Nt\epsilon_{\mathsf{Hide}} + \frac{2Nt\epsilon_{\mathsf{PRG}} + 4\epsilon_{\mathsf{LWR}}}{1 - N \cdot t \cdot m \cdot 2^{-\alpha}} + 2\gamma^{\omega},$$

where $\gamma \leq N \cdot t \cdot m \cdot 2^{-\alpha}$.

Proof (sketch). We split the experiments in smaller components, and then shift from the real experiment to the simulated one via hybrid arguments that gradually replace such components. Namely, the main components are the secret sharing algorithm Share, the PRG-based sampler Output, the commitment function H, the challenge distribution $\mathcal{N}_{\mathsf{ch}}$, and the LWR problems $\mathsf{LWR}, \mathsf{LWR}'$ with different parameters. We summarize the key steps below.

1. We move the challenge generation at the beginning of the experiment, since the distribution $\mathcal{N}_{\mathsf{ch}}$ does not depend on anything else. This allows to know the index i^* of the unopened share in each iteration from the start.
2. We replace the commitment values $\mathsf{H}(\boldsymbol{k}_{i^*})$ with commitment values of uniform, unrelated values. By reduction to the ϵ_{Hide}-hiding property of H, applied for N messages and t iterations per message, it incurs in a distinguishing advantage $Nt\epsilon_{\mathsf{Hide}}$.
3. Now we consider the event **Abort** that happens when the middle α bits of the reconstructed sum match the ones in the original $\boldsymbol{y}_{ext}$ vector in at least one component (i.e., when the Abort function outputs 1).
   - When $\boldsymbol{y}_{ext}$ comes from $\mathsf{LWR.Rand}$, this event happens with probability $2^{-\alpha}$ for each component of the vector, or, by applying the union bound, with an overall probability of at most $t \cdot m \cdot 2^{-\alpha}$ for each message.

- When $\boldsymbol{y}_{ext}$ comes from a real LWR sample, the probability of Abort for each message is upper bounded by $tm2^{-\alpha} + \epsilon_{\mathsf{LWR}}$, since otherwise detecting whether the event happens or not would already be an attack against LWR with advantage more than ϵ_{LWR}.

If the event **Abort** does not happen, it is then sufficient to know $\boldsymbol{y}_{ext}$ and all the $\boldsymbol{k}_i$ for $i \neq i^*$ to compute $\boldsymbol{y}_{i^*}$ without using $\boldsymbol{k}_{i^*}$.

4. Now that every other dependency to the unopened key share $\boldsymbol{k}_{i^*}$ has been removed, we can replace the pseudorandom output $G_{\mathcal{K}}(\boldsymbol{k}_{i^*})$ in the computation of Share with a uniform, unrelated random value. By reduction to the PRG $G_{\mathcal{K}}$, this incurs in the additional distinguishing advantage $Nt\epsilon_{\mathsf{PRG}}$.
5. The only remaining dependency on the key $\boldsymbol{k}$ is the computation of $\boldsymbol{y}_{ext}$, which contains the additional α for each component. From now on, instead of also getting the key $\boldsymbol{k}$ as input, the experiment can just get the LWR sample $(\boldsymbol{X}, \boldsymbol{y}_{ext})$. For this, we define a (efficiently computable) function Hyb that takes as input the LWR sample and outputs the corresponding transcript.
6. Additionally, we consider a simulator Sim_1 that only takes as input an LWR sample $(\boldsymbol{X}, \boldsymbol{y})$, and randomly samples the additional missing α bits. It is easy to see that, when the sample comes from a random instance, the output of Sim_1 and Hyb is identically distributed:

$$\mathsf{Hyb}\left(\mathsf{LWR}'.\mathsf{Rand}()\right) \equiv \mathsf{Sim}_1\left(\mathsf{LWR}.\mathsf{Rand}()\right).$$

Both Hyb and Sim_1 can be seen as reductions to the respective LWR problems. Therefore, moving from $\mathsf{LWR}'.\mathsf{Real}$ to $\mathsf{LWR}'.\mathsf{Rand}$ first and from $\mathsf{LWR}.\mathsf{Rand}$ to $\mathsf{LWR}.\mathsf{Real}$ later incurs in an additional distinguishing advantage $2\epsilon_{\mathsf{LWR}}$.

Notice that Steps 4 to 6 require that **Abort** does not happen. Let **Success** be the complementary event, and let $\epsilon_{\mathbf{Success}}$ be the distinguishing advantage whenever **Success** happens. A generic reduction to the distinguishing problem between the initial hybrid experiment of Step 4 and the final simulator of Step 6 shows that conditioning on **Success** can increase the advantage by a factor of $2/p_{\mathbf{Success}}$, where $p_{\mathbf{Success}}$ is the probability that **Success** happens. Indeed, by carefully crafting a reduction that cancels the advantage when **Abort** happens, and applying the total probability law, we get

$$\begin{aligned}\epsilon &= |p_{\mathbf{Success}_0} \Pr[\mathsf{A}(\mathcal{P}_0) = 1|\mathbf{Success}_0] - p_{\mathbf{Success}_1} \Pr[\mathsf{A}(\mathcal{P}_1) = 1|\mathbf{Success}_1]| \\ &\geq p_{\mathbf{Success}}\epsilon_{\mathbf{Success}} - \epsilon,\end{aligned}$$

where ϵ is the overall distinguishing advantage before conditioning on **Success**, $\mathbf{Success}_b$ for $b \in \{0,1\}$ is the event that **Success** happens in experiment $\mathcal{P}_b$, $\epsilon_{\mathbf{Success}} = |\Pr[\mathsf{A}(\mathcal{P}_0) = 1|\mathbf{Success}_0] - \Pr[\mathsf{A}(\mathcal{P}_1) = 1|\mathbf{Success}_1]|$, and the inequality comes from the application of the same reasoning as in Step 3: if the difference between $p_{\mathbf{Success}_0}$ and $p_{\mathbf{Success}_1}$ was more than ϵ, simply detecting the event happening would allow for a distinguishing advantage of more than ϵ, therefore we can upper bound this difference with ϵ. Since we are interested in $\epsilon_{\mathbf{Success}}$ and, in our case, the overall advantage after considering the event

Abort is $\epsilon = Nt\epsilon_{\mathsf{PRG}} + 2\epsilon_{\mathsf{LWR}}$, we get that

$$\epsilon_{\mathbf{Success}} = \frac{2Nt\epsilon_{\mathsf{PRG}} + 4\epsilon_{\mathsf{LWR}}}{p_{\mathbf{Success}}},$$

and we are free to choose any event that is not more than ϵ far from any between $\mathbf{Success}_0$ and $\mathbf{Success}_1$; for simplicity, we choose the event caused by the $\mathsf{LWR.Rand}$ experiment, which, as argued in Step 3, happens with probability at most $1 - N \cdot t \cdot m \cdot 2^{-\alpha}$. Finally, in order to get rid of the event **Abort** in the simulator, we construct an additional simulator Sim_ω that runs Sim_1 until a non-aborting transcript is generated, or aborts if the number of attempts exceed ω. □

Our scheme satisfies $(2, 0)$-special soundness, meaning that two accepting transcripts with the same first message and distinct challenges always enable the witness' reconstruction. Intuitively, this happens because two proofs with different challenges provide a complete sharing, and their validity with respect to the same first message guarantees that the reconstructed key is a valid witness. We state this observation in the next theorem and give its proof in Section E.3.

Theorem 6 (Special soundness). *Let Σ be the sigma protocol described in Fig. 3, and consider the relation $\mathcal{R} = \{((\boldsymbol{X}, \boldsymbol{y}), \boldsymbol{k}) \colon |\boldsymbol{y} - \lfloor \boldsymbol{X}\boldsymbol{k} \rfloor_{2^{p-\alpha}}| \leq 1\}$. Then Σ is a sigma protocol with $(s, \epsilon_{\Sigma.\mathsf{SS}})$-special soundness (cf. Definition 2) for $\mathcal{R}$, where $s = 2$ and $\epsilon_{\Sigma.\mathsf{SS}} = 0$.*

Proof (sketch). We observe that having $s = 2$ accepting proofs for the same commitment and different challenges provides the extractor with a complete sharing of a candidate witness $\boldsymbol{k}$. Therefore, extraction can be finalized by reconstructing $\boldsymbol{k}$ from its sharing. The validity of each transcript with respect to the same statement and first message, together with the almost key homomorphism of the rounding function, imply that this extraction strategy always provides a valid witness for the given statement. □

A fundamental property of our sigma protocol is having ϵ_{QCUR}-quantum computationally unique responses, ensuring hardness in finding two accepting transcripts with the same first message, same challenge, but distinct responses. Looking ahead, this relates to a powerful attack to the unforgeability of the signature scheme: the attacker could observe an honestly generated signature (which is a proof), and forge a valid one by just changing the response.

This attack is hard if the responses and the rest of the transcripts are related in a binding way. This is the reason why many MPCith-based schemes, including ours, hash the response with a collapsing hash function, and include the result in the first message. We state the corresponding theorem in the following with a sketch of proof and defer the complete proof to Section E.4.

Theorem 7 (Quantum Computationally Unique Responses). *Let Σ be the Sigma protocol defined in Fig. 3. Assume that the hash functions used are ϵ_{Coll}-collapsing (cf. Definition 13). Then Σ has ϵ_{QCUR}-quantum computationally unique responses, with*

$$\epsilon_{\mathsf{QCUR}} \leq t(d-1) \cdot \epsilon_{\mathsf{Coll}}.$$

Proof (sketch). This proof is very similar to the one provided for Picnic1 in Corollary 26 of [47]. To prove quantum computationally unique responses, we must argue about the event where an adversary provides two different responses for the same first message and challenge, and about the detection of a measurement in the computational basis. Notably, the response of our sigma protocol only contains preimages of the hash function. Therefore, the QCUR property reduces to the collapsingness of H on each of the $(d-1)t$ opened values. Using the bound on the concurrent composition of collapsing functions (Lemma 4, [51]), the theorem follows. □

Finally, we prove that our scheme has unpredictable commitments. Looking ahead to the signature scheme, this property ensures that the input to the Random Oracle (RO) is sufficiently random, thereby guaranteeing that the RO-generated challenge is also random. This allows to relate the correctness of the signature to the completeness of the sigma protocol and to prevent the forger from crafting "comfortable" challenges. Similarly to other MPCith-based scheme, ours has highly unpredictable commitments. This is because each iteration involves the sampling of a new sharing, and each first message contains the evaluation of such shares with a collapsing hash function. We make this intuition formal in the following theorem, and provide the full proof in Section E.5.

Theorem 8 (Unpredictable Commitments). *Let Σ be the Sigma protocol of Fig. 3. Assume that the hash function used to instantiate it is ϵ_{Coll}-collapsing (cf. Definition 13). Then Σ has $\epsilon_{\mathsf{ComGuess}}$-unpredictable commitments, with*

$$\epsilon_{\mathsf{ComGuess}} \leq \left(2\epsilon_{\mathsf{Coll}}\left(1-\frac{1}{|\mathcal{P}_{\mathcal{K}}|}\right)+\frac{1}{|\mathcal{P}_{\mathcal{K}}|}\right)^{(d-1)t}.$$

In other words, for every couple of statement and witness generated with $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$, the min entropy ν of the first message is at least

$$-\log\left[\left(2\epsilon_{\mathsf{Coll}}\left(1-\frac{1}{|\mathcal{P}_{\mathcal{K}}|}\right)+\frac{1}{|\mathcal{P}_{\mathcal{K}}|}\right)^{(d-1)t}\right].$$

Proof (sketch). To prove commitment unpredictability, we upper bound the probability that an honest prover outputs a fixed first message $\mathit{firstmes}^*$. This happens only if all hash values match those in $\mathit{firstmes}^*$. In turn, this requires that either all the PRG seeds equal those underlying $\mathit{firstmes}^*$, or that some of the PRG seeds differ but still yield identical hash outputs. Producing the same hash from two different seeds corresponds to finding a collision, which happens with probability at most $2\epsilon_{\mathsf{Coll}}$ for ϵ_{Coll}- collapsing hash functions (cf. Lemma 24 of [82]). Combining the independent contributions of all seeds and applying the binomial theorem, we obtain the theorem bound. □

6 Signatures from Sigma Protocols and HIGs

In this section, we provide the formal security analysis of our signature scheme. Similarly to other well-known signatures, our scheme is constructed from a sigma

protocol and a hard instance via the Fiat-Shamir transform with aborts (see Section 3.6). Specifically, we build on the sigma protocol defined in Section 5 and the hard-instance generator of Section 4.1. This construction is proved correct and post-quantum secure under various combinations of assumptions on the underlying sigma protocol. We rely on the results of Devevey *et al.* [43], Fehr *et al.* [47], and Barbosa *et al.* [14] (reported in Section 3.6). Plugging our parameters from Section 4.1 and 5 into their theorems, we get the following bounds.

Theorem 9 (Correctness of Bittersweet Signatures). *Let Σ be the sigma protocol defined in Section 5, and $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$ be the hard-instance generator of Section 4.1. Let Π_{Sign} be the signature scheme constructed from $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$ and Σ using the Fiat-Shamir transform with Abort (cf. Fig. 2). Then Π_{Sign} is ϵ_{Corr}-correct, where ϵ_{Corr} is at most*

$$\frac{\left(2\epsilon_{\mathsf{Coll}}\left(1-\frac{1}{|\mathcal{P}_{\mathcal{K}}|}\right)+\frac{1}{|\mathcal{P}_{\mathcal{K}}|}\right)^{(d-1)t}}{(1-tm\cdot 2^{-\alpha}+t\cdot\epsilon_{\mathsf{PRG},\mathcal{K}}+\epsilon_{\mathsf{LWR}})^3}.$$

Theorem 10 (Unforgeability of Bittersweet Signatures). *Let Σ be the sigma protocol defined in Section 5, and $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$ be the hard-instance generator of Section 4.1. Let Π_{Sign} be the signature scheme constructed from $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$ and Σ using the Fiat-Shamir transform with Abort (cf. Fig. 2). Then Π_{Sign} is $(\epsilon_{\mathsf{Sign},N_{\mathsf{Sign}}}, N_H, N_{\mathsf{Sign}})$-unforgeable, with*

$$\begin{aligned}
\epsilon_{\mathsf{Sign},N_{\mathsf{Sign}}} \leq \Bigg(&\left(\epsilon_{\mathsf{LWR}}^{(p-\alpha)}\left(1+3^m\frac{2^{qn}}{2^{(p-\alpha)m}}\right)+3^m\frac{2^{qn}}{2^{2qnN}}\right)\cdot N_H^6\\
&+\frac{1}{d^t}+2t(d-1)\cdot\epsilon_{\mathsf{Coll}}+\frac{1}{O(N_H^5)\cdot C}+\frac{1}{8N_H^3C^3}\Bigg)^{\frac{1}{3}}\\
&+\frac{2N_{\mathsf{Sign}}\sqrt{\left(2\epsilon_{\mathsf{Coll}}\left(1-\frac{1}{|\mathcal{P}_{\mathcal{K}}|}\right)+\frac{1}{|\mathcal{P}_{\mathcal{K}}|}\right)^{(d-1)t}}}{1-(tm\cdot 2^{-\alpha}+t\cdot\epsilon_{\mathsf{PRG},\mathcal{K}}+\epsilon_{\mathsf{LWR}})}\\
&\cdot\sqrt{N_H+1+\frac{N_{\mathsf{Sign}}}{1-(tm\cdot 2^{-\alpha}+t\cdot\epsilon_{\mathsf{PRG},\mathcal{K}}+\epsilon_{\mathsf{LWR}})}}\\
&+2(N_H+1)\sqrt{\frac{N_{\mathsf{Sign}}\left(2\epsilon_{\mathsf{Coll}}\left(1-\frac{1}{|\mathcal{P}_{\mathcal{K}}|}\right)+\frac{1}{|\mathcal{P}_{\mathcal{K}}|}\right)^{(d-1)t}}{1-(tm\cdot 2^{-\alpha}+t\cdot\epsilon_{\mathsf{PRG},\mathcal{K}}+\epsilon_{\mathsf{LWR}})}}\\
&+N_{\mathsf{Sign}}t\epsilon_{\mathsf{Hide}}+\frac{2N_{\mathsf{Sign}}t\epsilon_{\mathsf{PRG}}+4\epsilon_{\mathsf{LWR}}}{1-N_{\mathsf{Sign}}\cdot t\cdot m\cdot 2^{-\alpha}}+2\gamma^{\omega},
\end{aligned}$$

where $\gamma \leq N\cdot t\cdot m\cdot 2^{-\alpha}$, and ω is any value in $\mathbb{N}$ related to the (polynomial) runtime of the simulator.

The proofs of Theorem 9 and Theorem 10 are in Section C.7 and Section C.8.

7 Instantiation and Performance

We complete the paper by examining the instantiation of Bittersweet signatures. We start with a more operational description, follow with a discussion of parameters' choices and conclude with performance evaluation.

7.1 Operational Specifications

Let us first define a function that outputs m LWR samples:

$$\mathsf{LWR}_{\boldsymbol{k}}(x) : \lambda \times \mathbb{Z}_{2^q}^n \to \mathbb{Z}_{2^{p-\alpha}}^n : x \mapsto \boldsymbol{y} = \lfloor \mathsf{G}(x) \cdot \boldsymbol{k} \rfloor_{2^{p-\alpha}} ,$$

where $\mathsf{G}(x)$ is a function that expands the λ-bit input x into a $m \times n$ matrix $\boldsymbol{X} \in \mathbb{Z}_{2^q}^{m \times n}$, typically instantiable with an eXtendable Output Function (XOF) [75].

The Bittersweet signatures' blueprint is given in Figure 5, which illustrates the parallel nature of their design. It is based on the following steps, which essentially match the descriptions in the previous sections, with a few additional details (separation bits, salt, ...) mimicking what can be found in Picnic [36, 35].

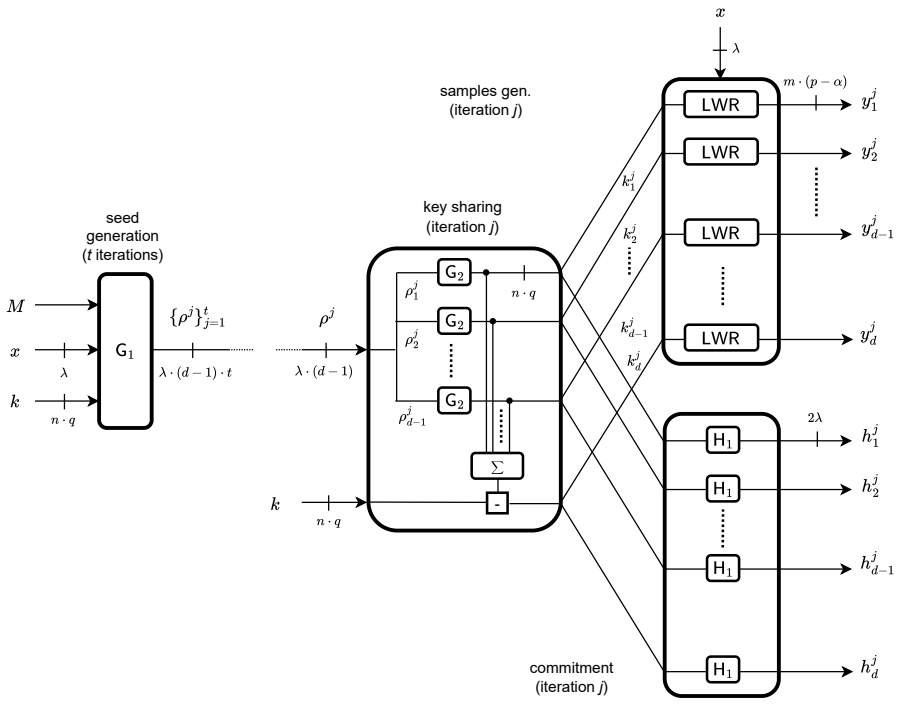

Fig. 5. Bittersweet signatures' blueprint.

Key generation. Pick up a plaintext $x \xleftarrow{\$} \{0,1\}^\lambda$ and a secret key $\boldsymbol{k} \xleftarrow{\$} \mathbb{Z}_{2^q}^n$ uniformly at random. Compute the public key $pk := (x, \boldsymbol{y})$ with $\boldsymbol{y} = \mathsf{LWR}_{\boldsymbol{k}}(x)$.

Seeds generation. Compute seeds $\{\rho^j\}_{j=1}^t = \mathsf{G}_1(M, x, \boldsymbol{k}, salt)$, where M is the message to sign, x the plaintext part of the public key, $\boldsymbol{k}$ the secret (signature) key, the λ-bit $salt$ is used to prevent multi-target attacks [45] and t is the number of iterations of the signature process. The resulting seeds have bitsize $\lambda \cdot (d-1) \cdot t$, with d being the number of parties. For each iteration j, the seed ρ^j is split into $d-1$ equal parts denoted as ρ_i^j, $1 \leq i \leq d-1$, each of them having bitsize λ.

Key sharing. For each iteration j, $1 \leq j \leq t$ and for each party i, $1 \leq i \leq d-1$, compute the expanded key shares $\boldsymbol{k}_i^j = \mathsf{G}_2(\rho_i^j, i, j, salt)$ and $\boldsymbol{k}_d^j = \boldsymbol{k} - \sum_{i=1}^{d-1} \boldsymbol{k}_i \mod 2^q$, so that every key share $\boldsymbol{k}_i^j$, $1 \leq i \leq d$, $1 \leq j \leq t$, has bitsize $n \cdot q$.

Commitment. For each iteration j, $1 \leq j \leq t$, commit on the key share by computing $h_i^j = \mathsf{H}_1(\boldsymbol{k}_i^j, i, j, salt)$, $1 \leq i \leq d$, so that each h_i^j has bitsize 2λ.

LWR samples generation. For each iteration j, $1 \leq j \leq t$, compute the output shares $\boldsymbol{y}_i^j = \mathsf{LWR}_{\boldsymbol{k}_i}(x)$, $1 \leq i \leq d$, so that each $\boldsymbol{y}_i^j$ has bitsize $m \cdot (p - \alpha)$.

Challenge. Compute the hash of the message to sign M, the commitments $h_{1:d}^{1:t}$, the output shares $\boldsymbol{y}_{1:d}^{1:t}$, the signer's public key $(x, \boldsymbol{y})$ and the salt as:

$$c = \mathsf{H}_2(\boldsymbol{y}_{1:d}^{1:t}, h_{1:d}^{1:t}, M, x, \boldsymbol{y}, salt).$$

Signature assembly. Assume $d = 2^\delta$ and $|h| = \delta t$ for simplicity. For each iteration j, denote the δ-bit index of the key share that is not opened as $nop(j)$, and the indexes of the $d-1$ key shares that are opened as $op(j)$. The signature σ is obtained by concatenating the $salt$ with all the output shares $\boldsymbol{y}_{1:d}^{1:t}$ and commitments $h_{1:d}^{1:t}$ as well as the opened key shares $\{\boldsymbol{k}_{op(j)}^j\}_{j=1}^t$.

$$\sigma = (salt \mid\mid \boldsymbol{y}_{1:d}^{1:t} \mid\mid h_{1:d}^{1:t} \mid\mid \{\boldsymbol{k}_{op(j)}^j\}_{j=1}^t)$$

Aborts. For each iteration j, $1 \leq j \leq t$, compute the first addend of the right-hand-side of Eq. (4), this time rounded to p (rather than $p - \alpha$) bits, *i.e.*, $\left\lfloor \sum_{op(j)} \mathsf{G}(x) \cdot \boldsymbol{k}_{op(j)}^j \mod 2^q \right\rfloor_{2^p}$. Compare it with $\boldsymbol{y}_{ext}$, which is the extended representation of $\boldsymbol{y}$ where $\mathsf{G}(x) \cdot \boldsymbol{k}$ is rounded to p (rather than $p - \alpha$) bits. Abort if the α LSBs of these values match in any of the m vector components.

Verification. Derive the challenge c from the signature. For each iteration j, $1 \leq j \leq t$, recompute the commitments and output shares of the opened parties using $\{\boldsymbol{k}_{op(j)}^j\}_{j=1}^t$. Then check that they correspond to the ones present in the signature using the indexing associated to the challenge. If this holds, compute the noisy outputs for each iteration:

$$\tilde{\boldsymbol{y}}^j = \left\lfloor \sum_{op(j)} \mathsf{G}(x) \cdot \boldsymbol{k}_{op(j)}^j \mod 2^q \right\rfloor_{2^{p-\alpha}} + \boldsymbol{y}_{nop(j)}^j \mod 2^{p-\alpha}. \tag{4}$$

If the inequality $\boldsymbol{y}^j - \tilde{\boldsymbol{y}}^j \leq 1$ holds component-wise, accept the signature.

Optimizations. We use the next optimizations To reduce the signature size. First, the output share of the opened parties $\{\boldsymbol{y}_{op(j)}^j\}_{j=1}^t$ can be removed from the

signature since they can be recomputed using the opened key shares $\{\boldsymbol{k}^j_{op(j)}\}^t_{j=1}$ and the public key x. Second, the commitments of the opened parties $\{h^j_{op(j)}\}^t_{j=1}$ can also be removed from the signature as they can be recomputed from the opened key shares $\{\boldsymbol{k}^j_{op(j)}\}^t_{j=1}$. To guarantee that these opened key shares were correctly committed, we include the challenge c in the signature. This challenge acts as a "commitment to the commitments" [36], since the verifier checks that the recomputed commitments are consistent by verifying that the recalculated challenge matches the one present in the signature. Third, the size of the first $d-1$ key shares can be reduced by transmitting the corresponding seeds. Using a GGM-tree, only the root seed of the tree must be sent if the last party remains unopened. Otherwise, the last key share is sent with the sibling-path of the GGM tree associated to the unopened party, which consists of $\log_2(d)$ seeds.

By contrast, while most MPCitH schemes can get rid of the unopened output share by deriving it from the public statement and the opened output shares, this cannot be applied to Bittersweet due to presence of a small reconstruction error. Yet, since the unopened key shares $\boldsymbol{y}^j_{nop(j)}$ can be computed from the error vector $\boldsymbol{e}^j = \boldsymbol{y}^j - \tilde{\boldsymbol{y}}^j$, we can replace each $\boldsymbol{y}^j_{nop(j)}$ by $\boldsymbol{e}^j$ to reduce its size.

Signature size. Using the different aforementioned optimizations, the averaged size (in bits) of Bittersweet signatures is:

$$
\begin{aligned}
|\sigma| = t \cdot \left(\frac{d-1}{d} \cdot \left(n \cdot q + \lceil \log_2(d) \rceil \cdot \lambda\right) + \frac{1}{d} \cdot \lambda \right) \quad &\rightarrow \text{ seeds and key shares} \\
+ t \cdot m \quad &\rightarrow \text{ error vectors} \\
+ t \cdot 2\lambda \quad &\rightarrow \text{ commitments} \\
+ \lambda + t \cdot \lceil \log_2(d) \rceil \quad &\rightarrow \text{ salt and challenge}
\end{aligned}
$$

7.2 Parameters' Choices

The functions and parameters of Bittersweet signatures depend on the security target that can be bounded using the theorems in the previous sections. We next show how they can be selected concretely by identifying the dominating terms of our main bounds, making clear when heuristic choices are operated.

First, the hiding guarantees of the commitments and the quantum computationally unique responses requirement drive the choice of the hash function. We use SHA3-256 (*resp.*, SHA3-384 and SHA3-512) for NIST's security level 1 (*resp.*, 3, 5). Similarly, the PRG security drives the selection of the XOF. We use SHAKE-128 (*resp.*, SHAKE-256) for NIST's security level 1 (*resp.*, 3, 5).

The next step is to select the LWR instance. The most demanding requirement in this respect comes from the ZK analysis of Theorem 5, where security is required for an instance of LWR rounding to p bits, despite the specifications of the previous subsection all round to $p' = p - \alpha$ bits (which is sufficient to make it hard to recover $\boldsymbol{k}$ from $\boldsymbol{y}$). The reason of this requirement is intuitive. First rewrite Eq. (4) as $\tilde{\boldsymbol{y}}^j = \tilde{\boldsymbol{y}}^j_{\text{rec}} + \tilde{\boldsymbol{y}}^j_{nop(j)} \mod 2^{p'}$. Then, write

$$\begin{array}{rcccccc} \tilde{\boldsymbol{y}}^j_{\text{rec}} = & \overbrace{\mathsf{MSBs}(\tilde{\boldsymbol{y}}^j_{\text{rec}})}^{p' \text{ bits}} & \| & \overbrace{\mathsf{buffer}(\tilde{\boldsymbol{y}}^j_{\text{rec}})}^{\alpha \text{ bits}} & \| & \overbrace{\mathsf{LSBs}(\tilde{\boldsymbol{y}}^j_{\text{rec}})}^{q-p \text{ bits}}, \\ \boldsymbol{y} = & \mathsf{MSBs}(\boldsymbol{y}) & \| & \mathsf{buffer}(\boldsymbol{y}) & \| & \mathsf{LSBs}(\boldsymbol{y}). \end{array}$$

It allows expressing the abort condition as checking that the α-bit buffer differs for all components of $\tilde{\boldsymbol{y}}^j_{\text{rec}}$ and $\boldsymbol{y}$. Such a condition is necessary to ensure that information on the (secret) LSBs of $\boldsymbol{y}$ cannot be forwarded to its (public) MSBs due to carry propagation. This is because each time an adversary observes a signature (that did not abort), it can reject one value of the buffer. After observing a number of signatures N_{Sign} which is a small multiple of $2^{\alpha - \log_2(t)}$, one can get the buffer in full, justifying the requirement of Theorem 5.

It is nevertheless also possible to limit the carry leakage due to the observation of N_{Sign} signatures by increasing the buffer size. Say for example that $q = 96$, $\alpha = 90$, $p' = 3$ and N_{Sign} is bounded by the application to 2^{32} (more conservative) or 2^{64} (more aggressive). Then, one can consider Bittersweet signatures relying on a LWR instance that is rounding to p' bits with a small leakage on the buffer. Namely, the statistical distance between the uniform buffer distribution and the leaky one is $\frac{N_{\mathsf{Sign}}}{2^\alpha}$ for each component, and roughly m times more for the full vectors, leading to an abort probability of $P_{\text{abort}} \approx m \cdot t \cdot 2^{-\alpha}$. We posit that this leakage does not significantly affect the security of the corresponding LWR instance as long as N_{Sign} is (very) small in front of 2^α (*e.g.,* 2^{32} or 2^{64} in front of 2^{90} in the conservative and aggressive parameters that we propose above).

Albrecht et al.'s estimator. We now detail how we used the estimator from Albrecht *et al.* [5] to select LWR parameters, as it provides security estimates against a range of known attacks on LWE instances. We first show how to convert the parameters of an LWR problem into an equivalent LWE instance. Then we apply the estimator to find parameter sets (q, p', n) for each security level.

Converting LWR to LWE. As shown in [4], an LWR instance of the form $\boldsymbol{y} = \left\lfloor \boldsymbol{X}\boldsymbol{k}\frac{P'}{Q} \right\rceil$ can be transformed into an LWE instance $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{k} + \mathbf{e}$, where the error vector is sampled from $\mathbf{e} \xleftarrow{\$} \mathcal{U}^m_{Q,P'}$, with $\mathcal{U}_{Q,P'}$ the discrete uniform distribution over the set $[-\frac{Q}{2 \cdot P'} + 1; \frac{Q}{2 \cdot P'}]$. Therefore, a distribution $\mathsf{LWR}\,(Q, P', n, m)$ with a secret $\boldsymbol{k} \in \mathbb{Z}^n_Q$ can be converted, without loss of security, into a LWE distribution $\mathsf{LWE}\,(Q, n, m)$ with same secret distribution and where $\mathbf{e} \xleftarrow{\$} \mathcal{U}^m_{Q,P'}$. The estimator from [5] evaluates the security of an LWE instance based on its parameters (Q, n, m) and the standard deviation of the secret and noise distribution (denoted by σ_k and σ_e). Table 1 summarizes the relationship between those parameters and the corresponding LWR instance with rounding from q to p'.

Table 1. LWE parameters derived from an LWR instance

Q	σ_k	σ_e
2^q	$\sqrt{\frac{(2^q)^2-1}{12}}$	$\sqrt{\frac{(2^{q-p'})^2-1}{12}}$

Proposed instances. To choose the parameters (q, p', n), we first impose some restrictions on q. In order to support signing up to 2^{64} messages while limiting carry leakage, we require $q \gg 64$. Moreover, we restrict q to multiples of 16 to leverage efficient SIMD multiplication instructions on 16 and 32 bits words in software execution. We additionally cap q at 128 and therefore consider the candidate set $q \in \{80, 96, 112, 128\}$. For p', we limit the choices to $p' \in \{2, 3, 4\}$ as it provides small abort probability and small signature sizes.

Using the LWE estimator [5], we determine the smallest n for each parameter pair (q, p') that achieves the three NIST security levels. The resulting security estimates and signature sizes are summarized in Section F, Table 9 where the parameter set $(q, p') = (96, 3)$ yields the smallest signature size for security level L1 and L3 and relatively small signature size for L5, making it a natural choice for our Bittersweet instances. From these parameters, we can compute the number of samples m needed to ensure a hard instance thanks to Lemma 1 or 3: $m = \left\lceil \frac{n \cdot q}{p' - \log_2(3)} \right\rceil$, meaning $m = 747$ (*resp.,* 1222, 1697) for security level L1 (*resp.,* L3, L5).[12] The number of iteration t is finally chosen such that the soundness error, achieved with the multiple iterations, gets smaller than the security level $t = \lceil \frac{\lambda}{\log_2(d)} \rceil$. These parameters are in Table 2 for $d = \{32, 256\}$.

Table 2. Bittersweet parameters set for NIST security levels L1, L3 and L5.

Name	NIST Security (bits)	Parameters								Sizes (Bytes)		
		LWR					MPCitH					
	λ	q	p'	α	n	m	d	t	P_{abort}	$\lvert sk \rvert$	$\lvert pk \rvert$	$\lvert \sigma \rvert$
Bittersweet L1	128	96	3	90	11	747	32	26	$2^{-75.7}$	132	297	8 645
							256	16	$2^{-76.4}$	132	297	6 183
Bittersweet L3	192	96	3	90	18	1222	32	39	$2^{-74.4}$	216	483	20 602
							256	24	$2^{-75.2}$	216	483	14 622
Bittersweet L5	256	96	3	90	25	1697	32	52	$2^{-73.6}$	300	669	37 648
							256	32	$2^{-74.3}$	300	669	26 627

7.3 Performances Evaluation

We now study the size and computation time of Bittersweet signatures in function of the number of parties d for the NIST's three security levels.

Performances. The signing time is dominated, in decreasing order, by:

[12] As in the case of Picnic 2, this choice of parameters ignores the N_H^6 factor and the cubic root loss found in Theorem 10, which would both result in overly large parameters and are conjectured to be due to a non-tight security proof.

- $d \cdot m \cdot t$ inner products between (nq)-bit vectors.
- $(d-1) \cdot t$ extensions/XOFs from λ-bit seeds to (nq)-bit key shares.
- $d \cdot t$ commitments (i.e., hashes from $n \cdot q$ bits to 2λ bits).
- The generation of the (nqm)-bit public matrices (with a XOF).

In order to highlight the interest of Bittersweet signatures' simple design for hardware implementations, we implemented the three versions on a Xilinx Artix-7 FPGA (XC7A200T-3) and synthetized them with Vivado v2022.1. The hardware comparison between Bittersweet and other signature schemes, for which FPGA implementations are available, is given in Table 4 for performances and Table 5 for resources' utilization. For completeness, software performance comparisons between Bittersweet and other signature schemes are given in Tables 6 to 8 for all security levels. Cycle counts for Bittersweet are obtained using an Intel Core i7-1355U CPU with 16 GB of RAM, running Ubuntu 22.04.5. The code is compiled using GCC 11.4.0 with optimization flags `-O3 -march=native -std=gnu11`, on a system supporting SSE2 and AVX2 intrinsics.

In general, increasingly d leads to a tradeoff between signature size and computation time. Increasing the security levels leads to increased signature size due to the independent increase of the vector size n (to have a secure LWR instance) and the number of iterations t (to decrease the soundness error). The signing and verification processes are dominated by the inner product computations with a large modulus $Q = 2^{96}$ and the rejection probability is always below 2^{-70}.

As for the comparison with natural competitors, our main conclusion is that while Bittersweet signatures lead to mild overheads in the software case, they gain interest in the hardware case. In particular, Bittersweet instances consistently lead to better signature sizes compared with MPCitH signatures like Picnic [63, 68] and SDitH [42], while maintaining comparable performance and hardware footprint. Hardware performances could even be further improved by increasing the level of parallelism, for example by instantiating additional multipliers. By contrast, Bittersweet admittedly lags behind Mirath, which is based on TCitH framework [55], even in the hardware implementation context [71].

Finally, Dilithium remains out of reach from the signature size and performances' viewpoint (at the cost of larger private and public keys). Yet, on the one hand, Bittersweet signatures do not rely on structured lattices, which explains most of the performance gap. Existing examples of signature schemes relying on unstructured lattices are indeed known to be significantly more expensive as highlighted in Table 3. So the fact that Bittersweet signatures are competitive without such structured assumptions is actually interesting from the "not putting all eggs in the same basket" viewpoint. On the other hand, their potential for leakage-resilient implementations, which are known to be expensive for Dilithium [11], should reduce this gap. Not only because the key-homomorphic nature of its computations should make masking significantly simpler and more efficient than for MPC-in-the-head signatures based on symmetric cryptographic primitives [78, 10]. Also because the large modulus $Q = 2^{96}$ should make the

recombination of the information leaked by different shares significantly more difficult, especially with inner product masking [50]. The combination of these features with the optimizations and generalizations mentioned in the introduction could make Bittersweet signatures a solution of choice for hardware and leakage-resilient implementations with improved (physical) security vs. cost tradeoff, which is therefore an interesting direction for further research.

Table 3. Comparison of key and signature sizes for unstructured lattice-based signature schemes targeting 128-bit security (NIST security Level 1).

Scheme	Assumption	Size (Bytes)		
		$\|sk\|$	$\|pk\|$	$\|\sigma\|$
Bittersweet L1	LWR	132	297	6 183
TESLA-1 [7]	LWE	4 332 000	11 559 900	2 444
Bai and Galbraith [12]	LWE/SIS	1 272 917	2 119 519	1 553
Lyubashevsky [70]	LWE/SIS	561 917	92 682	1 813
`Squirrels`-I [49]	LWE/SIS	2 593	681 780	1 019

Acknowledgments. We are grateful to Martin Albrecht, Cecilia Boschini, Yu-Hsuan Huang, and Andreas Hülsing for our very insightful discussions. Carmit Hazay is supported by ISF grant No. 607/25. François-Xavier Standaert is a research director of the F.R.S.-FNRS. This work has been funded by the German Research Foundation (DFG) CRC 1119 CROSSING (project S7), by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE, and by the European Research Concil (ERC) Consolidator Grant 101044770 (CRYPTOLAYER) and Advanced Grant 101096871 (BRIDGE). Views and opinions expressed in the paper are those of the authors and do not necessarily reflect those of the European Union nor the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

Table 4. Hardware performance comparison of Bittersweet signatures with other MPCitH-, TCitH- and lattice-based signature schemes evaluated on Xililinx Artix-7 (XC7A200T-3) platform. Schemes marked with † have been evalutated on a Xilinx Kintex-7 (XC7K480T-3). The latencies are expressed in Mcycles and the times in ms.

Framework	**Assumption**	**Scheme**		d	**KeyGen**		**Sign**		**Verify**		**Size (Bytes)**		
					Latency	Time	Latency	Time	Latency	Time	$\|sk\|$	$\|pk\|$	$\|\sigma\|$
MPCitH & Fiat-Shamir with aborts	LWR	Bittersweet	L1	32	0.015	0.12	1.06	8.48	0.91	7.28	132	297	8 645
			L3		0.049	0.39	2.43	19.44	2.01	16.08	216	483	20 602
			L5		0.095	0.76	4.54	36.32	3.78	30.24	300	669	37 648
			L1	256	0.015	0.12	5.19	41.25	4.39	35.12	132	297	6 183
			L3		0.049	0.39	11.79	94.32	10.03	80.24	216	483	14 622
			L5		0.095	0.76	22.02	176.16	18.50	148.00	300	669	26 627
MPCitH & Fiat-Shamir	LowMC	Picnic1 [63]	L1-FS	3	-	-	0.03	0.25	0.03	0.24	32	16	32 862
			L5-FS		-	-	0.15	1.24	0.15	1.17	64	32	128 201
		Picnic3† [68]	L1	4	-	-	0.02	0.08	-	-	17	34	19 573
			L5		-	-	0.03	0.18	-	-	32	64	75 721
	Syndrome Decoding	SDitH [42] hypercube	L1-GF256	256	0.055	0.34	6.73	41.04	8.69	52.98	404	120	8 260
			L1-GF251		0.057	0.35	21.14	130.77	23.40	142.71			
			L3-GF256	256	0.046	0.28	7.74	47.17	10.96	66.85	616	183	19 206
			L3-GF251		0.047	0.29	24.6	149.99	24.60	149.99			
			L5-GF256	256	0.083	0.51	17.48	106.60	24.94	152.10	812	234	33 448
			L5-GF251		0.086	0.53	45.28	276.07	30.11	183.57			
TCitH & Fiat-Shamir	MinRank	Mirath [71]	1b-fast	256	0.016	0.063	0.55	2.18	0.50	1.97	32	57	3 456
			5b-fast		0.061	0.278	1.37	6.29	1.21	6.56	64	112	14 262
Fiat-Shamir with aborts	MLWE MSIS	Dilithium [83]	L2		0.004	0.04	0.03	0.29	0.004	0.05	2 528	1 312	2 420
			L3		0.006	0.06	0.04	0.46	0.006	0.06	4 000	1 952	3 293
			L5		0.009	0.09	0.05	0.51	0.009	0.09	4 864	2 592	4 595

Table 5. Hardware resources comparison of Bittersweet with other MPCitH-, TCitH- and lattice-based signatures. Scheme marked with † include only signature generation.

Scheme		d	FPGA Utilisation				Frequency (MHz)	Platform
			LUT	FF	DSP	BRAM		
Bittersweet	L1	32	15 891	9 242	180	60	125	XC7A200T-3
	L3		19 372	11 541	294	110		
	L5		21 170	12 254	410	144		
	L1	256	18 425	9 617	180	286	125	XC7A200T-3
	L3		22 352	12 171	294	384		
	L5		24 512	13 514	410	512		
Picnic1 [63]	L1-FS	3	90 037	23 105	0	52.5	125	Kintex-7
	L5-FS		167 530	33 164	0	98.5		
Picnic3† [68]	L1	4	82 789	38 554	0	65.5	199	XC7K480T-3
	L5		190 469	65 551	0	159.5		
SDitH-hyp [42]	L1-GF256	256	16 592	8 778	0	164.5	164	XC7A200T-3
	L1-GF251		17 423	16 336	196	164.5		
	L3-GF256	256	22 569	13 881	0	356	164	XC7A200T-3
	L3-GF251		29 961	25 794	382	356		
	L5-GF256	256	23 323	14 962	0	520.5	164	XC7A200T-3
	L5-GF251		34 456	31 409	472	521.5		
Mirath [71]	1b-fast	256	15 400	9 445	0	42	254	XC7A200T-3
	5b-fast		30 415	18 223	0	96	218	
Dilithium [83]	L2							
	L3		29,998	10,366	10	11	97	Artix-7
	L5							

Table 6. Software performance comparison of Bittersweet signatures with other MPCitH-, TCitH-, VOLEitH- and Lattice-based signature schemes for L1 security level. The reported runtimes are from the AVX optimized implementation. Values marked with * have been converted from milliseconds to cycles, as the cycle count was not provided.

Framework	**Assumption**	**Scheme**		d	**Runtime (Mcycles)**			**Size (Bytes)**		
		Name	Parameter set		Keygen	Sign	Verify	$\|sk\|$	$\|pk\|$	$\|\sigma\|$
MPCitH & Fiat-Shamir with aborts	LWR	Bittersweet	L1	32	0.54	13.83	13.04	132	297	8 645
				256	0.54	63.53	65.52	132	297	6 183
				2048	0.54	377.94	373.83	132	297	5 232
MPCitH & Fiat-Shamir	LowMC	Picnic1 [35]	L1-FS	3	0.01	5.19	4.15	32	16	32 862
		Picnic3 [64]	L1	16	-	*18.61	*14.26	32	16	12 595
	Permuted Kernel	PERK [1]	I-fast-5	32	0.09	7.00	4.90	16	240	8 030
			I-short-5	256	0.09	35.00	25.00	16	240	5 780
TCitH & Fiat-Shamir	MinRank	Mirath [2]	1b-fast	256	0.60	15.10	12.20	32	57	3 456
			1b-short	4096	0.60	152.00	101.00	32	57	2 902
	Multivariate Quadratic	MQOM2 [20]	L1-gf2-fast-3r	256	1.90	14.12	11.69	52	72	3 212
			L1-gf2-short-3r	2048	1.74	26.52	25.11	52	72	2 868
	Rank Syndrome Decoding	RYDE [9]	1-Fast (gf2)	256	0.04	11.90	11.50	32	69	3 597
			1-Short (gf2)	4096	0.04	123.70	117.00	32	69	2 988
VOLEitH & Fiat-Shamir	AES	FAEST2 [15]	128-f		0.01	1.72	1.41	32	32	5 924
			128-s		0.01	12.79	9.78	32	32	4 506
	Syndrome Decoding	SDitH2 [73]	L1-gf2-fast		*1.45	*7.25	*6.54	163	70	4 484
			L1-gf2-short		*1.49	*22.86	*20.04	163	70	3 705
Fiat-Shamir with aborts	MLWE MSIS	Dilithium [79]	L2		0.12	0.33	0.12	2 528	1 312	2 420
Hash & Sign	NTRU	Falcon [57]	512		19.87	-	-	1 281	897	666
	MLIP	HAWK2 [32]	512		8.43	0.08	0.15	184	1024	555

Table 7. Software performance comparison of Bittersweet signatures with other MPCitH-, TCitH-, VOLEitH- and Lattice-based signature schemes for L3 security level. The reported runtimes are from the AVX optimized implementation. Values marked with * have been converted from milliseconds to cycles, as the cycle count was not provided.

Framework	**Assumption**	**Scheme**		d	**Runtime (Mcycles)**			**Size (Bytes)**		
		Name	Parameter set		Keygen	Sign	Verify	$\|sk\|$	$\|pk\|$	$\|\sigma\|$
MPCitH & Fiat-Shamir with aborts	LWR	Bittersweet	L3	32	1.74	50.11	47.26	216	483	20 602
				256	1.74	237.78	234.76	216	483	14 622
				2048	1.74	1 464.41	1 456.87	216	483	12 299
MPCitH & Fiat-Shamir	LowMC	Picnic1 [35]	L3-FS	3	0.01	12.14	10.05	48	24	74 134
		Picnic3 [64]	L3	16	-	*37.56	*29.20	48	24	27 551
	Permuted Kernel	PERK [1]	III-fast-5	32	0.19	15.00	11.00	24	370	18 000
			III-short-5	256	0.20	75.00	59.00	24	370	13 200
TCitH & Fiat-Shamir	MinRank	Mirath [2]	3b-fast	256	1.20	55.00	51.60	48	84	7 936
			3b-short	4096	1.20	520.00	327.00	48	84	6 514
	Multivariate Quadratic	MQOM2 [20]	L3-gf2-fast-3r	256	7.85	57.37	51.19	78	108	7 576
			L3-gf2-short-3r	2048	8.23	114.90	103.60	78	108	6 388
	Rank Syndrome Decoding	RYDE [9]	3-Fast	256	0.06	32.10	31.40	48	101	8 264
			3-Short	4096	0.06	368.00	331.10	48	101	6 728
VOLEitH & Fiat-Shamir	AES	FAEST2 [15]	192f		0.01	7.04	6.08	40	48	14 948
			192s		0.01	54.69	42.29	40	48	11 260
	Syndrome Decoding	SDitH2 [73]	L3-gf2-fast		*3.94	*19.26	*17.42	232	98	9 916
			L3-gf2-short		*4.19	*109.12	*101.38	232	98	7 964
Fiat-Shamir with aborts	MLWE MSIS	Dilithium [79]	L3		0.26	0.53	0.18	4 000	1 952	3 293

Table 8. Software performance comparison of Bittersweet signatures with other MPCitH-, TCitH-, VOLEitH- and Lattice-based signature schemes for L5 security level. The reported runtimes are from the AVX optimized implementation. Values marked with * have been converted from milliseconds to cycles, as the cycle count was not provided.

Framework	**Assumption**	**Scheme**		d	**Runtime (Mcycles)**			**Size (Bytes)**		
		Name	Parameter set		Keygen	Sign	Verify	$\|sk\|$	$\|pk\|$	$\|\sigma\|$
MPCitH & Fiat-Shamir with aborts	LWR	Bittersweet	L5	32	3.41	121.19	114.34	300	669	37 648
				256	3.41	570.06	565.34	300	669	26 627
				2048	3.41	3 549.47	3 521.74	300	669	22 333
MPCitH & Fiat-Shamir	LowMC	Picnic1 [35]	L5-FS	3	0.02	21.13	17.71	64	32	128 176
		Picnic3 [64]	L5	16	-	*65.34	*46.87	64	32	48 716
	Permuted Kernel	PERK [1]	V-fast-5	32	0.32	34.00	26.00	32	510	31 700
			V-short-5	256	0.33	168.00	131.00	32	510	23 000
TCitH & Fiat-Shamir	MinRank	Mirath [2]	5b-fast	256	2.00	121.00	88.00	64	112	14 262
			5b-short	4096	2.00	1 421.00	630.00	64	112	11 620
	Multivariate Quadratic	MQOM2 [20]	L5-gf2-fast-3r	256	13.71	145.51	129.97	104	144	13 412
			L5-gf2-short-3r	2048	13.81	278.92	266.51	104	144	11 764
	Rank Syndrome Decoding	RYDE [9]	5-Fast	256	0.08	65.90	61.60	64	133	14 609
			5-Short	4096	0.08	799.30	623.00	64	133	11 819
VOLEitH & Fiat-Shamir	AES	FAEST2 [15]	256f		0.01	11.07	10.24	48	48	26 548
			256s		0.01	76.33	74.55	48	48	20 696
	Syndrome Decoding	SDitH2 [73]	L5-gf2-fast		*4.70	*27.37	*25.06	307	132	17 540
			L5-gf2-short		*4.75	*152.34	*141.02	307	132	14 121
Fiat-Shamir with aborts	MLWE MSIS	Dilithium [79]	L5		0.30	0.64	0.28	4 864	2 592	4 595
Hash & Sign	NTRU	Falcon [57]	512		63.13	-	-	2 305	1 793	1 280
	MLIP	HAWK2 [32]	512		43.66	0.18	0.30	360	2 440	1 221

References

1. Najwa Aaraj, Slim Bettaieb, Loïc Bidoux, Alessandro Budronand, Victor Dyseryn, Andre Esser, Thibauld Feneuil, Philippe Gaborit, Mukul Kulkarni, Victor Mateu, Marco Palumbi, Lucas Perin, Matthieu Rivain, Jean-Pierre Tillich, and Keita Xawaga. PERK. Round 2 : Submission to the NIST Post-Quantum Cryptography : Additional Digital Signature Schemes, 2025. https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-2/spec-files/perk-spec-round2-web.pdf.
2. Gora Adj, Nicolas Aragon, Stefano Barbero, Magali Bardet, Emanuele Bellini, Loïc Bidoux, Jesús-Javier Chi-Domínguez, Victor Dyseryn, Andre Esser, Thibauld Feneuil, Philippe Gaborit, Romaric Neveu, Matthieu Rivain, Luis Rivera-Zamarripa, Carlo Sanna, Jean-Pierre Tillich, Javier Verbel, and Floyd Zweydinger. Mirath Signature Scheme, Algorithm Specifications and Supporting Documentation. Round 2 : Submission to the NIST Post-Quantum Cryptography : Additional Digital Signature Schemes, 2025. https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-2/spec-files/mirath-spec-round2-web.pdf.
3. Gora Adj, Stefano Barbero, Emanuele Bellini, Andre Esser, Luis Rivera-Zamarripa, Carlo Sanna, Javier A. Verbel, and Floyd Zweydinger. MiRitH: Efficient Post-Quantum Signatures from MinRank in the Head. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2024(2):304–328, 2024.
4. Martin R. Albrecht, Benjamin R. Curtis, Amit Deo, Alex Davidson, Rachel Player, Eamonn W. Postlethwaite, Fernando Virdia, and Thomas Wunderer. Estimate all the {LWE, NTRU} schemes! In Dario Catalano and Roberto De Prisco, editors, *Security and Cryptography for Networks - 11th International Conference, SCN 2018, Amalfi, Italy, September 5-7, 2018, Proceedings*, volume 11035 of *Lecture Notes in Computer Science*, pages 351–367. Springer, 2018.
5. Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *J. Math. Cryptol.*, 9(3):169–203, 2015.
6. Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 430–454. Springer, Heidelberg, April 2015.
7. Erdem Alkim, Nina Bindel, Johannes Buchmann, Özgür Dagdelen, Edward Eaton, Gus Gutoski, Juliane Krämer, and Filip Pawlega. Revisiting TESLA in the quantum random oracle model. In *PQCrypto*, volume 10346 of *Lecture Notes in Computer Science*, pages 143–162. Springer, 2017.
8. Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *CCS*, pages 2087–2104. ACM, 2017.
9. Nicolas Aragon, Magali Bardet, Loïc Bidoux, Jesús-Javier Chi-Domínguez, Victor Dyseryn, Thibauld Feneuil, Philippe Gaborit, Antoine Joux, Romaric Neveu, Matthieu Rivain, Jean-Pierre Tillich, and Adrien Vinçotte. RYDE Signature Scheme, Algorithm Specifications and Supporting Documentation. Round 2 : Submission to the NIST Post-Quantum Cryptography : Additional Digital Signature Schemes, 2025. https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-2/spec-files/ryde-spec-round2-web.pdf.
10. Diego F. Aranha, Sebastian Berndt, Thomas Eisenbarth, Okan Seker, Akira Takahashi, Luca Wilke, and Greg Zaverucha. Side-channel protections for picnic signatures. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(4):239–282, 2021.

11. Melissa Azouaoui, Olivier Bronchain, Gaëtan Cassiers, Clément Hoffmann, Yulia Kuzovkova, Joost Renes, Tobias Schneider, Markus Schönauer, François-Xavier Standaert, and Christine van Vredendaal. Protecting Dilithium against Leakage Revisited Sensitivity Analysis and Improved Implementations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(4):58–79, 2023.
12. Shi Bai and Steven D. Galbraith. An improved compression technique for signatures based on learning with errors. In *CT-RSA*, volume 8366 of *Lecture Notes in Computer Science*, pages 28–47. Springer, 2014.
13. Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In *EUROCRYPT*, pages 719–737. Springer, 2012.
14. Manuel Barbosa, Gilles Barthe, Christian Doczkal, Jelle Don, Serge Fehr, Benjamin Grégoire, Yu-Hsuan Huang, Andreas Hülsing, Yi Lee, and Xiaodi Wu. Fixing and mechanizing the security proof of fiat-shamir with aborts and dilithium. In *CRYPTO*, pages 358–389. Springer, 2023.
15. Carsten Baum, Ward Beullens, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß, Christian Majenz, Shibam Mukherjee, Emmanuela Orsini, Sebastian Ramacher, Christian Rechberger, Lawrence Roy, and Peter Scholl. FAEST v2: Algorithm Specifications, Version 2.0. Round 2 : Submission to the NIST Post-Quantum Cryptography : Additional Digital Signature Schemes, 2025. https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-2/spec-files/faest-spec-round2-web.pdf.
16. Carsten Baum, Ward Beullens, Shibam Mukherjee, Emmanuela Orsini, Sebastian Ramacher, Christian Rechberger, Lawrence Roy, and Peter Scholl. One tree to rule them all: Optimizing GGM trees and owfs for post-quantum signatures. In *ASIACRYPT*, pages 463–493. Springer, 2024.
17. Carsten Baum, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß, Emmanuela Orsini, Lawrence Roy, and Peter Scholl. Publicly verifiable zero-knowledge and post-quantum signatures from vole-in-the-head. In *CRYPTO*, pages 581–615. Springer, 2023.
18. Carsten Baum, Cyprien Delpech de Saint Guilhem, Daniel Kales, Emmanuela Orsini, Peter Scholl, and Greg Zaverucha. Banquet: Short and Fast Signatures from AES. In *PKC*, pages 266–297. Springer, 2021.
19. Carsten Baum and Ariel Nof. Concretely-efficient zero-knowledge arguments for arithmetic circuits and their application to lattice-based cryptography. In *PKC*, pages 495–526. Springer, 2020.
20. Ryad Benadjila, Charles Bouillaguet, Thibauld Feneuil, and Matthieu Rivain. MQOM: MQ on my Mind, Algorithm Specifications and Supporting Documentation (Version 2.0). Round 2 : Submission to the NIST Post-Quantum Cryptography : Additional Digital Signature Schemes, 2025. https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-2/spec-files/mqom-spec-round2-web.pdf.
21. Ryad Benadjila, Thibauld Feneuil, and Matthieu Rivain. MQ on my mind: Post-quantum signatures from the non-structured multivariate quadratic problem. In *EuroS&P*, pages 468–485. IEEE, 2024.
22. Rikke Bendlin and Ivan Damgård. Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In *TCC*, pages 201–218. Springer, 2010.
23. Itay Berman, Akshay Degwekar, Ron D. Rothblum, and Prashant Nalini Vasudevan. From laconic zero-knowledge to public-key cryptography - extended abstract. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 674–697. Springer, Heidelberg, August 2018.

24. Slim Bettaieb, Loïc Bidoux, Victor Dyseryn, Andre Esser, Philippe Gaborit, Mukul Kulkarni, and Marco Palumbi. PERK: compact signature scheme based on a new variant of the permuted kernel problem. *Des. Codes Cryptogr.*, 92(8):2131–2157, 2024.
25. Ward Beullens. Sigma protocols for mq, PKP and sis, and fishy signature schemes. In *EUROCRYPT*, pages 183–211. Springer, 2020.
26. Ward Beullens and Cyprien Delpech de Saint Guilhem. LegRoast: Efficient Post-quantum Signatures from the Legendre PRF. In *PQCrypto*, pages 130–150. Springer, 2020.
27. Rishabh Bhadauria, Zhiyong Fang, Carmit Hazay, Muthuramakrishnan Venkitasubramaniam, Tiancheng Xie, and Yupeng Zhang. Ligero++: A new optimized sublinear IOP. In *CCS*, pages 2025–2038. ACM, 2020.
28. Loïc Bidoux, Philippe Gaborit, Mukul Kulkarni, and Víctor Mateu. Code-based Signatures from New Proofs of Knowledge for the Syndrome Decoding Problem. *Des. Codes Cryptogr.*, 91(2):497–544, 2023.
29. Gerrit Bleumer. Threshold Signatures. In *Encyclopedia of Cryptography and Security (2nd Ed.)*, pages 1294–1296. Springer, 2011.
30. Dan Boneh, Yuval Ishai, Alain Passelègue, Amit Sahai, and David J. Wu. Exploring crypto dark matter: - new simple PRF candidates and their applications. In *TCC*, pages 699–729. Springer, 2018.
31. Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic prfs and their applications. In *CRYPTO*, pages 410–428. Springer, 2013.
32. Joppe W. Bos, Olivier Bronchain, Léo Ducas, Serge Fehr, Yu-Hsuan Huang, Thomas Pornin, Eamonn W. Postlethwaite, Thomas Prest, Ludo N. Pulles, and Wessel van Woerden. Hawk, version 1.1. Round 2 : Submission to the NIST Post-Quantum Cryptography : Additional Digital Signature Schemes, 2025. https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-2/spec-files/hawk-spec-round2-web.pdf.
33. Dung Bui, Eliana Carozza, Geoffroy Couteau, Dahmun Goudarzi, and Antoine Joux. Faster signatures from mpc-in-the-head. In *ASIACRYPT*, pages 396–428. Springer, 2024.
34. Eliana Carozza, Geoffroy Couteau, and Antoine Joux. Short signatures from regular syndrome decoding in the head. In *EUROCRYPT*, pages 532–563. Springer, 2023.
35. Melissa Chase, David Derler, Steven Goldfeder, Jonathan Katz, Vladimir Kolesnikov, Claudio Orlandi, Sebastian Ramacher, Chirstian Rechberger, Daniel Slamanig, Xiao Wang, et al. The Picnic Signature Scheme. *The Picnic Signature Scheme - Design Document. Version 2.2, 14 April 2020.*
36. Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-Quantum Zero-Knowledge and Signatures from Symmetric-Key Primitives. In *CCS*, pages 1825–1842. ACM, 2017.
37. David Chaum and Eugène van Heyst. Group Signatures. In *EUROCRYPT*, pages 257–265. Springer, 1991.
38. Keng-Yu Chen and Jiun-Peng Chen. Masking Floating-Point Number Multiplication and Addition of Falcon First- and Higher-order Implementations and Evaluations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2024(2):276–303, 2024.
39. Hongrui Cui, Hanlin Liu, Di Yan, Kang Yang, Yu Yu, and Kaiyi Zhang. sfresolved: Shorter signatures from regular syndrome decoding and vole-in-the-head. In *PKC*, pages 229–258. Springer, 2024.

40. Cyprien Delpech de Saint Guilhem, Lauren De Meyer, Emmanuela Orsini, and Nigel P. Smart. BBQ: Using AES in Picnic Signatures. In *SAC*, pages 669–692. Springer, 2019.
41. Cyprien Delpech de Saint Guilhem, Emmanuela Orsini, and Titouan Tanguy. Limbo: Efficient Zero-Knowledge MPCitH-based Arguments. In *CCS*, pages 3022–3036. ACM, 2021.
42. Sanjay Deshpande, James Howe, Jakub Szefer, and Dongze Yue. Sdith in hardware. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2024(2):215–251, 2024.
43. Julien Devevey, Pouria Fallahpour, Alain Passelègue, and Damien Stehlé. A detailed analysis of fiat-shamir with aborts. In *CRYPTO*, pages 327–357. Springer, 2023.
44. Itai Dinur, Steven Goldfeder, Tzipora Halevi, Yuval Ishai, Mahimna Kelkar, Vivek Sharma, and Greg Zaverucha. Mpc-friendly symmetric cryptography from alternating moduli: Candidates, protocols, and applications. In *CRYPTO*, pages 517–547. Springer, 2021.
45. Itai Dinur and Niv Nadler. Multi-target attacks on the picnic signature scheme and related protocols. In *EUROCRYPT*, pages 699–727. Springer, 2019.
46. Christoph Dobraunig, Daniel Kales, Christian Rechberger, Markus Schofnegger, and Greg Zaverucha. Shorter Signatures Based on Tailor-Made Minimalist Symmetric-Key Crypto. In *CCS*, pages 843–857. ACM, 2022.
47. Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Security of the Fiat-Shamir transformation in the quantum random-oracle model. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 356–383. Springer, Heidelberg, August 2019.
48. Stefan Dziembowski, Sebastian Faust, Gottfried Herold, Anthony Journault, Daniel Masny, and François-Xavier Standaert. Towards sound fresh re-keying with hard (physical) learning problems. In *CRYPTO (2)*, pages 272–301. Springer, 2016.
49. Thomas Espitau, Guilhem Niot, Chao Sun, and Mehdi Tibouchi. Squirrels, Square Unstructured Integer Euclidean Lattice Signature. Round 1 : Submission to the NIST Post-Quantum Cryptography : Additional Digital Signature Schemes, 2023. https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-1/spec-files/Squirrels-spec-web.pdf.
50. Sebastian Faust, Loïc Masure, Elena Michel, Hai Hoang Nguyen, Maximilian Orlt, and François-Xavier Standaert. IP masking with generic security guarantees under minimum assumptions, and applications. In *ASIACRYPT (2)*, volume 16246 of *Lecture Notes in Computer Science*, pages 544–575. Springer, 2025.
51. Serge Fehr. Classical proofs for the quantum collapsing property of classical hash functions. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 315–338. Springer, Heidelberg, November 2018.
52. Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. Syndrome Decoding in the Head: Shorter Signatures from Zero-Knowledge Proofs. In *CRYPTO*, pages 541–572. Springer, 2022.
53. Thibauld Feneuil, Jules Maire, Matthieu Rivain, and Damien Vergnaud. Zero-Knowledge Protocols for the Subset Sum Problem from MPC-in-the-Head with Rejection. In *ASIACRYPT*, pages 371–402. Springer, 2022.
54. Thibauld Feneuil and Matthieu Rivain. Threshold linear secret sharing to the rescue of mpc-in-the-head. In *ASIACRYPT (1)*, pages 441–473. Springer, 2023.
55. Thibauld Feneuil and Matthieu Rivain. Threshold computation in the head: Improved framework for post-quantum signatures and zero-knowledge arguments. *J. Cryptol.*, 38(3):28, 2025.

56. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987.
57. Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon: Fast-fourier lattice-based compact signatures over ntru (specification v1.2). Round-3 : Submission to the NIST Post-Quantum Cryptography Standardization Project, 2020. https://falcon-sign.info/falcon.pdf.
58. Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. ZKBoo: Faster Zero-Knowledge for Boolean Circuits. In *USENIX Security Symposium*, pages 1069–1083. USENIX Association, 2016.
59. Alex B. Grilo, Kathrin Hövelmanns, Andreas Hülsing, and Christian Majenz. Tight adaptive reprogramming in the QROM. In *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part I*, pages 637–667. Springer, 2021.
60. Shay Gueron, Edoardo Persichetti, and Paolo Santini. Designing a Practical Code-Based Signature Scheme from Zero-Knowledge Proofs with Trusted Setup. *Cryptogr.*, 6(1):5, 2022.
61. Carmit Hazay, Muthuramakrishnan Venkitasubramaniam, and Mor Weiss. Beyond mpc-in-the-head: Black-box constructions of short zero-knowledge proofs. In *TCC*, pages 3–33. Springer, 2023.
62. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-Knowledge from Secure Multiparty Computation. In *STOC*, pages 21–30. ACM, 2007.
63. Daniel Kales, Sebastian Ramacher, Christian Rechberger, Roman Walch, and Mario Werner. Efficient FPGA implementations of lowmc and picnic. In Stanislaw Jarecki, editor, *Topics in Cryptology - CT-RSA 2020 - The Cryptographers' Track at the RSA Conference 2020, San Francisco, CA, USA, February 24-28, 2020, Proceedings*, volume 12006 of *Lecture Notes in Computer Science*, pages 417–441. Springer, 2020.
64. Daniel Kales and Greg Zaverucha. Improving the performance of the picnic signature scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(4):154–188, 2020.
65. Matthias Kannwischer, Joost Rijneveld, Peter Schwabe, and Ko Stoffelen. PQM4: Post-Quantum Crypto Library for the ARM Cortex-M4. https://github.com/mupq/pqm4, 2019.
66. Matthias J. Kannwischer, Aymeric Genêt, Denis Butin, Juliane Krämer, and Johannes Buchmann. Differential power analysis of XMSS and SPHINCS. In *COSADE*, pages 168–188. Springer, 2018.
67. Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In *CCS*, pages 525–537. ACM, 2018.
68. Guoxiao Liu, Keting Jia, Puwen Wei, and Lei Ju. High-performance hardware implementation of mpcith and picnic3. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2024(2):190–214, 2024.
69. Vadim Lyubashevsky. Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In *ASIACRYPT*, pages 598–616. Springer, 2009.
70. Vadim Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 738–755. Springer, 2012.

71. Stelios Manasidis, Quinten Norga, Suparna Kundu, and Ingrid Verbauwhede. Mpspeed: Implementing and optimizing mpc-in-the-head digital signatures in hardware. Cryptology ePrint Archive, Paper 2026/206, 2026. https://eprint.iacr.org/2026/206.
72. Loïc Masure, Pierrick Méaux, Thorben Moos, and François-Xavier Standaert. Effective and efficient masking with low noise using small-mersenne-prime ciphers. In *EUROCRYPT (4)*, pages 596–627. Springer, 2023.
73. Carlos Aguilar Melchor, Slim Bettaieb, Loïc Bidoux, Thibauld Feneuil, Nicolas Gama, Shay Gueron, James Howe, Andreas Hülsing, David Joseph, Antoine Joux, Mukul Kulkarni, Edoardo Persichetti, Tovohery Hajatiana Randrianarisoa, Matthieu Rivain, and Dongze Yue. The Syndrome Decoding in the Head (SD-in-the-Head) Signature Scheme, Algorithm Specifications and Supporting Documentation – Version 2.0. Round 2 : Submission to the NIST Post-Quantum Cryptography : Additional Digital Signature Schemes, 2025. https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-2/spec-files/sdith-spec-round2-web.pdf.
74. Carlos Aguilar Melchor, Nicolas Gama, James Howe, Andreas Hülsing, David Joseph, and Dongze Yue. The return of the sdith. In *EUROCRYPT (5)*, volume 14008 of *Lecture Notes in Computer Science*, pages 564–596. Springer, 2023.
75. Arno Mittelbach and Marc Fischlin. *The Theory of Hash Functions and Random Oracles - An Approach to Modern Cryptography*. Information Security and Cryptography. Springer, 2021.
76. Ying Ouyang, Deng Tang, and Yanhong Xu. Code-based zero-knowledge from vole-in-the-head and their applications: Simpler, faster, and smaller. In *ASIACRYPT*, pages 436–470. Springer, 2024.
77. Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to Leak a Secret. In *ASIACRYPT*, pages 552–565. Springer, 2001.
78. Okan Seker, Sebastian Berndt, Luca Wilke, and Thomas Eisenbarth. Sni-in-the-head: Protecting mpc-in-the-head protocols against side-channel analysis. In *CCS*, pages 1033–1049. ACM, 2020.
79. Láo Ducas Shi Bai, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS–dilithium: Algorithm specification and supporting documentation (version 3.1). Round-3 : Submission to the NIST Post-Quantum Cryptography Standardization Project, 2020. https://cryptojedi.org/papers/#dilithiumnistr3.
80. Dominique Unruh. Quantum proofs of knowledge. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 135–152. Springer, Heidelberg, April 2012.
81. Dominique Unruh. Non-interactive zero-knowledge proofs in the quantum random oracle model. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 755–784. Springer, Heidelberg, April 2015.
82. Dominique Unruh. Computationally binding quantum commitments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 497–527. Springer, Heidelberg, May 2016.
83. Cankun Zhao, Neng Zhang, Hanning Wang, Bohan Yang, Wenping Zhu, Zhengdong Li, Min Zhu, Shouyi Yin, Shaojun Wei, and Leibo Liu. A compact and high-performance hardware architecture for crystals-dilithium. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(1):270–295, 2022.

A New Variants of the LWR Problem

The standard LWR assumption is defined for a uniform matrix $\boldsymbol{X}$ and uniform vectors $\boldsymbol{k}$, $\mathbf{U}$, as

$$(\boldsymbol{X}, \lfloor \boldsymbol{X}\boldsymbol{k} \rfloor) \approx (\boldsymbol{X}, \mathbf{U}) .$$

In the Sigma protocol introduced in Section 5, however, some extra information never makes it to the adversary's view thanks to the abort event. Simplifying, the key $\boldsymbol{k} \in \mathbb{Z}_Q^n$ is shared into shares $\boldsymbol{k}_1, \boldsymbol{k}_2 \in \mathbb{Z}_Q^n$,[13] then the LWR on the single keys is computed by $\boldsymbol{y}_i \leftarrow \lfloor \boldsymbol{X}\boldsymbol{k}_i \rfloor$ for $i \in \{1, 2\}$, and finally $\boldsymbol{y}, \boldsymbol{y}_1, \boldsymbol{y}_2$ are output in the first message and one of the key shares is revealed to the adversary in the response message. Assuming that we reveal the first share, the view of the adversary is given by the random variable.

$$(\boldsymbol{X}, \lfloor \boldsymbol{X}\boldsymbol{k} \rfloor , \lfloor \boldsymbol{X}\boldsymbol{k}_2 \rfloor , \boldsymbol{k}_1) .$$

Notice that, since both $\boldsymbol{X}$ and $\boldsymbol{k}_1$ are given to the adversary, we can just simplify and omit $\lfloor \boldsymbol{X}\boldsymbol{k}_1 \rfloor$ from the view. This process is repeated several times, and the end result is that the adversary's view is

$$\left(\boldsymbol{X}, \lfloor \boldsymbol{X}\boldsymbol{k} \rfloor , \left(\left\lfloor \boldsymbol{X}\boldsymbol{k}_2^{(j)} \right\rfloor , \boldsymbol{k}_1^{(j)} \right)_{j \in [N]} \right) , \tag{5}$$

for some N that depends on the number of iterations in the Sigma protocol and the number of signatures. In principle, this is bad because now the adversary gets many samples of the error between $\lfloor \boldsymbol{X}\boldsymbol{k} \rfloor$ and $\lfloor \boldsymbol{X}\boldsymbol{k}_1 \rfloor + \lfloor \boldsymbol{X}\boldsymbol{k}_2 \rfloor$, along with the share $\boldsymbol{k}_1$ in full, which allows for averaging attacks. We define this new problem in Section A.1, and we define a useful abstraction in Section A.2 that allows us to deal with the problem more easily.

A.1 Learning with Rounding with Misuse

Here, we formally introduce the problem, which we call *Learning with Rounding with Misuse*, or *LWRwM* for short. The name comes from reusing the same matrix $\boldsymbol{X}$ to compute different instances of LWR, thus allowing averaging attacks on the secret. We describe the LWRwM distributions in Fig. 6, and we define the LWRwM assumption as

$$\mathsf{LWRwM.Real}(\boldsymbol{k}) \approx \mathsf{LWRwM.Rand}().$$

Notice that the definition in Fig. 6 is slightly different from what we stated at the beginning of the section; namely, instead of giving $\boldsymbol{y}_2$ to the adversary, we give $\boldsymbol{u} - \boldsymbol{r}_2$, where $\boldsymbol{r}_2 = (\boldsymbol{v} \mod P)$. However, by writing

$$\boldsymbol{u} = \frac{Q}{P}\boldsymbol{y} + \boldsymbol{r} \qquad\qquad (= \boldsymbol{X}\boldsymbol{k} \text{ in the } \mathsf{Real} \text{ case}) ,$$

[13] In our protocol, we sample d shares instead of just two, but it is easy to see that this can be derived by sharing $\boldsymbol{k}$ into $\boldsymbol{k}_1, \boldsymbol{k}_2$ and then sharing again $\boldsymbol{k}_1$ into $\boldsymbol{k}_{1,1}, \ldots, \boldsymbol{k}_{1,d-1}$; additionally, we generate most of the shares using a PRG, which we omit in this section for simplicity.

$\mathsf{LWRwM}(Q,P,n,m,N).\mathsf{Real}(\boldsymbol{k})$	$\mathsf{LWRwM}(Q,P,n,m,N).\mathsf{Rand}()$
$\boldsymbol{X} \leftarrow\!\!\$\ \mathbb{Z}_Q^{m\times n}$, $\boldsymbol{u} \leftarrow \boldsymbol{X}\boldsymbol{k}$	$\boldsymbol{X} \leftarrow\!\!\$\ \mathbb{Z}_Q^{m\times n}$, $\boldsymbol{u} \leftarrow\!\!\$\ \mathbb{Z}_Q^{m}$
$\boldsymbol{y} \leftarrow \lfloor \boldsymbol{u} \rfloor_P$	$\boldsymbol{y} \leftarrow \lfloor \boldsymbol{u} \rfloor_P$
for $i \leftarrow [N]$ **do**	**for** $i \leftarrow [N]$ **do**
$\boldsymbol{k}_1 \leftarrow\!\!\$\ \mathbb{Z}_Q^n$	$\boldsymbol{k}_1 \leftarrow\!\!\$\ \mathbb{Z}_Q^n$
$\boldsymbol{v} \leftarrow \boldsymbol{u} - \boldsymbol{X}\boldsymbol{k}_1$	$\boldsymbol{v} \leftarrow \boldsymbol{u} - \boldsymbol{X}\boldsymbol{k}_1$
$\boldsymbol{r}_2 \leftarrow \boldsymbol{v} - \frac{Q}{P}\lfloor \boldsymbol{v} \rfloor_P$	$\boldsymbol{r}_2 \leftarrow \boldsymbol{v} - \frac{Q}{P}\lfloor \boldsymbol{v} \rfloor_P$
$s^{(i)} \leftarrow (\boldsymbol{u} - \boldsymbol{r}_2, \boldsymbol{k}_1)$	$s^{(i)} \leftarrow (\boldsymbol{u} - \boldsymbol{r}_2, \boldsymbol{k}_1)$
return $\left(\boldsymbol{X}, \boldsymbol{y}, \left(s^{(i)}\right)_{i\in[N]}\right)$	**return** $\left(\boldsymbol{X}, \boldsymbol{y}, \left(s^{(i)}\right)_{i\in[N]}\right)$

Fig. 6. The real and random distributions of LWRwM, parametrized by the usual LWR parameters and, additionally, the number N of samples.

$$\begin{aligned} \boldsymbol{X}\boldsymbol{k}_1 &= \frac{Q}{P}\boldsymbol{y}_1 + \boldsymbol{r}_1, \\ \boldsymbol{v} = \boldsymbol{u} - \boldsymbol{X}\boldsymbol{k}_1 &= \frac{Q}{P}\boldsymbol{y}_2 + \boldsymbol{r}_2 \qquad (= \boldsymbol{X}\boldsymbol{k}_2 \text{ in the } \mathsf{Real} \text{ case}), \end{aligned}$$

with $\boldsymbol{r}, \boldsymbol{r}_1, \boldsymbol{r}_2 \in \mathbb{Z}_{Q/P}^m$, we can write the output $\boldsymbol{u} - \boldsymbol{r}_2$ in the LWRwM distribution as

$$\begin{aligned} \boldsymbol{u} - \boldsymbol{r}_2 &= \boldsymbol{u} - \boldsymbol{v} + \frac{Q}{P}\lfloor \boldsymbol{v} \rfloor_P \\ &= \boldsymbol{u} - \boldsymbol{u} + \boldsymbol{X}\boldsymbol{k}_1 + \frac{Q}{P}\lfloor \boldsymbol{u} - \boldsymbol{X}\boldsymbol{k}_1 \rfloor_P \\ &= \boldsymbol{X}\boldsymbol{k}_1 + \frac{Q}{P}\lfloor \boldsymbol{u} - \boldsymbol{X}\boldsymbol{k}_1 \rfloor_P . \end{aligned}$$

When the sample comes from the real distribution, $\boldsymbol{u} = \boldsymbol{X}\boldsymbol{k}$, and

$$\lfloor \boldsymbol{X}\boldsymbol{k} - \boldsymbol{X}\boldsymbol{k}_1 \rfloor_P = \lfloor \boldsymbol{X}\boldsymbol{k}_2 \rfloor_P ,$$

which is exactly the value appearing in Eq. (5). On the other side, if $\boldsymbol{u}$ is random, there is no $\boldsymbol{k}_2$, and the value $\lfloor \boldsymbol{X}\boldsymbol{k}_2 \rfloor_P$ must be simulated by $\lfloor \boldsymbol{u} - \boldsymbol{X}\boldsymbol{k}_1 \rfloor_P$, that is, again, what we get in the above derivation.

The LWRwM problem is useful to understand why the adversary gets more information. In particular, by writing the extra information as $\boldsymbol{u} - \boldsymbol{r}_2$, we can see that the adversary gets the unrounded LWR value $\boldsymbol{u}$, which can be either $\boldsymbol{X}\boldsymbol{k}$ or uniform, with many samples of the error $\boldsymbol{r}_2$, which can be used to average the distribution and leak information from $\boldsymbol{u}$.

A.2 Learning with Rounding and Carry Leakage

$\mathsf{LWRCL}(Q,P,n,m,N).\mathsf{Real}(\boldsymbol{k})$	$\mathsf{LWRCL}(Q,P,n,m,N).\mathsf{Rand}()$
$\boldsymbol{X} \leftarrow\!\!\$\ \mathbb{Z}_Q^{m\times n}$, $\boldsymbol{u} \leftarrow \boldsymbol{X}\boldsymbol{k}$	$\boldsymbol{X} \leftarrow\!\!\$\ \mathbb{Z}_Q^{m\times n}$, $\boldsymbol{u} \leftarrow\!\!\$\ \mathbb{Z}_Q^{m}$
for $i \leftarrow [N]$ **do**	**for** $i \leftarrow [N]$ **do**
$\quad \boldsymbol{k}_1 \leftarrow\!\!\$\ \mathbb{Z}_Q^n$	$\quad \boldsymbol{k}_1 \leftarrow\!\!\$\ \mathbb{Z}_Q^n$
$\quad \boldsymbol{y}_1 \leftarrow \lfloor \boldsymbol{X}\boldsymbol{k}_1 \rfloor_P$	$\quad \boldsymbol{y}_1 \leftarrow \lfloor \boldsymbol{X}\boldsymbol{k}_1 \rfloor_P$
$\quad \boldsymbol{y}_2 \leftarrow \lfloor \boldsymbol{u} - \boldsymbol{X}\boldsymbol{k}_1 \rfloor_P$	$\quad \boldsymbol{y}_2 \leftarrow \lfloor \boldsymbol{u} - \boldsymbol{X}\boldsymbol{k}_1 \rfloor_P$
$\quad \boldsymbol{c} \leftarrow \lfloor \boldsymbol{u} \rfloor_P - \boldsymbol{y}_1 - \boldsymbol{y}_2$	$\quad \boldsymbol{c} \leftarrow \lfloor \boldsymbol{u} \rfloor_P - \boldsymbol{y}_1 - \boldsymbol{y}_2$
$\quad s^{(i)} \leftarrow (\boldsymbol{k}_1, \boldsymbol{c})$	$\quad s^{(i)} \leftarrow (\boldsymbol{k}_1, \boldsymbol{c})$
return $\left(\boldsymbol{X}, \lfloor \boldsymbol{u} \rfloor_P, \left(s^{(i)}\right)_{i\in[N]}\right)$	**return** $\left(\boldsymbol{X}, \lfloor \boldsymbol{u} \rfloor_P, \left(s^{(i)}\right)_{i\in[N]}\right)$

Fig. 7. The real and random distributions of LWRCL, parametrized by the usual LWR parameters and, additionally, the number N of samples.

In Fig. 7 introduce another variant of LWR, which we name *learning with rounding and carry leakage* (LWRCL). In this setting, instead of getting the $\lfloor \boldsymbol{X}\boldsymbol{k}_2 \rfloor_P$, the adversary learns the error between the original LWR value and the sum of the rounded shares:

$$\boldsymbol{c} = \lfloor \boldsymbol{X}\boldsymbol{k} \rfloor_P - \lfloor \boldsymbol{X}\boldsymbol{k}_1 \rfloor_P - \lfloor \boldsymbol{X}\boldsymbol{k}_2 \rfloor_P .$$

Transformed for the general case, the carry can be defined as

$$\boldsymbol{c} = \lfloor \boldsymbol{u} \rfloor_P - \lfloor \boldsymbol{X}\boldsymbol{k}_1 \rfloor_P - \lfloor \boldsymbol{v} \rfloor_P .$$

The name *carry leakage* comes from the fact that, for $Q = 2^q$ and $P = 2^p$, the value $\boldsymbol{c}$ is the carry of the sum $\boldsymbol{r}_1 + \boldsymbol{r}_2$.

As before, we define the LWRCL assumption as

$$\mathsf{LWRCL.Real}(\boldsymbol{k}) \approx \mathsf{LWR.Rand}().$$

We also show that the two problems are equivalent by constructing an LWRCL sample from an LWRwM sample and vice versa. Recall that, from the previous section,

$$\boldsymbol{r}_2 = (\boldsymbol{v} \mod P) = \boldsymbol{v} - \frac{Q}{P} \lfloor \boldsymbol{v} \rfloor_P = \boldsymbol{v} - \frac{Q}{P}\boldsymbol{y}_2,$$

and, therefore,

$$\begin{aligned} \lfloor \boldsymbol{u} \rfloor_P &= \lfloor \boldsymbol{X}\boldsymbol{k}_1 + \boldsymbol{v} \rfloor_P \\ &= \left\lfloor \frac{Q}{P}\boldsymbol{y}_1 + \boldsymbol{r}_1 + \frac{Q}{P}\boldsymbol{y}_2 + \boldsymbol{r}_2 \right\rfloor_P \end{aligned}$$

$$= \boldsymbol{y}_1 + \boldsymbol{y}_2 + \lfloor \boldsymbol{r}_1 + \boldsymbol{r}_2 \rfloor_P ,$$

therefore, by plugging in $\boldsymbol{y}_1 = \lfloor \boldsymbol{X}\boldsymbol{k}_1 \rfloor_P$ and $\boldsymbol{y}_2 = \lfloor \boldsymbol{X}\boldsymbol{k}_2 \rfloor_P$, we get that

$$\boldsymbol{c} = \lfloor \boldsymbol{r}_1 + \boldsymbol{r}_2 \rfloor_P .$$

Getting carry leakage from misuse sampling. Given an instance of LWRwM, the adversary obtains $\boldsymbol{k}_1$ and $\boldsymbol{u} - \boldsymbol{r}_2$, and can compute

$$\begin{aligned}(\boldsymbol{u} - \boldsymbol{r}_2) - \boldsymbol{r}_1 &= (\boldsymbol{X}\boldsymbol{k}_1 + \boldsymbol{v} - \boldsymbol{r}_2) - \boldsymbol{r}_1 \\ &= \frac{Q}{P}(\boldsymbol{y}_1 + \boldsymbol{y}_2),\end{aligned}$$

hence the carry can be obtained by

$$\begin{aligned}\boldsymbol{c} &= \lfloor \boldsymbol{u} \rfloor_P - \lfloor \boldsymbol{X}\boldsymbol{k}_1 \rfloor_P - \lfloor \boldsymbol{v} \rfloor_P \\ &= \lfloor \boldsymbol{u} \rfloor_P - (\boldsymbol{y}_1 + \boldsymbol{y}_2) \\ &= \lfloor \boldsymbol{u} \rfloor_P - \lfloor (\boldsymbol{u} - \boldsymbol{r}_2) - \boldsymbol{r}_1 \rfloor_P .\end{aligned}$$

Getting misuse sampling from carry leakage. Given an instance of LWRCL, the adversary obtains $\boldsymbol{k}_1$ and the carry $\boldsymbol{c}$, and can compute

$$\begin{aligned}(\boldsymbol{u} - \boldsymbol{r}_2) &= (\boldsymbol{X}\boldsymbol{k}_1 + \boldsymbol{v} - \boldsymbol{r}_2) \\ &= \frac{Q}{P}\boldsymbol{y}_1 + \boldsymbol{r}_1 + \frac{Q}{P}\boldsymbol{y}_2 + \boldsymbol{r}_2 - \boldsymbol{r}_2 \\ &= \frac{Q}{P}(\boldsymbol{y}_1 + \boldsymbol{y}_2 + \boldsymbol{c} - \boldsymbol{c}) + \boldsymbol{r}_1 \\ &= \frac{Q}{P}(\lfloor \boldsymbol{u} \rfloor_P - \boldsymbol{c}) + \boldsymbol{r}_1,\end{aligned}$$

thus getting back the misuse sampling.

A.3 Reducing LWRCL to LWR

In this subsection, we show that, for $Q = 2^q$ and $P = 2^p$, we can easily reduce the LWRCL problem to the LWR problem, provided that we have α extra bits to use for the reduction. The theorem below proves exactly this fact by adding a distinguishing error of $2^{1-\alpha}Nm + \epsilon_{\mathsf{LWR}}$ caused by the fact that, intuitively, one of the shares $\boldsymbol{k}_1$ given to the adversary is "too close" to the original key $\boldsymbol{k}$ after the matrix multiplication.

Theorem 11. *Let $q, p, n, m, N, \alpha \in \mathbb{N}$ such that $q > p$ and $2^\alpha > Nm$. Assume that the decisional assumption*

$$\mathsf{LWR}(2^q, 2^{p+\alpha}, n, m)$$

holds with advantage ϵ_{LWR}, for some $\epsilon_{\mathsf{LWR}} \in [0, 1]$. Then, the decisional assumption

$$\mathsf{LWRCL}(2^q, 2^p, n, m, N)$$

holds with advantage

$$\epsilon_{\mathsf{LWRCL}} \coloneqq 2 \cdot \left(2^{-\alpha} Nm + \epsilon_{\mathsf{LWR}}\right).$$

Looking ahead, the check of being "too close" is performed by the reduction using the α additional bits given in the sample. For this reason, the original LWR instance must be secure even with parameters $(2^q, 2^{p+\alpha}, n, m)$. Unfortunately, this means that α should be quite large in order to achieve a negligible advantage. Interestingly, if we limit the adversary to only see LWRCL samples that are not "too close" to the original LWR sample (i.e., in which the α bits are different), α does not need to be large anymore. This strategy is what we use in our Sigma protocol in Section 5, where the protocol aborts if the α bits of the shares match the ones in the original LWR sample.

$\mathcal{R}\left(\boldsymbol{X}, \boldsymbol{y}\right)$ where $\boldsymbol{X} \in \mathbb{Z}_{2^q}^{m \times n}, \boldsymbol{y} \in \mathbb{Z}_{2^{p+\alpha}}^{m}$

1 : $r_{\mathsf{S}} \leftarrow q - p - \alpha,\ r_{\mathsf{E}} \leftarrow r_{\mathsf{S}} + \alpha$
2 : **for** $i \leftarrow 1..N$ **do**
3 : $\quad \boldsymbol{k}_1 \leftarrow\!\!\$\ \mathbb{Z}_{2^q}^{n}$
4 : $\quad$ **for** $j \leftarrow 1..m$ **do**
$\quad\quad$ // Get the j-th component of the vector $\boldsymbol{X}\boldsymbol{k}_1 \in \mathbb{Z}_{2^q}^{m}$.
5 : $\quad\quad z \leftarrow \left(\boldsymbol{X}\boldsymbol{k}_1\right)_j$
$\quad\quad$ // Compare z with the challenge $\boldsymbol{y}$ and set the carry accordingly.
6 : $\quad\quad$ **if** $z[r_{\mathsf{S}}..r_{\mathsf{E}}] < y_j[0..\alpha]$ **then** $c_j \leftarrow 0$
7 : $\quad\quad$ **elseif** $z[r_{\mathsf{S}}..r_{\mathsf{E}}] > y_j[0..\alpha]$ **then** $c_j \leftarrow 1$
$\quad\quad$ // If a strict ordering is not possible, the carry is determined by one of the
$\quad\quad$ // least significant bits of $\boldsymbol{X}\boldsymbol{k}$, hence $\mathcal{R}$ has not enough information.
8 : $\quad\quad$ **else abort**
9 : $\quad \boldsymbol{c} \coloneqq (c_1, \ldots, c_m)$
10 : $\quad s^{(i)} \leftarrow (\boldsymbol{k}_1, \boldsymbol{c})$
11 : **return** $\left(\boldsymbol{X}, \lfloor \boldsymbol{y} \rfloor_{2^p}, \left(s^{(i)}\right)_{i \in [N]}\right)$

Fig. 8. The reduction from LWRCL to LWR

The rest of this section is dedicated to the proof of the above theorem. We proceed by reduction. Namely, consider the reduction $\mathcal{R}$ depicted in Fig. 8.

Let $\textsc{Bad}_j^{(i)}$ be the event where $z[r_{\mathsf{S}}..r_{\mathsf{E}}] = y_j[0..\alpha]$ in iteration i, and let

$$\textsc{Bad} \coloneqq \bigcup_{i \in [N]} \bigcup_{j \in n} \textsc{Bad}_j^{(i)}.$$

In particular, $\textsc{Bad}$ happens in the reduction if and only if $\mathcal{R}$ aborts.

Lemma 4. *Let $q, p, n, m, N, \alpha \in \mathbb{N}$ such that $q > p$ and $2^\alpha > Nm$. Assume that* Bad *does not happen. Then, the reduction $\mathcal{R}$ is perfect.*

Proof. The only difference between the output of LWRCL and the output of the reduction $\mathcal{R}$ lies in how the carry vector is computed. Assuming that the two ways to compute the carry lead to the same result, it is easy to see that the reduction perfectly simulates LWRCL.Real when receiving a sample from LWR.Real as input and perfectly simulates LWRCL.Rand when receiving a sample from LWR.Rand as input. Therefore, the rest of the proof will focus on the carry computation.

In the original LWRCL experiments, the carry is computed as

$$\boldsymbol{c} \leftarrow \lfloor \boldsymbol{u} \rfloor_{2^p} - \lfloor \boldsymbol{X}\boldsymbol{k}_1 \rfloor_{2^p} - \lfloor \boldsymbol{u} - \boldsymbol{X}\boldsymbol{k}_1 \rfloor_{2^p} ,$$

where $\boldsymbol{u}$ is either the real computation $\boldsymbol{X}\boldsymbol{k}$ or a uniformly sampled vector in $\mathbb{Z}_{2^q}^m$. By letting $\boldsymbol{z} \coloneqq \boldsymbol{X}\boldsymbol{k}_1$, this means that, for every component $j \in [m]$ of the vectors, the carry is computed as

$$c_j \leftarrow \lfloor u_j \rfloor_{2^p} - \lfloor z_j \rfloor_{2^p} - \lfloor u_j - z_j \rfloor_{2^p} .$$

We can now multiply by 2^{q-p} and apply Proposition 2 to get

$$\begin{aligned} 2^{q-p}c_j = u_j &- u_j[0..(q-p)] \\ &- z_j + z_j[0..(q-p)] \\ &- (u_j - z_j) + (u_j - z_j)[0..(q-p)] \\ = (u_j - z_j)&[0..(q-p)] + z_j[0..(q-p)] - u_j[0..(q-p)]. \end{aligned}$$

Notice that all the values on the right-hand side of the equality are $(q-p)$-bits long, hence they are all strictly less than 2^{q-p}. This means that

$$-2^{q-p} < 2^{q-p}c_j < 2 \cdot 2^{q-p}.$$

Since c_j is an integer, the only two possible values for it are either $c_j = 0$ or $c_j = 1$. We have:

- if $z_j[0..(q-p)] \leq u_j[0..(q-p)]$,

  $$2^{q-p}c_j \leq (u_j - z_j)[0..(q-p)] < 2^{q-p},$$

  which means that c_j cannot be 1, hence $c_j = 0$;
- if $z_j[0..(q-p)] > u_j[0..(q-p)]$,

  $$2^{q-p}c_j > (u_j - z_j)[0..(q-p)] \geq 0,$$

  which means that c_j cannot be 0, hence $c_j = 1$.

To conclude, it suffices to check that the reduction does the same. First of all, notice that $y_j = \lfloor u_j \rfloor_{2^{p+\alpha}}$ or, by applying Proposition 2 and recalling the definition of r_{S} and r_{E},

$$y_j = u_j[(q-p-\alpha)..q] = u_j[r_{\mathsf{S}}..q].$$

Then, by applying again Proposition 2 and $r_{\mathsf{E}} = r_{\mathsf{S}} + \alpha$, we get that

$$y_j[0..\alpha] = u_j[r_{\mathsf{S}}..q][0..\alpha] = u_j[r_{\mathsf{S}}..(r_{\mathsf{S}} + \alpha)] = u_j[r_{\mathsf{S}}..r_{\mathsf{E}}].$$

Now, assuming that BAD does not happen, there are only two possibilities for the comparison between $z[r_{\mathsf{S}}..r_{\mathsf{E}}]$ and $y_j[0..\alpha] = u_j[r_{\mathsf{S}}..r_{\mathsf{E}}]$

- If $z[r_{\mathsf{S}}..r_{\mathsf{E}}] < u_j[r_{\mathsf{S}}..r_{\mathsf{E}}]$, by Proposition 2, $z[0..r_{\mathsf{E}}] < u_j[0..r_{\mathsf{E}}]$, and both $\mathcal{R}$ and the original LWRCL experiments set $c_j = 0$.
- If $z[r_{\mathsf{S}}..r_{\mathsf{E}}] > u_j[r_{\mathsf{S}}..r_{\mathsf{E}}]$, by Proposition 2, $z[0..r_{\mathsf{E}}] > u_j[0..r_{\mathsf{E}}]$, and both $\mathcal{R}$ and the original LWRCL experiments set $c_j = 1$.

This concludes the proof.

Lemma 5. *Let $q, p, n, m, N, \alpha \in \mathbb{N}$ such that $q > p$ and $2^\alpha > Nm$. Then,*

$$\mathbb{P}\left[\text{BAD } \textit{happens} \mid (\boldsymbol{X}, \boldsymbol{y}) \leftarrow\!\!\$\ \mathsf{LWR.Rand}()\right] \leq 2^{-\alpha} Nm.$$

Proof. We start by bounding the probability of the event $\text{BAD}_j^{(i)}$; then, the proof will follow by the union bound. The event $\text{BAD}_j^{(i)}$ happens if and only if, when sampling z in line 5 of the reduction, it happens that $z[r_{\mathsf{S}}..r_{\mathsf{E}}] = y_j[0..\alpha]$. However, when the reduction receives $(\boldsymbol{X}, \boldsymbol{y})$ from $\mathsf{LWR.Rand}$, the value y_j is uniformly sampled, and the value z is sampled independently of it. This means that

$$\begin{aligned}
&\mathbb{P}\left[\text{BAD}_j^{(i)} \text{ happens} \mid (\boldsymbol{X}, \boldsymbol{y}) \leftarrow\!\!\$\ \mathsf{LWR.Rand}()\right] \\
&= \mathbb{P}\left[z[r_{\mathsf{S}}..r_{\mathsf{E}}] = y_j[0..\alpha] \mid (\boldsymbol{X}, \boldsymbol{y}) \leftarrow\!\!\$\ \mathsf{LWR.Rand}()\right] \\
&= \mathbb{P}\left[z[r_{\mathsf{S}}..r_{\mathsf{E}}] = \mathbf{U}_{\mathbb{Z}_{2^\alpha}}\right] \\
&= 2^{-\alpha}.
\end{aligned}$$

Since

$$\text{BAD} = \bigcup_{i \in [N]} \bigcup_{j \in n} \text{BAD}_j^{(i)},$$

the proof follows by the union bound. □

Lemma 6. *Let $q, p, n, m, N, \alpha \in \mathbb{N}$ such that $q > p$ and $2^\alpha > Nm$. Then, for a uniformly sampled $\boldsymbol{k}$,*

$$\mathbb{P}\left[\text{BAD } \textit{happens} \mid (\boldsymbol{X}, \boldsymbol{y}) \leftarrow\!\!\$\ \mathsf{LWR.Real}(\boldsymbol{k})\right] \leq 2^{-\alpha} Nm + \epsilon_{\mathsf{LWR}}.$$

Proof. By reduction to the $\mathsf{LWR}(2^q, 2^{p+\alpha}, n, m)$ problem. Assume that BAD happens with probability more than $2^{-\alpha} Nm + \epsilon_{\mathsf{LWR}}$ when the sample comes from the real distribution, and consider the following distinguisher.

$\mathcal{D}(\boldsymbol{X}, \boldsymbol{y})$ where $\boldsymbol{X} \in \mathbb{Z}_{2^q}^{m \times n}, \boldsymbol{y} \in \mathbb{Z}_{2^{p+\alpha}}^m$

```
1 : run R(X, y)
2 : if R aborts then
3 :     return 1
4 : else
5 :     return 0
```

Recall that, by hypothesis,

$$\mathbb{P}\left[\textsc{Bad} \mid (\boldsymbol{X}, \boldsymbol{y}) \leftarrow\!\!\$\ \mathsf{LWR.Real}(\boldsymbol{k})\right] > 2^{-\alpha} Nm + \epsilon_{\mathsf{LWR}},$$

and, by Lemma 5,

$$2^{-\alpha} Nm \geq \mathbb{P}\left[\textsc{Bad} \mid (\boldsymbol{X}, \boldsymbol{y}) \leftarrow\!\!\$\ \mathsf{LWR.Rand}()\right].$$

Therefore, the difference

$$\mathbb{P}\left[\textsc{Bad} \mid (\boldsymbol{X}, \boldsymbol{y}) \leftarrow\!\!\$\ \mathsf{LWR.Real}(\boldsymbol{k})\right] - \mathbb{P}\left[\textsc{Bad} \mid (\boldsymbol{X}, \boldsymbol{y}) \leftarrow\!\!\$\ \mathsf{LWR.Rand}()\right]$$

is positive and we can drop the absolute value. Furthermore, by definition of $\mathcal{D}$,

$$\mathbb{P}\left[\mathcal{D}(\boldsymbol{X}, \boldsymbol{y}) = 1 \mid (\boldsymbol{X}, \boldsymbol{y}) \leftarrow\!\!\$\ \mathsf{Real}(\boldsymbol{k})\right] = \mathbb{P}\left[\textsc{Bad} \mid (\boldsymbol{X}, \boldsymbol{y}) \leftarrow\!\!\$\ \mathsf{LWR.Real}(\boldsymbol{k})\right]$$

and

$$\mathbb{P}\left[\mathcal{D}(\boldsymbol{X}, \boldsymbol{y}) = 1 \mid (\boldsymbol{X}, \boldsymbol{y}) \leftarrow\!\!\$\ \mathsf{Rand}()\right] = \mathbb{P}\left[\textsc{Bad} \mid (\boldsymbol{X}, \boldsymbol{y}) \leftarrow\!\!\$\ \mathsf{LWR.Rand}()\right].$$

With these considerations, we can compute the distinguishing advantage of $\mathcal{D}$ as

$$\begin{aligned}
&\left|\mathbb{P}\left[\mathcal{D}(\boldsymbol{X}, \boldsymbol{y}) = 1 \mid (\boldsymbol{X}, \boldsymbol{y}) \leftarrow\!\!\$\ \mathsf{Real}(\boldsymbol{k})\right]\right. \\
&\quad \left.-\mathbb{P}\left[\mathcal{D}(\boldsymbol{X}, \boldsymbol{y}) = 1 \mid (\boldsymbol{X}, \boldsymbol{y}) \leftarrow\!\!\$\ \mathsf{Rand}()\right]\right| \\
&= \left|\mathbb{P}\left[\textsc{Bad} \mid (\boldsymbol{X}, \boldsymbol{y}) \leftarrow\!\!\$\ \mathsf{Real}(\boldsymbol{k})\right]\right. && \text{(by definition of } \mathcal{D}) \\
&\quad \left.-\mathbb{P}\left[\textsc{Bad} \mid (\boldsymbol{X}, \boldsymbol{y}) \leftarrow\!\!\$\ \mathsf{Rand}()\right]\right| \\
&= \mathbb{P}\left[\textsc{Bad} \mid (\boldsymbol{X}, \boldsymbol{y}) \leftarrow\!\!\$\ \mathsf{Real}(\boldsymbol{k})\right] && \text{(by positivity of the difference)} \\
&\quad - \mathbb{P}\left[\textsc{Bad} \mid (\boldsymbol{X}, \boldsymbol{y}) \leftarrow\!\!\$\ \mathsf{Rand}()\right] \\
&> 2^{-\alpha} Nm + \epsilon_{\mathsf{LWR}} - 2^{-\alpha} Nm && \text{(by hypothesis and Lemma 5)} \\
&= \epsilon_{\mathsf{LWR}}.
\end{aligned}$$

This means that $\mathcal{D}$ is a distinguisher for the $\mathsf{LWR}(2^q, 2^{p+\alpha}, n, m)$ with advantage more than ϵ_{LWR}, which is against the assumption of hardness of the decisional LWR problem. □

We are now ready to prove Theorem 11.

Proof. The proof proceeds by a reduction $\mathcal{R}$ to the $\mathsf{LWR}(2^q, 2^{p+\alpha}, n, m)$ problem. Assume that there exists a distinguisher $\mathcal{D}$ such that

$$\left|\mathbb{P}\left[\mathcal{D}\left(\mathsf{LWRCL.Real}(\boldsymbol{k})\right) = 1\right] - \mathbb{P}\left[\mathcal{D}\left(\mathsf{LWRCL.Rand}()\right) = 1\right]\right| > \epsilon_{\mathsf{LWRCL}}.$$

We show that $\mathcal{D} \circ \mathcal{R}$ is a distinguisher for the LWR problem with an advantage of more than ϵ_{LWR}. We start by observing that, since the only difference between the computation of $\mathcal{D}\left(\mathcal{R}\left(\mathsf{LWR.Real}(\boldsymbol{k})\right)\right)$ and $\mathcal{D}\left(\mathsf{LWRCL.Real}(\boldsymbol{k})\right)$ is that $\mathcal{D} \circ \mathcal{R}$ aborts when Bad happens,

$$\left|\mathbb{P}\left[\mathcal{D}\left(\mathcal{R}\left(\mathsf{LWR.Real}(\boldsymbol{k})\right)\right) = 1\right] - \mathbb{P}\left[\mathcal{D}\left(\mathsf{LWRCL.Real}(\boldsymbol{k})\right) = 1\right]\right|$$

$$\begin{aligned}
&\leq \mathbb{P}\left[\textsc{Bad}\right] \\
&\leq 2^{-\alpha} Nm + \epsilon_{\mathsf{LWR}},
\end{aligned}$$

where the last inequality is simply the result of Lemma 6. Similarly, the event $\mathcal{R}$ is the only difference between $\mathcal{D}\left(\mathcal{R}\left(\mathsf{LWR.Rand}()\right)\right)$ and $\mathcal{D}\left(\mathsf{LWRCL.Rand}()\right)$ as well, therefore

$$\begin{aligned}
&\left|\mathbb{P}\left[\mathcal{D}\left(\mathcal{R}\left(\mathsf{LWR.Rand}()\right)\right) = 1\right] - \mathbb{P}\left[\mathcal{D}\left(\mathsf{LWRCL.Rand}()\right) = 1\right]\right| \\
&\leq \mathbb{P}\left[\textsc{Bad}\right] \\
&\leq 2^{-\alpha} Nm,
\end{aligned}$$

where the last inequality is simply the result of Lemma 5. From the above observation, and the assumption that $\mathcal{D}$ is a distinguisher for the $\mathsf{LWRCL}(2^q, 2^{p+\alpha}, n, m, N)$ problem,

$$\begin{aligned}
&\left|\mathbb{P}\left[\mathcal{D}\left(\mathcal{R}\left(\mathsf{LWR.Real}(\boldsymbol{k})\right)\right) = 1\right] - \mathbb{P}\left[\mathcal{D}\left(\mathcal{R}\left(\mathsf{LWR.Rand}\right)\right) = 1\right]\right| \\
&\geq \left|\mathbb{P}\left[\mathcal{D}\left(\mathsf{LWRCL.Real}(\boldsymbol{k})\right) = 1\right] - \mathbb{P}\left[\mathcal{D}\left(\mathsf{LWRCL.Rand}()\right) = 1\right]\right| \\
&\quad - \left|\mathbb{P}\left[\mathcal{D}\left(\mathcal{R}\left(\mathsf{LWR.Real}(\boldsymbol{k})\right)\right) = 1\right] - \mathbb{P}\left[\mathcal{D}\left(\mathsf{LWRCL.Real}(\boldsymbol{k})\right) = 1\right]\right| \\
&\quad - \left|\mathbb{P}\left[\mathcal{D}\left(\mathcal{R}\left(\mathsf{LWR.Rand}()\right)\right) = 1\right] - \mathbb{P}\left[\mathcal{D}\left(\mathsf{LWRCL.Rand}()\right) = 1\right]\right| \\
&> \epsilon_{\mathsf{LWRCL}} - 2 \cdot 2^{-\alpha} Nm - \epsilon_{\mathsf{LWR}} \\
&= \epsilon_{\mathsf{LWR}},
\end{aligned}$$

which concludes the proof. □

B Standard Cryptographic Primitives

B.1 Signature Schemes

Let ROdist be a distribution on functions, modeling the distribution of the random oracle H. A *signature scheme* is a triple of classical polynomial-time algorithms $\Pi_{\mathsf{Sign}} = (\mathsf{S.Gen}, \mathsf{Sign}, \mathsf{S.Verify})$ specified as follows.

Key Generation: The probabilistic key generation algorithm $\mathsf{S.Gen}$ takes as input the security parameter $\lambda \in \mathbb{N}$ (in unary) and returns a pair of keys $(vk, sk) \in \mathcal{V} \times \mathcal{S}$. We call vk the *verification key* and sk the *signing key*.

Signature Generation: The probabilistic signing algorithm Sign takes as input a message $msg \in \mathcal{M}$ and a signing key sk, and outputs a signature $\sigma \leftarrow\!\!\$\ \mathsf{Sign}^H(sk, msg)$ with $\sigma \in \Sigma$.

Signature Verification: The deterministic verification algorithm $\mathsf{S.Verify}$ takes as input a pair (msg, σ) and a verification key vk, and outputs a bit. A signature σ is called *valid* with respect to msg and vk iff $\mathsf{S.Verify}^H(vk, msg, \sigma) = 1$.

In what follows, we define *correctness* and *unforgeability* and we refer to the experiments in Fig. 9. The scheme satisfies ϵ_{Corr}-*correctness* if for every quantum polynomial-time adversary A,

$$\mathbb{P}\left[\mathsf{Corr}_{\Pi_{\mathsf{Sign}}, H}(\mathsf{A}) = 0\right] \leq \epsilon_{\mathsf{Corr}}.$$

$\mathsf{Corr}_{\Pi_{\mathsf{Sign}},H}(\mathsf{A})$

$H \leftarrow\!\!\$\ \mathsf{ROdist}$
$(vk, sk) \leftarrow\!\!\$\ \mathsf{S.Gen}(1^\lambda)$
$msg \leftarrow\!\!\$\ \mathsf{A}^H(vk)$
$\sigma \leftarrow\!\!\$\ \mathsf{Sign}^H(sk, msg)$
$b = \mathsf{S.Verify}^H(sk, msg, \sigma)$
return b

$\mathsf{UF}-\mathsf{CMA}_{\Pi_{\mathsf{Sign}},H}(\mathsf{A})$

$H \leftarrow\!\!\$\ \mathsf{ROdist}$
$(vk, sk) \leftarrow\!\!\$\ \mathsf{S.Gen}(1^\lambda)$
$(msg^*, \sigma^*) \leftarrow\!\!\$\ \mathsf{A}^{H(\cdot),\mathsf{Sign}(sk,\cdot)}(vk)$
if $msg^* \in \mathcal{Q}_{\mathsf{Sign}}$
  return 0
else
  $b = \mathsf{S.Verify}^H(vk, msg^*, \sigma^*)$
  return b

Fig. 9. The experiments Corr and $\mathsf{UF}-\mathsf{CMA}$ for correctness and unforgeability. Here, $\mathcal{Q}_{\mathsf{Sign}}$ is the list of all queries m to $\mathsf{Sign}^H(sk,\cdot)$. The oracle access to Sign is assumed to be classical, while the one to H is quantum.

As for security, we consider *unforgeability*. That is, no efficient attacker can forge a valid message-signature pair (msg^*, σ^*) for a fresh message msg^* without knowledge of the signing key.

Definition 9 (Unforgeable Signature). *We call a signature scheme Π_{Sign} $(\epsilon_{\mathsf{Sign}}, N_H, N_{\mathsf{Sign}})$-unforgeable if for all quantum polynomial-time adversaries A asking at most N_H queries to H and N_{Sign} queries to Sign, then:*

$$\mathbb{P}\left[\mathsf{UF}-\mathsf{CMA}_{\Pi_{\mathsf{Sign}},H}(\mathsf{A}) = 1\right] \leq \epsilon_{\mathsf{Sign}}(\lambda).$$

We say that Π_{Sign} is $(\epsilon_{\mathsf{Sign}}, N_H)$-unforgeable against no-message attacks if it is $(\epsilon_{\mathsf{Sign}}, N_H, 0)$-unforgeable.

B.2 Pseudorandom Generators

A pseudorandom generator (PRG) is a deterministic polynomial-time algorithm G that, upon input a seed of true randomness, expands it into a much longer pseudorandom string.

Definition 10 (Pseudorandom generator). *We call $G : \mathcal{P} \to \mathcal{K}$ a ϵ_{PRG}-secure PRG if it holds that $|\mathcal{K}| > |\mathcal{P}|$, and additionally that for every quantum polynomial-time adversary A,*

$$\left|\mathbb{P}\left[\mathsf{A}\left(G\left(\mathbf{U}_{\mathcal{P}}\right)\right) = 1\right] - \mathbb{P}\left[\mathsf{A}\left(\mathbf{U}_{\mathcal{K}}\right) = 1\right]\right| \leq \epsilon_{\mathsf{PRG}}$$

where $\mathbf{U}_S$ denotes the uniform distribution over the set S.

B.3 Weak Pseudo-Random Functions

Let $\lambda \in \mathbb{N}$, let $\mathcal{K}(\lambda), \mathcal{X}(\lambda)$ and $\mathcal{Y}(\lambda)$ be sets. Let $F(\lambda)\colon \mathcal{K}(\lambda) \times \mathcal{X}(\lambda) \to \mathcal{Y}(\lambda)$ be an efficient function. For succinctness, we omit λ from now on. The property we

consider is that of *weak-pseudorandomness*. Intuitively, this means that N evaluations of F over a random key and a random input look random as long as the key stays secret. In other words, we work with a relaxed version of conventional *pseudorandomness*, where the inputs to the function are randomly sampled. We make weak pseudorandomness formal by considering the experiments in Fig. 10.

$\mathsf{wPRF.Real}_F(\mathsf{A})$

$k \leftarrow\!\!\$\ \mathcal{K}$
$(x_i)_{i\in[N]} \leftarrow\!\!\$\ \mathcal{X}^N$
$(y_i)_{i\in[N]} \leftarrow (F_k(x_i))_{i\in[N]}$
$b \leftarrow\!\!\$\ \mathsf{A}\left((x_i, y_i)_{i\in[N]}\right)$
return b

$\mathsf{wPRF.Rand}_F(\mathsf{A})$

$k \leftarrow\!\!\$\ \mathcal{K}$
$(x_i)_{i\in[N]} \leftarrow\!\!\$\ \mathcal{X}^N$
$(y_i)_{i\in[N]} \leftarrow\!\!\$\ \mathcal{Y}^N$
$b \leftarrow\!\!\$\ \mathsf{A}\left((x_i, y_i)_{i\in[N]}\right)$
return b

Fig. 10. The experiments defining the weak pseudorandomness of F.

Definition 11 (Weak Pseudorandom Function). *F is a $(N, \epsilon_{\mathsf{wPRF}})$-weak pseudorandom function (wPRF) if for every quantum polynomial-time adversary* A*,*

$$\left|\mathbb{P}\left[\mathsf{wPRF.Real}_{\mathsf{F}}(\mathsf{A}) = 1\right] - \mathbb{P}\left[\mathsf{wPRF.Rand}_{\mathsf{F}}(\mathsf{A}) = 1\right]\right| \leq \epsilon_{\mathsf{wPRF}}.$$

We additionally consider the conventional notion of *weak security against key recovery*, where the adversary gets an input-output pair and wins when it finds the key used to generate it. As for the previous definition, the input is not adversarially chosen, but uniformly random.

Definition 12 (Weak Security against Key Recovery). *F is ϵ_{wKR}-weakly secure against key recovery if for every quantum polynomial time adversary* A*,*

$$\mathbb{P}\left[\mathsf{KR}_F(\mathsf{A}) = 1\right] \leq \epsilon_{\mathsf{wKR}},$$

where $\mathsf{KR}_F(\mathsf{A})$ is the experiment defined as follows:

$\mathsf{KR}_F(\mathsf{A})$

$k \leftarrow\!\!\$\ \mathcal{K}$
$x \leftarrow\!\!\$\ \mathcal{X}$,
$k' \leftarrow\!\!\$\ \mathsf{A}\left(x, F_k(x)\right)$
if $k' = k$
  return 1
return 0

It is well known that any pseudorandom function is also secure against key recovery. The same observation stays true for uniformly random inputs. We make it formal in the following lemma, and for completeness, provide a proof in Section C.4.

Lemma 7. *Let F be a $(N, \epsilon_{\mathsf{wPRF}})$-weak pseudorandom function with output space $\mathcal{Y}$, with $N \geq 2$. Then, F is also ϵ_{wKR}-secure against key recovery, with $\epsilon_{\mathsf{wKR}} \leq \epsilon_{\mathsf{wPRF}} + \frac{1}{|\mathcal{Y}|^{N-1}}$.*

B.4 Hash Functions

We call *hash function* a non-injective function H that maps arbitrary-length inputs into fixed-length outputs.

The notion of *collapsing* hash function was introduced by Unruh [82] to extend the notion of collision-resistance to the quantum setting. Intuitively, it states that a quantum adversary able to find collisions to the hash functions cannot tell whether they are measured or not. This notion is used to prove that our sigma protocol achieves quantum computationally unique responses (*cf.* Section 3.4).

Definition 13 (Collapsing Hash Function [82]). *Let H be a hash function, and $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ be a two-stage adversary. Consider the following two games*

$\mathsf{Game}_1(\mathsf{A})$	$\mathsf{Game}_2(\mathsf{A})$
$(state, K, h) \leftarrow \mathsf{A}_1$	$(state, K, h) \leftarrow \mathsf{A}_1$
$k \leftarrow \mathcal{M}(K)$	
$b \leftarrow \mathsf{A}_2(state, K)$	$b \leftarrow \mathsf{A}_2(state, K)$
return b	**return** b

Here, $\mathcal{M}$ denotes a measurement in the computational basis. We call an adversary A valid if $\Pr[H(k) = h] = 1$ when we run $(state, K, h) \leftarrow \mathsf{A}$ and measure K in the computational basis as k. A function H is collapsing iff for any quantum-polynomial-time valid adversary A, then

$$|\Pr[\mathsf{Game}_1(\mathsf{A}) = 1] - \Pr[\mathsf{Game}_2(\mathsf{A}) = 1]| \leq \epsilon_{\mathsf{Coll}}.$$

For a hash function H, we call *canonical commitment scheme* the couple of algorithms $(\mathsf{Commit}, \mathsf{Open})$ defined as follows. Commit takes as input one value x, concatenates it with some fresh randomness ρ, and outputs $\mathsf{H}(x \parallel \rho)$. Open outputs x and ρ.

The canonical commitment scheme is implicitly used in our construction, as we hash the key shares with uniformly random coins. Proving its zero-knowledge requires relying on its *hiding property*, which we report below for completeness.

Definition 14 (Computationally Hiding). *We call a (hash-based) commitment scheme H ϵ_{Hide}-*computationally hiding *if for every pair of inputs x_0, x_1, random coins ρ_0, ρ_1 and every quantum polynomial-time adversary A,*

$$|\mathbb{P}[\mathsf{A}(\mathsf{H}(x_0 \parallel \rho_0)) = 1] - \mathbb{P}[\mathsf{A}(\mathsf{H}(x_1 \parallel \rho_1)) = 1]| \leq \epsilon_{\mathsf{Hide}}.$$

The hiding property of the canonical commitment scheme is proven in the random oracle model in Lemma 17 of [82].

C Other results and omitted proofs

C.1 Proof of Lemma 1

First, we observe that f always produces pairs $((x,y),k) \in \mathcal{R}$. This proves the first part of the definition of HIG with $\epsilon_{\mathsf{Rel}} = 0$. For the analysis of ϵ_{HIG}, we introduce the following variants of $f.\mathsf{HIG}(\mathsf{A})$:

$\mathsf{Hyb}^{(1)}(\mathsf{A})$	$\mathsf{Hyb}^{(2)}(\mathsf{A})$
$x \leftarrow\!\!\$\ \mathcal{X}$,	$x \leftarrow\!\!\$\ \mathcal{X}$,
$y \leftarrow\!\!\$\ \mathbb{Z}_P^m$,	$y \leftarrow\!\!\$\ \mathbb{Z}_P^m$,
$k' \leftarrow\!\!\$\ \mathsf{A}(x,y)$	$k' \leftarrow\!\!\$\ \mathsf{A}(x,y)$
if $\lvert F_{k'}(x) - y\rvert \le \delta$	**if** $\lvert F_{k'}(x) - y\rvert = 0$
return 1	**return** 1
return 0	**return** 0

Lemma 1 follows from the claims below.

Claim 1 $\left|\Pr[f.\mathsf{HIG}(\mathsf{A}) = 1] - \Pr[\mathsf{Hyb}^{(1)}(\mathsf{A}) = 1]\right| \le \epsilon_{\mathsf{wPRF}}$.

Claim 2 *For every* A *there exists* $\tilde{\mathsf{A}}$ *such that*

$$\Pr[\mathsf{Hyb}^{(1)}(\mathsf{A}) = 1] = (2\delta+1)^m \cdot \Pr[\mathsf{Hyb}^{(2)}(\tilde{\mathsf{A}}) = 1].$$

Claim 3 $\Pr[\mathsf{Hyb}^{(2)}(\mathsf{A}) = 1] \le \frac{|\mathcal{K}|}{|\mathbb{Z}_P^m|} \cdot \epsilon_{\mathsf{wKR}}$.

First, we show how to derive Lemma 1 from the three claims above.

Proof (Lemma 1). We can rewrite $\Pr[f.\mathsf{HIG}(\mathsf{A}) = 1]$ as

$$\Big(\Pr[f.\mathsf{HIG}(\mathsf{A}) = 1] - \Pr[\mathsf{Hyb}^{(1)}(\mathsf{A}) = 1]\Big) + \Pr[\mathsf{Hyb}^{(1)}(\mathsf{A}) = 1].$$

Claim 1 and Claim 2 allow to upper bound the above equation with $\epsilon_{\mathsf{wPRF}} + (2\delta+1)^m \Pr[\mathsf{Hyb}^{(2)}(\mathsf{A}) = 1]$, and thus the lemma follows by using Claim 3. □

We now proceed with the proofs of each of the above claims.

Proof (Claim 1). This is a standard reduction to the PRF. Let A be an adversary in the experiments $f.\mathsf{HIG}$ and $\mathsf{Hyb}^{(1)}(\mathsf{A})$, *i.e.* an algorithm that, upon receiving $(x,y) \in \mathcal{X} \times \mathbb{Z}_P^m$, outputs a candidate key k'. Let $\tilde{\mathsf{A}}$ be an adversary for ϵ_{wPRF} that runs A on input (x_1, y_1), and returns 1 whenever A outputs k' satisfying $|F_{k'}(x_1) - y_1| \le \delta$, and 0 otherwise. Then

$$\left|\mathbb{P}\left[b = 1 \;\middle|\; \begin{array}{l} k \leftarrow\!\!\$\ \mathcal{K}, \\ \{x_i\}_{i\in[N]} \leftarrow\!\!\$\ \mathcal{X}^N, \\ b \leftarrow\!\!\$\ \tilde{\mathsf{A}}\left(\{x_i, F_k(x_i)\}_{i\in[N]}\right) \end{array}\right] - \mathbb{P}\left[b = 1 \;\middle|\; \begin{array}{l} \{x_i\}_{i\in[N]} \leftarrow\!\!\$\ \mathcal{X}^N, \\ \{y_i\}_{i\in[N]} \leftarrow\!\!\$\ (\mathbb{Z}_P^m)^N, \\ b \leftarrow\!\!\$\ \tilde{\mathsf{A}}\left(\{x_i, y_i\}_{i\in[N]}\right) \end{array}\right]\right|$$

$$= \left| \mathbb{P}\left[|F_{k'}(x_1) - y_1| \le \delta \,\middle|\, \begin{array}{l} k \leftarrow\!\!\$\ \mathcal{K} \\ x_1 \leftarrow\!\!\$\ \mathcal{X} \\ k' \leftarrow\!\!\$\ \mathsf{A}\left(x_1, F_k(x_1)\right) \end{array} \right] - \mathbb{P}\left[|F_{k'}(x_1) - y_1| \le \delta \,\middle|\, \begin{array}{l} x_1 \leftarrow\!\!\$\ \mathcal{X} \\ y_1 \leftarrow\!\!\$\ \mathbb{Z}_P^m \\ k' \leftarrow\!\!\$\ \mathsf{A}\left(x_1, y_1\right) \end{array} \right] \right| .$$

Since the first term represents a particular instance of ϵ_{wPRF} with $N = 1$, while the second yields an arbitrary instance of $\left|\Pr[f.\mathsf{HIG}(\mathsf{A}) = 1] - \Pr[\mathsf{Hyb}^{(1)}(\mathsf{A}) = 1]\right|$, the claim follows.

Proof (Claim 2). This claim formalizes the intuition that finding k' satisfying $|F_{k'}(x) - y| \le \delta$ is $(2\delta + 1)^m$ times easier than finding k' satisfying $F_{k'}(x) = y$. We now prove this observation formally. Let A be an adversary for $\mathsf{Hyb}^{(1)}$. We construct an adversary $\tilde{\mathsf{A}}$ for $\mathsf{Hyb}^{(2)}$. Upon input two random values (x, y), $\tilde{\mathsf{A}}$ samples a random e in the interval $[-\delta, +\delta]^m$ and computes $y' = y + e$. It runs A upon input (x, y'), and outputs the same key k' as A. Note that whenever A produces a key k' for (x, y'), then k' satisfies $F_{k'}(x) = y$ with probability $\frac{1}{(2\delta+1)^m}$, as

$$\mathbb{P}\left[F_{k'}(x) = y \,\middle|\, \begin{array}{l} x \leftarrow\!\!\$\ \mathcal{X}, \\ y \leftarrow\!\!\$\ \mathbb{Z}_P^m, \\ k' \leftarrow\!\!\$\ \tilde{\mathsf{A}}\left(x, y\right) \end{array} \right] = \mathbb{P}\left[F_{k'}(x) = y \,\middle|\, \begin{array}{l} x \leftarrow\!\!\$\ \mathcal{X}, \\ y \leftarrow\!\!\$\ \mathbb{Z}_P^m, \\ e \leftarrow\!\!\$\ [-\delta, +\delta]^m, \\ y' = y + e, \\ k' \leftarrow\!\!\$\ \mathsf{A}\left(x, y'\right) \end{array} \right]$$

$$= \frac{1}{(2\delta+1)^m} \cdot \sum_{e^* \in [-\delta, +\delta]^m} \mathbb{P}\left[F_{k'}(x) = y' - e^* \,\middle|\, \begin{array}{l} x \leftarrow\!\!\$\ \mathcal{X}, \\ y' \leftarrow\!\!\$\ \mathbb{Z}_P^m, \\ k' \leftarrow\!\!\$\ \mathsf{A}\left(x, y'\right) \end{array} \right]$$

$$= \frac{1}{(2\delta+1)^m} \cdot \mathbb{P}\left[|F_{k'}(x) - y'| \le \delta \,\middle|\, \begin{array}{l} x \leftarrow\!\!\$\ \mathcal{X}, \\ y' \leftarrow\!\!\$\ \mathbb{Z}_P^m, \\ k' \leftarrow\!\!\$\ \mathsf{A}\left(x, y'\right) \end{array} \right] .$$

This proves that for every A there exists in $\mathsf{Hyb}^{(2)}$ such that $\Pr[\mathsf{Hyb}^{(1)}(\mathsf{A}) = 1] = (2\delta + 1)^m \cdot \Pr[\mathsf{Hyb}^{(2)}(\tilde{\mathsf{A}}) = 1]$.

Proof (Claim 3). This claim formalizes the observation that finding a key k' satisfying $F_{k'}(x) = y$ is easier than finding the exact key k used to compute y from x. To intuitively quantify how easier it is, we use the fact that there are in average $\frac{|\mathcal{K}|}{|\mathbb{Z}_P^m|}$ keys k' for every possible (x, y). Therefore, finding a key such that $F_{k'}(x) = y$ is $\frac{|\mathcal{K}|}{|\mathbb{Z}_P^m|}$ times easier than finding the exact key used for generating (x, y). Below, we prove this observation formally. By employing the law of total probability and Bayes' Theorem, we can rewrite $\Pr[\mathsf{Hyb}^{(2)}(\mathsf{A}) = 1]$ as

$$\mathbb{P}\left[F_{k'}(x) = y \,\middle|\, \begin{array}{l} x \leftarrow\!\!\$\ \mathcal{X}, \\ y \leftarrow\!\!\$\ \mathbb{Z}_P^m, \\ k' \leftarrow\!\!\$\ \mathsf{A}\left(x, y\right) \end{array} \right]$$

$$
\begin{aligned}
&= \sum_{(x^*,y^*)\in\mathcal{X}\times\mathbb{Z}_P^m} \mathbb{P}\left[\begin{array}{l|l} F_{k'}(x)=y, & x \leftarrow\!\!\$\ \mathcal{X}, \\ x=x^*, & y \leftarrow\!\!\$\ \mathbb{Z}_P^m, \\ y=y^* & k' \leftarrow\!\!\$\ \mathsf{A}(x,y) \end{array}\right] \\
&= \sum_{(x^*,y^*)\in\mathcal{X}\times\mathbb{Z}_P^m} \mathbb{P}\left[F_{k'}(x^*)=y^* \,\middle|\, k' \leftarrow\!\!\$\ \mathsf{A}(x^*,y^*)\right] \cdot \mathbb{P}\left[\begin{array}{l|l} x=x^*, & x \leftarrow\!\!\$\ \mathcal{X}, \\ y=y^* & y \leftarrow\!\!\$\ \mathbb{Z}_P^m \end{array}\right] \\
&= \frac{1}{|\mathcal{X}|\cdot|\mathbb{Z}_P^m|} \cdot \sum_{(x^*,y^*)\in\mathcal{X}\times\mathbb{Z}_P^m} \mathbb{P}\left[F_{k'}(x^*)=y^* \,\middle|\, k' \leftarrow\!\!\$\ \mathsf{A}(x^*,y^*)\right]. \qquad (6)
\end{aligned}
$$

In what follows, we show that $\frac{|\mathcal{K}|}{|\mathbb{Z}_P^m|}\Pr[\mathsf{KR}_F(\mathsf{A})=1]$ equals *Eq.* (6) as well. First, we use the fact that $k'=k$ implies that $F_{k'}(x)=F_k(x)$

$$
\begin{aligned}
\Pr[\mathsf{KR}_F(\mathsf{A})=1] &= \mathbb{P}\left[k'=k \,\middle|\, \begin{array}{l} k \leftarrow\!\!\$\ \mathcal{K}, \\ x \leftarrow\!\!\$\ \mathcal{X}, \\ k' \leftarrow\!\!\$\ \mathsf{A}(x,F_k(x)) \end{array}\right] \\
&= \mathbb{P}\left[\begin{array}{l} k'=k, \\ F_{k'}(x)=F_k(x) \end{array} \,\middle|\, \begin{array}{l} k \leftarrow\!\!\$\ \mathcal{K}, \\ x \leftarrow\!\!\$\ \mathcal{X}, \\ k' \leftarrow\!\!\$\ \mathsf{A}(x,F_k(x)) \end{array}\right].
\end{aligned}
$$

Next, we use both the law of total probabilities and Bayes' theorem, yielding

$$
\begin{aligned}
\Pr[\mathsf{KR}_F(\mathsf{A})=1] &= \frac{1}{|\mathcal{X}|}\sum_{x^*\in\mathcal{X}} \mathbb{P}\left[\begin{array}{l} k'=k, \\ F_{k'}(x^*)=F_k(x^*) \end{array} \,\middle|\, \begin{array}{l} k \leftarrow\!\!\$\ \mathcal{K}, \\ k' \leftarrow\!\!\$\ \mathsf{A}(x^*,F_k(x^*)) \end{array}\right] \\
&= \frac{1}{|\mathcal{X}|}\sum_{x^*\in\mathcal{X}} \mathbb{P}\left[k'=k \,\middle|\, \begin{array}{l} k \leftarrow\!\!\$\ \mathcal{K}, \\ k' \leftarrow\!\!\$\ \mathsf{A}(x^*,F_k(x^*)), \\ F_{k'}(x^*)=F_k(x^*) \end{array}\right] \\
&\qquad \cdot \mathbb{P}\left[F_{k'}(x^*)=F_k(x^*) \,\middle|\, \begin{array}{l} k \leftarrow\!\!\$\ \mathcal{K}, \\ k' \leftarrow\!\!\$\ \mathsf{A}(x^*,F_k(x^*)) \end{array}\right] \\
&= \frac{1}{|\mathcal{X}|}\sum_{x^*\in\mathcal{X}} \mathbb{P}\left[F_{k'}(x^*)=F_k(x^*) \,\middle|\, \begin{array}{l} k \leftarrow\!\!\$\ \mathcal{K}, \\ k' \leftarrow\!\!\$\ \mathsf{A}(x^*,F_k(x^*)) \end{array}\right] \\
&\qquad \cdot \sum_{y^*\in\mathbb{Z}_P^m} \mathbb{P}\left[\begin{array}{l} k'=k, \\ F_k(x^*)=y^* \end{array} \,\middle|\, \begin{array}{l} k \leftarrow\!\!\$\ \mathcal{K}, \\ k' \leftarrow\!\!\$\ \mathsf{A}(x^*,F_k(x^*)), \\ F_{k'}(x^*)=F_k(x^*) \end{array}\right] \\
&= \frac{1}{|\mathcal{X}|}\sum_{x^*\in\mathcal{X}} \mathbb{P}\left[F_{k'}(x^*)=F_k(x^*) \,\middle|\, \begin{array}{l} k \leftarrow\!\!\$\ \mathcal{K}, \\ k' \leftarrow\!\!\$\ \mathsf{A}(x^*,F_k(x^*)) \end{array}\right] \\
&\qquad \cdot \sum_{y^*\in\mathbb{Z}_P^m} \mathbb{P}\left[F_k(x^*)=y^* \,\middle|\, \begin{array}{l} k \leftarrow\!\!\$\ \mathcal{K}, \\ k' \leftarrow\!\!\$\ \mathsf{A}(x^*,F_k(x^*)), \\ F_{k'}(x^*)=F_k(x^*) \end{array}\right] \qquad (7) \\
&\qquad \cdot \mathbb{P}\left[k'=k \,\middle|\, \begin{array}{l} k \leftarrow\!\!\$\ \{k\colon F_k(x^*)=y^*\}, \\ k' \leftarrow\!\!\$\ \mathsf{A}(x^*,F_k(x^*)), \\ F_{k'}(x^*)=F_k(x^*) \end{array}\right].
\end{aligned}
$$

To conclude the proof, we analyze the probabilities appearing in Eq. (7). In the second probability, we can apply Bayes twice, yielding

$$\begin{aligned}
&\mathbb{P}\left[F_k(x^*) = y^* \;\middle|\; \begin{array}{l} k \leftarrow\!\!\$\ \mathcal{K}, \\ k' \leftarrow\!\!\$\ \mathsf{A}\left(x^*, F_k(x^*)\right), \\ F_{k'}(x^*) = F_k(x^*) \end{array} \right] \\
&= \mathbb{P}\left[\begin{array}{l} F_k(x^*) = y^*, \\ F_{k'}(x^*) = F_k(x^*) \end{array} \;\middle|\; \begin{array}{l} k \leftarrow\!\!\$\ \mathcal{K}, \\ k' \leftarrow\!\!\$\ \mathsf{A}\left(x^*, F_k(x^*)\right) \end{array} \right] \\
&\quad \cdot \left(\mathbb{P}\left[F_{k'}(x^*) = F_k(x^*) \;\middle|\; \begin{array}{l} k \leftarrow\!\!\$\ \mathcal{K}, \\ k' \leftarrow\!\!\$\ \mathsf{A}\left(x^*, F_k(x^*)\right) \end{array} \right] \right)^{-1} \\
&= \mathbb{P}\left[F_{k'}(x^*) = F_k(x^*) \;\middle|\; \begin{array}{l} k \leftarrow\!\!\$\ \mathcal{K}, \\ k' \leftarrow\!\!\$\ \mathsf{A}\left(x^*, F_k(x^*)\right), \\ F_k(x^*) = y^* \end{array} \right] \\
&\quad \cdot \mathbb{P}\left[F_k(x^*) = y^* \;\middle|\; \begin{array}{l} k \leftarrow\!\!\$\ \mathcal{K}, \\ k' \leftarrow\!\!\$\ \mathsf{A}\left(x^*, F_k(x^*)\right) \end{array} \right] \\
&\quad \cdot \left(\mathbb{P}\left[F_{k'}(x^*) = F_k(x^*) \;\middle|\; \begin{array}{l} k \leftarrow\!\!\$\ \mathcal{K}, \\ k' \leftarrow\!\!\$\ \mathsf{A}\left(x^*, F_k(x^*)\right) \end{array} \right] \right)^{-1}.
\end{aligned} \tag{8}$$

The first term of Eq. (8) can be restated as

$$\begin{aligned}
&\mathbb{P}\left[F_{k'}(x^*) = F_k(x^*) \;\middle|\; \begin{array}{l} k \leftarrow\!\!\$\ \mathcal{K}, \\ k' \leftarrow\!\!\$\ \mathsf{A}\left(x^*, F_k(x^*)\right), \\ F_k(x^*) = y^* \end{array} \right] \\
&= \mathbb{P}\left[F_{k'}(x^*) = y^* \;\middle|\; k' \leftarrow\!\!\$\ \mathsf{A}\left(x^*, y^*\right) \right].
\end{aligned}$$

For the second term of Eq. (8), we observe that the probability depends uniquely on the structure of the weak PRF

$$\begin{aligned}
&\mathbb{P}\left[F_k(x^*) = y^* \;\middle|\; \begin{array}{l} k \leftarrow\!\!\$\ \mathcal{K}, \\ k' \leftarrow\!\!\$\ \mathsf{A}\left(x^*, F_k(x^*)\right) \end{array} \right] \\
&= \mathbb{P}\left[F_k(x^*) = y^* \;\middle|\; k \leftarrow\!\!\$\ \mathcal{K} \right] \\
&= \frac{|\{k \colon F_k(x^*) = y^*\}|}{|\mathcal{K}|}.
\end{aligned}$$

For the last term of Eq. (8), we remark that it is independent of y^* and it's the inverse of the first term of Eq. (7). Summing up, the second probability in Eq. (7) can be rewritten as Eq. (8), which in turn equals

$$\begin{aligned}
&\mathbb{P}\left[F_{k'}(x^*) = y^* \;\middle|\; k' \leftarrow\!\!\$\ \mathsf{A}\left(x^*, y^*\right) \right] \\
&\cdot \frac{|\{k \colon F_k(x^*) = y^*\}|}{|\mathcal{K}|} \\
&\cdot \left(\mathbb{P}\left[F_{k'}(x^*) = F_k(x^*) \;\middle|\; \begin{array}{l} k \leftarrow\!\!\$\ \mathcal{K}, \\ k' \leftarrow\!\!\$\ \mathsf{A}\left(x^*, F_k(x^*)\right) \end{array} \right] \right)^{-1}.
\end{aligned}$$

In the last term of Eq. (7), we condition on the event where $F_{k'}(x^*) = F_k(x^*)$ and k' is chosen by A by only observing $(x^*, F_k(x^*))$. In this setting, A can only guess the exact key with uniform probability among the keys yielding the same output, *i.e.*

$$\mathbb{P}\left[k' = k, \middle| \begin{array}{l} k \leftarrow\!\!\$\ \{k\colon F_k(x^*) = y^*\}, \\ k' \leftarrow\!\!\$\ \mathsf{A}\left(x^*, F_k(x^*)\right), \\ F_{k'}(x^*) = F_k(x^*) \end{array}\right] = \frac{1}{|\{k\colon F_k(x^*) = y^*\}|}.$$

Going back to Eq. (7), we get

$$\begin{aligned} \Pr[\mathsf{KR}_F(\mathsf{A}) = 1] &= \frac{1}{|\mathcal{X}|} \sum_{(x^*,y^*)\in\mathcal{X}\times\mathbb{Z}_P^m} \mathbb{P}\left[F_{k'}(x^*) = y^* \;\middle|\; k' \leftarrow\!\!\$\ \mathsf{A}\left(x^*, y^*\right)\right] \\ &\qquad \cdot \frac{|\{k\colon F_k(x^*) = y^*\}|}{|\mathcal{K}|} \cdot \frac{1}{|\{k\colon F_k(x^*) = y^*\}|} \\ &= \frac{1}{|\mathcal{X}| \cdot |\mathcal{K}|} \sum_{(x^*,y^*)\in\mathcal{X}\times\mathbb{Z}_P^m} \mathbb{P}\left[F_{k'}(x^*) = y^* \;\middle|\; k' \leftarrow\!\!\$\ \mathsf{A}\left(x^*, y^*\right)\right]. \end{aligned}$$

This proves the claim.

C.2 Proof of Lemma 2

To prove that F is a weak PRF, we must upper bound the following quantity for every quantum polynomial adversary A:

$$\left|\mathbb{P}\left[\mathsf{wPRF.Real}_{\mathsf{F}}(\mathsf{A}) = 1\right] - \mathbb{P}\left[\mathsf{wPRF.Rand}_{\mathsf{F}}(\mathsf{A}) = 1\right]\right|,$$

where $\mathsf{wPRF.Real}_F(\mathsf{A})$ and $\mathsf{wPRF.Rand}_F(\mathsf{A})$ are the experiments defining the security of F as a weak PRF (*cf.* Section B.3). In what follows, we prove that

$$\mathsf{wPRF.Real}_{\mathsf{F}}(\mathsf{A}) \equiv \mathsf{LWR}\left(Q, P, n, Nm\right).\mathsf{Real}(\boldsymbol{k}) \tag{9}$$

and

$$\mathsf{wPRF.Rand}_{\mathsf{F}}(\mathsf{A}) \equiv \mathsf{LWR}\left(Q, P, n, Nm\right).\mathsf{Rand}(), \tag{10}$$

when $\boldsymbol{k} \leftarrow\!\!\$\ \mathbb{Z}_Q^n$. Note that Eq. (9) and Eq. (10) imply Lemma 2, as then

$$\begin{aligned} &\left|\mathbb{P}\left[\mathsf{wPRF.Real}_F(\mathsf{A}) = 1\right] - \mathbb{P}\left[\mathsf{wPRF.Rand}_F(\mathsf{A}) = 1\right]\right| \\ &= \left|\mathbb{P}\left[\mathsf{A}\left(\mathsf{LWR.Real}(\boldsymbol{k})\right) = 1 \mid \boldsymbol{k} \leftarrow\!\!\$\ \mathbb{Z}_Q^n\right] - \mathbb{P}\left[\mathsf{A}\left(\mathsf{LWR.Rand}()\right) = 1\right]\right| \\ &\leq \epsilon_{\mathsf{LWR}}. \end{aligned}$$

Therefore, in the rest of the proof we show Eq. (9) and Eq. (10). For a better visualization, we report the experiments of interest in Fig. 11.

The only difference between the experiments wPRF and LWR (both in the Real and Rand case) is that the first one samples N small matrices $(\boldsymbol{X}_i)_{i\in[N]}$ in $\mathbb{Z}_Q^{m\times n}$, while the second directly samples a big matrix $\boldsymbol{X} \in \mathbb{Z}_Q^{Nm\times n}$. The key

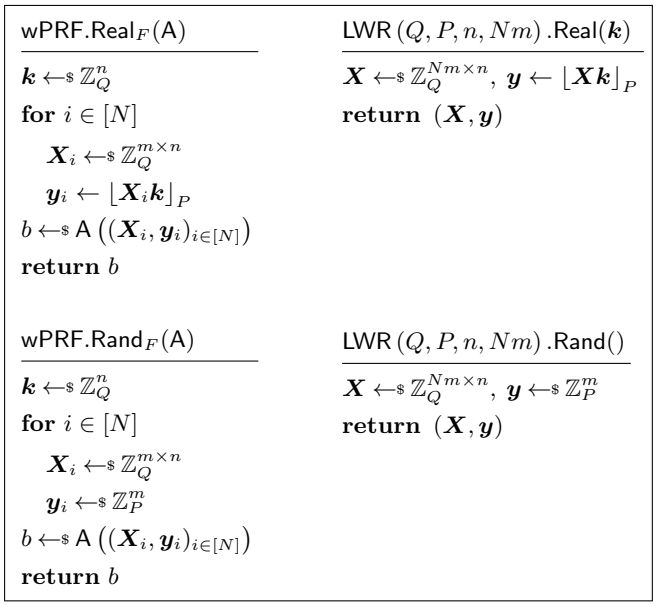

Fig. 11. The experiments involved in the proof of Lemma 2.

insight here is to think of $\boldsymbol{X}$ as the matrix where the small matrices $(\boldsymbol{X}_i)_{i\in[N]}$ are stacked one on top of the other, *i.e.*

$$\begin{pmatrix} \boldsymbol{X}_1 \\ \boldsymbol{X}_2 \\ \vdots \\ \boldsymbol{X}_N \end{pmatrix}.$$

Therefore, sampling $\boldsymbol{X}$ at random in $\mathbb{Z}_Q^{Nm\times n}$ is equivalent to sampling $(\boldsymbol{X}_i)_{i\in[N]}$ at random in $\mathbb{Z}_Q^{m\times n}$. In the Rand experiments, the same reasoning applies to $\boldsymbol{y}$ and $(\boldsymbol{y}_i)_{i\in[N]}$, and therefore Eq. (10) follows. To prove Eq. (9), we further observe that multiplying $\boldsymbol{X}$ by $\boldsymbol{k}$ is equivalent to multiplying $(\boldsymbol{X}_i)_{i\in[N]}$ by $\boldsymbol{k}$ individually and then stacking the results on top of each other, i.e.

$$\begin{pmatrix} \boldsymbol{X}_1 \\ \boldsymbol{X}_2 \\ \vdots \\ \boldsymbol{X}_N \end{pmatrix} \cdot (\boldsymbol{k}) = \begin{pmatrix} \boldsymbol{X}_1\boldsymbol{k} \\ \boldsymbol{X}_2\boldsymbol{k} \\ \vdots \\ \boldsymbol{X}_N\boldsymbol{k} \end{pmatrix}.$$

Since $(\boldsymbol{y}_i)_{i\in[N]}$ are computed by rounding $(\boldsymbol{X}_i\boldsymbol{k})_{i\in[N]}$, and since the rounding acts component-wise, then $\boldsymbol{y}$ is also the same as stacking $(\boldsymbol{y}_i)_{i\in[N]}$ on top of each other. This proves the claim.

C.3 Proof of Lemma 3

Lemma 1 guarantees that the PRF-based distribution $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$ is a hard-instance generator for $\mathcal{R}$, with $\epsilon_{\mathsf{Rel}} = 0$ and

$$\epsilon_{\mathsf{HIG}} = \epsilon_{\mathsf{wPRF}} + 3^m \cdot \frac{Q^n}{P^m} \cdot \epsilon_{\mathsf{wKR}}.$$

Here ϵ_{wPRF} and ϵ_{wKR} correspond, respectively, to the pseudorandomness and the security against key recovery of F. According to Lemma 7, a $(N, \epsilon_{\mathsf{wPRF}})$-weak pseudorandom function with output space $\mathbb{Z}_P^m$ is also ϵ_{wKR}-secure against key recovery with $N = 2$, with $\epsilon_{\mathsf{wKR}} \leq \epsilon_{\mathsf{wPRF}} + \frac{1}{P^{mN}}$. According to Lemma 2, $\epsilon_{\mathsf{wPRF}} \leq \epsilon_{\mathsf{LWR}}$. The theorem follows.

C.4 Proof of Lemma 7

This lemma describes the standard observation that recovering the key of a PRF is harder than distinguishing it from random. For completeness, we report its adaptation to the weaker PRF notion we consider. Let A be any adversary in the game underlying ϵ_{wKR}, *i.e.* an algorithm that, upon receiving a couple $(x, y) \in \mathcal{X} \times \mathcal{Y}$, returns a candidate key k'. We now use A to construct an adversary $\tilde{\mathsf{A}}$ for the experiment corresponding to ϵ_{wPRF}. Upon input $\{x, y\}_{i\in[N]}$, $\tilde{\mathsf{A}}$ runs A over the first pair (x_1, y_1). If the key k' returned by A satisfies $F_{k'}(x_i) = y_i$ for every $i \in [N]$, $\tilde{\mathsf{A}}$ outputs 1. Otherwise, the output is 0. This implies that

$$\begin{aligned}
&\left|\mathbb{P}\left[\mathsf{wPRF.Real}_{\mathsf{F}}(\tilde{\mathsf{A}}) = 1\right] - \mathbb{P}\left[\mathsf{wPRF.Rand}_{\mathsf{F}}(\tilde{\mathsf{A}}) = 1\right]\right| \\
&= \left|\mathbb{P}\left[b = 1 \;\middle|\; \begin{array}{l} k \leftarrow_{\$} \mathcal{K}, \\ \{x_i\}_{i\in[N]} \leftarrow_{\$} \mathcal{X}^N, \\ b \leftarrow_{\$} \tilde{\mathsf{A}}\left(\{x_i, F_k(x_i)\}_{i\in[N]}\right) \end{array}\right] - \mathbb{P}\left[b = 1 \;\middle|\; \begin{array}{l} \{x_i\}_{i\in[N]} \leftarrow_{\$} \mathcal{X}^N, \\ \{y_i\}_{i\in[N]} \leftarrow_{\$} \mathcal{Y}^N, \\ b \leftarrow_{\$} \tilde{\mathsf{A}}\left(\{x_i, y_i\}_{i\in[N]}\right) \end{array}\right]\right| \\
&\geq \mathbb{P}\left[b = 1 \;\middle|\; \begin{array}{l} k \leftarrow_{\$} \mathcal{K}, \\ \{x_i\}_{i\in[N]} \leftarrow_{\$} \mathcal{X}^N, \\ k' \leftarrow_{\$} \mathsf{A}\left(x_1, F_k(x_1)\right), \\ k' = k, \\ b \leftarrow_{\$} \tilde{\mathsf{A}}\left(\{x_i, F_k(x_i)\}_{i\in[N]}\right) \end{array}\right] \cdot \mathbb{P}\left[k' = k \;\middle|\; \begin{array}{l} k \leftarrow_{\$} \mathcal{K}, \\ x_1 \leftarrow_{\$} \mathcal{X}, \\ k' \leftarrow_{\$} \mathsf{A}\left(x_1, F_k(x_1)\right) \end{array}\right] \\
&\quad + \mathbb{P}\left[\begin{array}{l} b = 1, \\ k' \neq k \end{array} \;\middle|\; \begin{array}{l} k \leftarrow_{\$} \mathcal{K}, \\ \{x_i\}_{i\in[N]} \leftarrow_{\$} \mathcal{X}^N, \\ k' \leftarrow_{\$} \mathsf{A}\left(x_1, F_k(x_1)\right), \\ b \leftarrow_{\$} \tilde{\mathsf{A}}\left(\{x_i, F_k(x_i)\}_{i\in[N]}\right) \end{array}\right] \\
&\quad - \mathbb{P}\left[b = 1 \;\middle|\; \begin{array}{l} \{x_i\}_{i\in[N]} \leftarrow_{\$} \mathcal{X}^N, \\ \{y_i\}_{i\in[N]} \leftarrow_{\$} \mathcal{Y}^N, \\ k' \leftarrow_{\$} \mathsf{A}\left(x_1, F_k(x_1)\right), \\ b \leftarrow_{\$} \tilde{\mathsf{A}}\left(\{x_i, y_i\}_{i\in[N]}\right) \end{array}\right]
\end{aligned}$$

where the inequality above follows by the definition of conditional probability and the law of total probabilities. In the right-hand side of the equation, the first probability equals 1 by definition of $\tilde{\mathsf{A}}$, the second is an arbitrary instance for $\mathsf{KR}_F(\mathsf{A})$, and the third can be lower-bounded with 0. As for the last probability, $b = 1$ happens only when the randomly sampled elements $y_2, \ldots, y_N$ hit exactly the evaluations of $F_{k'}$ over $x_2, \ldots, x_N$, which happens with probability $\frac{1}{|\mathcal{Y}|^{N-1}}$. The lemma follows.

C.5 Proof of Theorem 2

The main purpose of this section is to show how Theorem 2 follows from the work of [47]. In particular, we restate some of their results using our notation, and include relations between parameters that they only state implicitly. Theorem 2 is directly implied by the two results below.

Theorem 12 (Theorem 25 of [47], restated, for $s = 2$ and $\epsilon_{\Sigma.\mathsf{SS}} = 0$). *Let Σ be a Sigma protocol with challenge space size C, $(s, \epsilon_{\Sigma.\mathsf{SS}})$-special soundness with $s = 2$ and $\epsilon_{\Sigma.\mathsf{SS}} = 0$, and ϵ_{QCUR}-quantum computationally unique responses. Then Σ is a computational proof of knowledge for adaptive adversaries as in Definition 14 of [47], with parameters $p = 1, d = 3$ and $\kappa = \frac{1}{C} + 2\epsilon_{\mathsf{QCUR}}$.*

Theorem 13 (Theorem 21 of [47], restated, for $d = 3$). *Let Σ be a Sigma protocol for a relation $\mathcal{R}$ with challenge space size C, satisfying the adaptive proof of knowledge property with parameters $p, d = 3$, and κ as in Definition 14 of [47]. Let $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$ be a ϵ_{HIG}-hard instance generator yielding a pair of statements and witnesses in $\mathcal{R}$. Let Π_{Sign} be the signature scheme constructed from $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$ and Σ via the Fiat-Shamir with Abort transform (cf. Fig. 2). Then Π_{Sign} is $(\epsilon_{\mathsf{Sign},0}, N_H)$-unforgeable against no-message attack, with*

$$\epsilon_{\mathsf{Sign},0} \leq \left(\epsilon_{\mathsf{HIG}} N_H^6 p(\eta) + \kappa(\eta) + \frac{1}{p(\eta)} \cdot \left(\frac{1}{8N_H^3 C^3} + \frac{1}{\mathcal{O}(N_H^5)C}\right)\right)^{1/3}$$

In the rest of this section, we relate Theorem 12 and Theorem 13 with the corresponding results in [47]. To do it, we also restate two of their intermediate results. We present them below.

Lemma 8 (Lemma 15 of [47], with parameters). *If Σ is a computational/statistical proof of knowledge for static A with parameters p, d, κ, then it is also a computational/statistical proof of knowledge for adaptive A with the same parameters (cf. Definition 14 of [47]).*

Corollary 1 (Corollary 16 of [47], with parameters, for $d = 3$). *Let Σ be a Sigma protocol with super polynomial challenge space size C. Let $\mathsf{FS}(\Sigma)$ be the Fiat-Shamir-based non-interactive proof system described in Section 3.2 of [47]. If Σ is a computational/statistical proof of knowledge for static adversaries with parameters p, d, κ, with $d = 3$, then $\mathsf{FS}(\Sigma)$ is a computational/statistical proof of knowledge for adaptive adversaries (cf. Definition 14 of [47]) with parameters*

p, d as in Σ, and $e = 6, \mu(\eta) = \kappa(\eta) + \frac{1}{p(\eta)} \cdot \left(\frac{1}{8N_H^3 C^3} + \frac{1}{\mathcal{O}(N_H^5)C}\right)$, where N_H is the number of queries to the random oracles made by the attacker to $\mathsf{FS}(\Sigma)$.

First, we prove Theorem 12 and Theorem 13 based on Lemma 8, Corollary 1, and the corresponding results in [47].

Proof (Theorem 12). The statement of Theorem 12 differs from the one of Theorem 25 in [47] in the following points:

1. They assume t-soundness, while we assume $(s, \epsilon_{\Sigma.\mathsf{SS}})$-special soundness with $s = 2$ and $\epsilon_{\Sigma.\mathsf{SS}} = 0$.
2. They claim that Sigma is a computational proof of knowledge, while we claim that it is a computational proof of knowledge *for adaptive adversaries*.
3. We include the parameters for the proof of knowledge property.

We discuss each of these points, starting from Item 1. In [47] (section 5), t-soundness is defined as follows: "for any first message a, for uniformly random chosen challenges $c_1, \ldots, c_t$, and for any responses $z_1, \ldots, z_t$ with $V(x, a_i, c_i, z_i)$ for all $i \in \{1, \ldots, t\}$, a witness w for x can be efficiently computed except with negligible probability (over the choices of c_i)". In other words, Theorem 25 [47] works with a weaker version of $(s, \epsilon_{\Sigma.\mathsf{SS}})$-special soundness, where the challenges are random instead of adversarially chosen, and $\epsilon_{\Sigma.\mathsf{SS}}$ is negligible. Therefore, its statements still holds under the stronger assumption of $(s, \epsilon_{\Sigma.\mathsf{SS}})$-special soundness. Moreover, since their result holds for any s and every negligible $\epsilon_{\Sigma.\mathsf{SS}}$, it also holds for $s = 2$ and $\epsilon_{\Sigma.\mathsf{SS}} = 0$.

We now move to Item 2. In their Theorem 25 [47], the authors do not mention whether they prove the computational proof of knowledge for static or adaptive adversaries. Nonetheless, their Lemma 15 (Lemma 8 in our work) guarantees that the two notions are equivalent. For readability, we here claim computational proof of knowledge for adaptive adversaries, while only proving it for static ones.

It remains to address Item 3, and thus motivate the parameters. For this part, we refer to both the proof of Theorem 25 in [47] and of Theorem 9 in Unruh's work [80], as the first only provides a proof sketch, while the second offers a comprehensive analysis of a similar statement. Both works consider the canonical extractor (see Section 3 of [80] for a detailed description), which works as follows. It runs the malicious prover A to the point where it outputs the first message *firstmes*. Then, it chooses a random challenge ch_1 and sends it to A, obtaining a response $resp_1$ by measuring A's corresponding register. The extractor E rewinds A on the measured state, and chooses and sends to A a fresh random challenge ch_2, resulting in a response $resp_2$. If $\Sigma.\mathsf{Verify}(x, \mathit{firstmes}, ch_s, resp_s) = 1$ for all $s \in \{1, 2\}$, then E can compute the witness w using the extractor E_{SS} corresponding to the special soundness property. Both proofs follow a hybrid argument, and we refer to the hybrid experiments Game 1, ... , Game 7 defined in the proof of Theorem 9 in [80]. Below, we only report the most relevant hybrids to our proof.

Game 1. Game 1 is the original experiment defining the proof of knowledge property (*cf.* Definition 14 of [47]). the success event in Game 1, denoted by Succ:Game 1, describes the event where E outputs a valid witness w for x.

Game 2. Game 2 is identical to Game 1, except for the definition of the success event. Success Succ in Game 2 corresponds to the event where the transcripts $(\mathit{firstmes}, ch_1, \mathit{resp}_1)$ and $(\mathit{firstmes}, ch_2, \mathit{resp}_2)$ are accepting for x and $ch_1 \neq ch_2$.

Game 5. Game 5 is a variant of the above experiments where, before measuring the responses $\mathit{resp}_1, \mathit{resp}_2$, the extractor E measures whether the transcripts $(\mathit{firstmes}, ch_1, \mathit{resp}_1)$ and $(\mathit{firstmes}, ch_2, \mathit{resp}_2)$ yield an accepting conversation. Moreover, the extractor E doesn't run E_{SS} anymore. Here, Succ denotes the event where $ch_1 \neq ch_2$ and both verification measurements succeed.

Game 6. Game 6 is identical to Game 5, except that the measurements of $\mathit{resp}_1, \mathit{resp}_2$ are removed.

The goal of the proof is to lower bound $\Pr_{\mathsf{E}} = \Pr[\mathsf{Succ} : \text{Game 1}]$. Since special soundness holds with $\epsilon_{\Sigma.\mathsf{SS}} = 0$, success in Game 2 always implies success in Game 1. Therefore, $\Pr_{\mathsf{E}} = \Pr[\mathsf{Succ} : \text{Game 1}] \geq \Pr[\mathsf{Succ} : \text{Game 2}]$. For the same reasons explained in the proof of Theorem 9 [80], $\Pr[\mathsf{Succ} : \text{Game 2}] = \Pr[\mathsf{Succ} : \text{Game 5}]$. Notably, the only difference between Game 5 and 6 is that the second doesn't measure the values $\mathit{resp}_1, \mathit{resp}_2$ involved in the verification measurements. Since Σ has quantum computationally unique responses (*cf.* Definition 6), by union bound it holds $|\Pr[\mathsf{Succ} : \text{Game 5}] - \Pr[\mathsf{Succ} : \text{Game 6}]| \leq 2\epsilon_{\mathsf{QCUR}}$. The rest is identical to Unruh's proof of Theorem 9. Namely,

$$\Pr[\mathsf{Succ} : \text{Game 6}] \geq \Pr_{\mathsf{V}}\left(\Pr_{\mathsf{V}}^2 - \frac{1}{C}\right) \geq \Pr_{\mathsf{V}}^3 - \frac{1}{C},$$

where $\Pr_{\mathsf{V}}$ denotes the probability that A generates an accepting proof in the static setting. Putting things together, we get

$$\begin{aligned}
\Pr_{\mathsf{E}} &= \Pr[\mathsf{Succ} : \text{Game 1}] \\
&\geq \Pr[\mathsf{Succ} : \text{Game 2}] \\
&= \Pr[\mathsf{Succ} : \text{Game 5}] \\
&\geq \Pr[\mathsf{Succ} : \text{Game 6}] - |\Pr[\mathsf{Succ} : \text{Game 5}] - \Pr[\mathsf{Succ} : \text{Game 6}]| \\
&\geq \Pr_{\mathsf{V}}^3 - \frac{1}{C} - 2\epsilon_{\mathsf{QCUR}}.
\end{aligned}$$

This concludes the proof.

Proof (Theorem 13). The statement of Theorem 13 differs from the one of Theorem 21 of [47] in the following points.

1. We assume that $d = 3$.
2. Theorem 13 assumes that $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$ is a hard-instance generator, while Theorem 21 assumes a hard relation $\mathcal{R}$.
3. Theorem 13 assumes adaptive proofs of knowledge, while Theorem 21 does not specify the static/adaptive variant.
4. Theorem 13 refers to the Fiat-Shamir *with abort* transform (*cf.* Fig. 2), while Theorem 21 proves unforgeability for standard Fiat-Shamir signatures (*cf.* Definition 19 in [47]).

5. Theorem 13 provides an explicit upper bound based on the underlying parameters.

In what follows, we address each of these points, starting from Item 1. Theorem 21 is true for every d, therefore the same holds for a specific choice of d. We now move to Item 2. Since $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$ is a distribution yielding a pair of statements and witnesses in $\mathcal{R}$, assuming that $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$ is a hard-instance generator (as in Definition 7) is the same as assuming that $\mathcal{R}$ is a hard relation (with generator $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$, as in Definition 18 of [47]). To motivate Item 3, we remark that Lemma 8 guarantees that the static and adaptive notions are equivalent for proofs of knowledge. Regarding Item 4, we compare the unforgeability against no message attack of standard Fiat-Shamir signatures with the same property for Fiat-Shamir-with-Aborts signatures. We observe that the first implies the second in the natural scenario where proofs with $resp = \perp$ are not accepted. In other words, the possibility of abort only reduces the probability that an adversary can generate an accepting signature. To conclude with Item 5, we discuss how the parameters of Theorem 13 follow from the proof of Theorem 21. As a first step, Theorem 21 leverages Corollary 1 to argue that $\mathsf{FS}(\Sigma)$ is a proof of knowledge for adaptive adversaries, with parameters $p, d = 3$ as in Σ, and $e = 6, \mu(\eta) = \kappa(\eta) + \frac{1}{p(\eta)} \cdot \left(\frac{1}{8N_H^3 C^3} + \frac{1}{\mathcal{O}(N_H^5)C}\right)$. This implies their Equation 6. Taking the expectation in Equation 6 and using Jensen's inequality, they get the following:

$$\mathbb{E}_{x\leftarrow G}\left[\left(\Pr\left[x'\|m \in X \wedge (x', w) \in R : (x'\|m, w) \leftarrow \mathcal{E}^A\right] q^e p(\eta) + \mu(\eta)\right)^{1/d}\right]$$
$$\leq \left(\mathbb{E}_{x\leftarrow G} \Pr\left[x'\|m \in X \wedge (x', w) \in R : (x'\|m, w) \leftarrow \mathcal{E}^A\right] q^e p(\eta) + \mu(\eta)\right)^{1/d}.$$

Notably, the expected value in the right-hand side corresponds to ϵ_{HIG}. The theorem follows by converting the proof according to our notation, and by replacing the values p, d, e, μ and κ.

We now conclude the analysis with the proofs of Lemma 8 and Corollary 1.

Proof (Lemma 8). The only difference between Lemma 8 and Lemma 15 in [47] is that the first makes the parameters explicit. Thus, it suffices to focus on the parameters and, in particular, to show that they remain unchanged. This can be deducted by the proof of Lemma 15 (Appendix A of [47]). The parameters p, d, κ appearing there come directly from the assumption that Σ is a proof of knowledge for static adversaries (used in the first inequality), and remain the same throughout the proof. This happens because neither Jensen's inequality nor the compact representation of the probability impact the parameters.

Proof (Corollary 1). For the same reason discussed in the proof of Lemma 8, we only need to motivate the relations between parameters. Again, these relations naturally emerge from the proof of Corollary 16 in [47]. From the last equation of their proof, one can readily deduce that p and d are the same as in Σ, while $e = 2d$. On the other hand, the description of μ stays implicit. In the rest of the

proof, we provide a description for μ in the special case where $d = 3$. We can rewrite their second last equation by expanding the cube of a binomial, *i.e.*

$$\begin{aligned}
&\frac{1}{p(\eta)}\left(\frac{1}{\mathcal{O}(q^2)}\sum_{x_0\in X}\Pr_H\left[x = x_0 \wedge V_{FS}^H(x,\pi)\colon (x,\pi)\leftarrow\mathcal{A}^H\right] - \frac{1}{2q|C|}\right)^3 - \kappa(\eta)\\
&= \frac{1}{p(\eta)\mathcal{O}(q^6)}\left(\sum_{x_0\in X}\Pr_H\left[x = x_0 \wedge V_{FS}^H(x,\pi)\colon (x,\pi)\leftarrow\mathcal{A}^H\right]\right)^3\\
&\quad - \frac{1}{p(\eta)\mathcal{O}(q^5)|C|}\left(\sum_{x_0\in X}\Pr_H\left[x = x_0 \wedge V_{FS}^H(x,\pi)\colon (x,\pi)\leftarrow\mathcal{A}^H\right]\right)^2 \qquad (11)\\
&\quad + \frac{1}{p(\eta)\mathcal{O}(q^4)|C|^2}\left(\sum_{x_0\in X}\Pr_H\left[x = x_0 \wedge V_{FS}^H(x,\pi)\colon (x,\pi)\leftarrow\mathcal{A}^H\right]\right) \qquad (12)\\
&\quad - \frac{1}{8p(\eta)q^3|C|^3} - \kappa(\eta)\\
&\geq \frac{1}{p(\eta)\mathcal{O}(q^6)}\left(\sum_{x_0\in X}\Pr_H\left[x = x_0 \wedge V_{FS}^H(x,\pi)\colon (x,\pi)\leftarrow\mathcal{A}^H\right]\right)^3\\
&\quad - \left(\frac{1}{p(\eta)\mathcal{O}(q^5)|C|} + \frac{1}{8p(\eta)q^3|C|^3} + \kappa(\eta)\right),
\end{aligned}$$

where we get the last inequality by ignoring the positive factor of Eq. (12) and by upper-bounding the probability in Eq. (11) with 1. Moving back to our notation, where q is replaced with N_H and $|C|$ is replaced with C, the corollary follows.

C.6 Proof of Theorem 3

We consider the same hybrid experiments as in [14], depicted in Fig. 12. In particular, we use the fact, already proved in [14], that, for every quantum polynomial-time distinguisher D querying the signing oracle at most N_{Sign} times and the random oracle at most N_H times,

$$\left|\mathbb{P}\left[\mathsf{D}^{\mathbf{Sign},H} = 1\right] - \mathbb{P}\left[\mathsf{D}^{\mathbf{Prog},H}\right]\right| \leq \frac{2N_{\mathsf{Sign}}\sqrt{\epsilon_{\mathsf{ComGuess}}}}{1-\epsilon_{\mathsf{HAbort}}}\sqrt{N_H + \frac{N_{\mathsf{Sign}}}{1-\epsilon_{\mathsf{HAbort}}}} \qquad (13)$$

and

$$\left|\mathbb{P}\left[\mathsf{D}^{\mathbf{Prog},H} = 1\right] - \mathbb{P}\left[\mathsf{D}^{\mathbf{Trans},H}\right]\right| \leq 2N_H\sqrt{\frac{N_{\mathsf{Sign}}\epsilon_{\mathsf{ComGuess}}}{1-\epsilon_{\mathsf{HAbort}}}}, \qquad (14)$$

for every quantum polynomial-time adversary A querying the random oracle at most N_H times, since these hybrids do not depend on the zero-knowledge property.

Lemma 9. *For every quantum polynomial-time adversary* A *querying the random oracle at most* N_H *times,*

$$\left|\mathbb{P}\left[\mathsf{A}^{\mathbf{Trans},H} = 1\right] - \mathbb{P}\left[\mathsf{A}^{\mathbf{Sim},H} = 1\right]\right| \leq \epsilon_{\Sigma.\mathsf{HVZK}}.$$

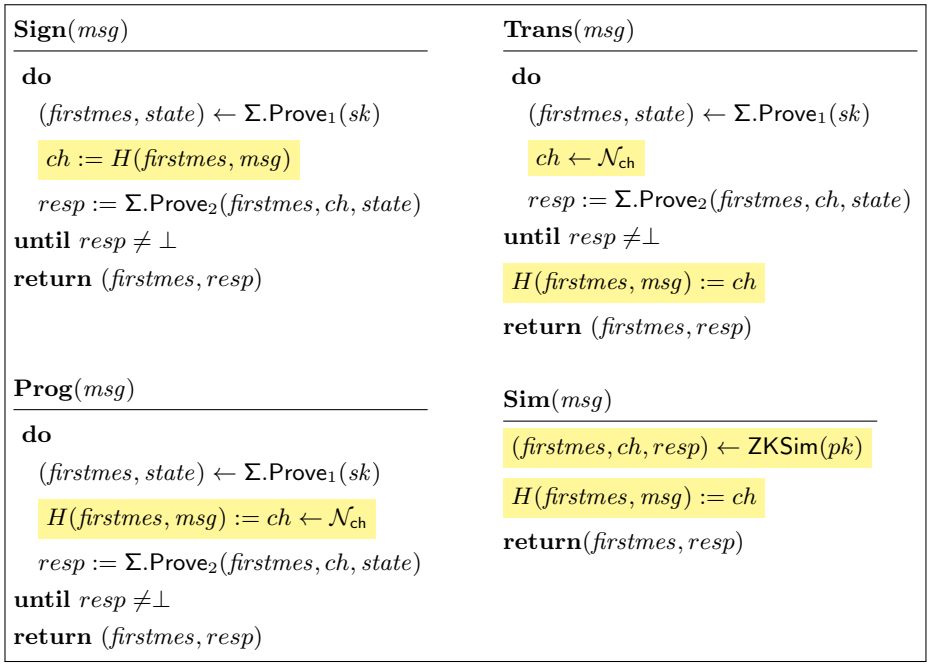

Fig. 12. The hybrid experiments used in the proof of Theorem 3.

Proof. We replace the oracles **Trans** and **Sim** with the oracles **Multi**$\langle\langle \mathbf{T} \rangle\rangle$ and **Multi**$\langle\langle \mathbf{S} \rangle\rangle$ depicted in Fig. 13. Notice that the only difference between **Trans** and **Multi**$\langle\langle \mathbf{T} \rangle\rangle$ is syntactic. Indeed, in both experiments the challenge ch is sampled independently of the message msg, and then the random oracle is reprogrammed using $firstmes, msg$. The only difference is that **Trans** lazily samples $firstmes, resp$ during every query, while **Multi**$\langle\langle \mathbf{T} \rangle\rangle$ samples everything at the beginning; hence, **Trans** $\equiv$ **Multi**$\langle\langle \mathbf{T} \rangle\rangle$. With the same reasoning, **Sim** $\equiv$ **Multi**$\langle\langle \mathbf{S} \rangle\rangle$. Finally, the only difference between **Multi**$\langle\langle \mathbf{T} \rangle\rangle$ and **Multi**$\langle\langle \mathbf{S} \rangle\rangle$ lies in how the transcripts are sampled. Namely, **Multi**$\langle\langle \mathbf{T} \rangle\rangle$ samples the transcripts as in the real setting, while **Multi**$\langle\langle \mathbf{S} \rangle\rangle$ invokes the zero-knowledge simulator. An easy reduction shows that, for every quantum polynomial-time adversary A querying the random oracle at most N_H times,

$$\begin{aligned}
&\left|\mathbb{P}\left[\mathsf{A}^{\mathbf{Trans},H}=1\right]-\mathbb{P}\left[\mathsf{A}^{\mathbf{Sim},H}=1\right]\right| \\
&=\left|\mathbb{P}\left[\mathsf{A}^{\mathbf{Multi}\langle\langle \mathbf{T} \rangle\rangle,H}=1\right]-\mathbb{P}\left[\mathsf{A}^{\mathbf{Multi}\langle\langle \mathbf{S} \rangle\rangle,H}=1\right]\right| \\
&\leq \epsilon_{\Sigma.\mathsf{HVZK}},
\end{aligned}$$

thus concluding the proof. □

The next lemma is a simplified version of [14, Lemma 1], in which we consider Γ to be the event that is always true.

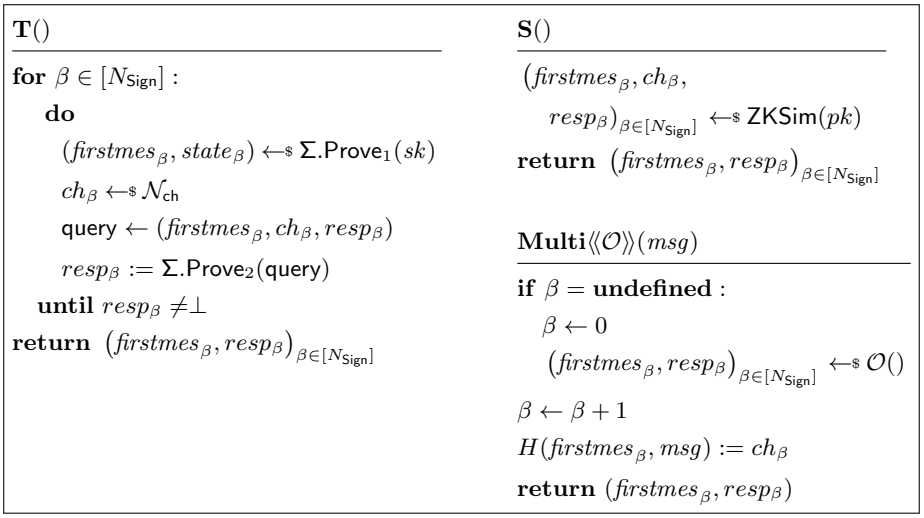

Fig. 13. The oracles **Multi**$\langle\langle \mathbf{T} \rangle\rangle$ and **Multi**$\langle\langle \mathbf{S} \rangle\rangle$ that compute every transcript at the beginning instead of on request by the adversary.

Lemma 10 (Lemma 1 of [14], adapted to Multi-HVZK and $\Gamma = \mathsf{true}$). *Let $\epsilon_{\mathsf{ComGuess}} \geq 0$ and $\epsilon_{\mathsf{HAbort}} < 1$. Let $\mathsf{A}^{\mathbf{Sign},H}_{N_{\mathsf{Sign}},N_H}$ be a quantum polynomial-time adversary CMA attacker against Σ that makes at most N_{Sign} queries to the signing oracle* **Sign** *and N_H queries to the random oracle H. Then, there exists a quantum polynomial-time adversary NMA attacker B^H against Σ and a pair of quantum polynomial-time distinguishers $\mathsf{D}^{\cdot,H}, \mathsf{D}'^{\cdot,H}$,*

$$\begin{aligned}\mathbb{P}\left[\mathsf{A}^{\mathbf{Sign},H}_{N_{\mathsf{Sign}},N_H} \ \mathit{wins}\right] \leq\ & \mathbb{P}\left[\mathsf{B}^H \ \mathit{wins}\right] \\ & + \left|\mathbb{P}\left[\mathsf{D}^{\mathbf{Sign},H}_{N_{\mathsf{Sign}},N_H+1} = 1\right] - \mathbb{P}\left[\mathsf{D}^{\mathbf{Prog},H}_{N_{\mathsf{Sign}},N_H+1} = 1\right]\right| \\ & + \left|\mathbb{P}\left[\mathsf{D}'^{\mathbf{Prog},H}_{N_{\mathsf{Sign}},N_H+1} = 1\right] - \mathbb{P}\left[\mathsf{D}'^{\mathbf{Trans},H}_{N_{\mathsf{Sign}},N_H+1} = 1\right]\right| \\ & + \epsilon_{\Sigma.\mathsf{HVZK}}.\end{aligned}$$

Proof (sketch). As in the original proof, we consider an adversary $\widetilde{\mathsf{A}}$ that, upon input the public key vk internally runs A, verifies the correctness of the produced forgery and outputs 1 if and only if it is a valid forgery (and 0 otherwise). Since $\widetilde{\mathsf{A}}$ requires an additional oracle query for the verification, the amount of oracle queries from $\widetilde{\mathsf{A}}$ is at most $N_H + 1$ and, moreover,

$$\mathbb{P}\left[\mathsf{A}^{\mathbf{Sign},H}_{N_{\mathsf{Sign}},N_H} \ \mathrm{wins}\right] = \mathbb{E}\left[\mathbb{P}\left[\widetilde{\mathsf{A}}^{\mathbf{Sign},H}_{N_{\mathsf{Sign}},N_H+1} = 1\right]\right].$$

We can upper bound the guessing probability of $\widetilde{\mathsf{A}}$ with

$$\mathbb{P}\left[\widetilde{\mathsf{A}}^{\mathbf{Sign},H}_{N_{\mathsf{Sign}},N_H+1} = 1\right] \leq \mathbb{P}\left[\widetilde{\mathsf{A}}^{\mathbf{Sim},H}_{N_{\mathsf{Sign}},N_H+1} = 1\right]$$

$$+\left|\mathbb{P}\left[\widetilde{\mathsf{A}}^{\mathbf{Sign},H}_{N_{\mathsf{Sign}},N_H+1}=1\right]-\mathbb{P}\left[\widetilde{\mathsf{A}}^{\mathbf{Prog},H}_{N_{\mathsf{Sign}},N_H+1}=1\right]\right|$$
$$+\left|\mathbb{P}\left[\widetilde{\mathsf{A}}^{\mathbf{'Prog},H}_{N_{\mathsf{Sign}},N_H+1}=1\right]-\mathbb{P}\left[\widetilde{\mathsf{A}}^{\mathbf{'Trans},H}_{N_{\mathsf{Sign}},N_H+1}=1\right]\right|$$
$$+\left|\mathbb{P}\left[\widetilde{\mathsf{A}}^{\mathbf{'Trans},H}_{N_{\mathsf{Sign}},N_H+1}=1\right]-\mathbb{P}\left[\widetilde{\mathsf{A}}^{\mathbf{'Sim},H}_{N_{\mathsf{Sign}},N_H+1}=1\right]\right|$$

and, by applying Lemma 9 and averaging over the choice of vk, there exists a pair of quantum polynomial-time distinguishers $\mathsf{D}^{\cdot,H},\mathsf{D}'^{\cdot,H}$ such that

$$\begin{aligned}\mathbb{P}\left[\mathsf{A}^{\mathbf{Sign},H}_{N_{\mathsf{Sign}},N_H}\ \text{wins}\right]\leq\ &\mathbb{P}\left[\widetilde{\mathsf{A}}^{\mathbf{Sim},H}_{N_{\mathsf{Sign}},N_H+1}=1\right]\\&+\left|\mathbb{P}\left[\mathsf{D}^{\mathbf{Sign},H}_{N_{\mathsf{Sign}},N_H+1}=1\right]-\mathbb{P}\left[\mathsf{D}^{\mathbf{Prog},H}_{N_{\mathsf{Sign}},N_H+1}=1\right]\right|\\&+\left|\mathbb{P}\left[\mathsf{D}'^{\mathbf{Prog},H}_{N_{\mathsf{Sign}},N_H+1}=1\right]-\mathbb{P}\left[\mathsf{D}'^{\mathbf{Trans},H}_{N_{\mathsf{Sign}},N_H+1}=1\right]\right|\\&+\epsilon_{\Sigma.\mathsf{HVZK}}.\end{aligned}$$

Recall that, for a random key pair (vk,sk), $\widetilde{\mathsf{A}}^{\mathbf{Sim},H}_{N_{\mathsf{Sign}},N_H+1}$ runs A against the CMA game in which all the signing queries are answered by **Sim** instead of **Sign**, which means that no signature queries are actually performed. This implies that there exists an adversary $\mathsf{B}^{H}_{N_{\mathsf{Sign}},N_H}$ with the same success probability against the NMA game. The lemma follows. □

Finally, Theorem 3 follows by combining Eqs. (13) and (14) and Lemma 10.

C.7 Proof of Theorem 9

Our scheme meets all the assumptions of Theorem 1, with

$$\begin{aligned}\epsilon_{\Sigma.\mathsf{Comp}}&=tm\cdot 2^{-\alpha}+t\cdot\epsilon_{\mathsf{PRG},\mathcal{K}}+\epsilon_{\mathsf{LWR}},\\\epsilon_{\mathsf{HAbort}}&=tm\cdot 2^{-\alpha}+t\cdot\epsilon_{\mathsf{PRG},\mathcal{K}}+\epsilon_{\mathsf{LWR}},\\\nu&=-\log\left[\left(2\epsilon_{\mathsf{Coll}}\left(1-\frac{1}{|\mathcal{P}_{\mathcal{K}}|}\right)+\frac{1}{|\mathcal{P}_{\mathcal{K}}|}\right)^{(d-1)t}\right],\end{aligned}$$

according to, respectively, Theorem 4, Theorem 14, and Theorem 8. By plugging these parameters into Theorem 1, Theorem 9 follows.

C.8 Proof of Theorem 10

Combining the results of Fehr *et al.* [47] and Barbosa *et al.* [14] (reported in Section 3.6), we get

$$\begin{aligned}\epsilon_{\mathsf{Sign},N_{\mathsf{Sign}}}\leq&\left(\epsilon_{\mathsf{HIG}}\cdot N_H^6+\frac{1}{C}+2\epsilon_{\mathsf{QCUR}}+\frac{1}{O(N_H^5)\cdot C}+\frac{1}{8N_H^3C^3}\right)^{\frac{1}{3}}\\&+\frac{2N_{\mathsf{Sign}}\sqrt{\epsilon_{\mathsf{ComGuess}}}}{1-\epsilon_{\mathsf{HAbort}}}\sqrt{N_H+1+\frac{N_{\mathsf{Sign}}}{1-\epsilon_{\mathsf{HAbort}}}}\end{aligned}$$

$$+ 2(N_H + 1)\sqrt{\frac{N_{\mathsf{Sign}} \cdot \epsilon_{\mathsf{ComGuess}}}{1 - \epsilon_{\mathsf{HAbort}}}} + N_{\mathsf{Sign}}\epsilon_{\Sigma.\mathsf{HVZK}}.$$

From Lemma 3, we know that

$$\epsilon_{\mathsf{HIG}} \leq \epsilon_{\mathsf{LWR}}^{(p-\alpha)} \left(1 + 3^m \frac{2^{qn}}{2^{(p-\alpha)m}}\right) + 3^m \frac{2^{qn}}{2^{2qnN}}.$$

Our sigma protocol has challenge size $C = d^t$, and from Section 5 we derive the parameters

$$\begin{aligned}
&\epsilon_{\mathsf{HAbort}} \leq tm \cdot 2^{-\alpha} + t \cdot \epsilon_{\mathsf{PRG},\mathcal{K}} + \epsilon_{\mathsf{LWR}},\\
&\epsilon_{\Sigma.\mathsf{HVZK}} = Nt\epsilon_{\mathsf{Hide}} + \frac{2Nt\epsilon_{\mathsf{PRG}} + 4\epsilon_{\mathsf{LWR}}}{1 - N \cdot t \cdot m \cdot 2^{-\alpha}} + 2\gamma^{\omega},\\
&\epsilon_{\mathsf{QCUR}} \leq t(d-1) \cdot \epsilon_{\mathsf{Coll}},\\
&\epsilon_{\mathsf{ComGuess}} \leq \left(2\epsilon_{\mathsf{Coll}}\left(1 - \frac{1}{|\mathcal{P}_{\mathcal{K}}|}\right) + \frac{1}{|\mathcal{P}_{\mathcal{K}}|}\right)^{(d-1)t}.
\end{aligned}$$

The lemma follows.

D Properties of the Rounding Function

In this section, we introduce simple properties of the rounding function that we use throughout our proofs.

Proposition 1 (Generic properties of the rounding function). *Let Q, P be integers such that $Q > P$, and let $x \in \mathbb{Z}_Q$. Then the following properties hold*

1. *Let $P' \in \mathbb{N}$ such that $P'|P$. Moreover, assume that $P|Q$. Then*
$$\lfloor x \rceil_{P'} = \lfloor \lfloor x \rceil_P \rceil_{P'}.$$
2. *If $P|Q$, then*
$$x = \frac{Q}{P}\lfloor x \rceil_P + \left(x \mod \frac{Q}{P}\right).$$
3. *Let $x' \in \mathbb{Z}_Q$, and assume that $P|Q$. Then*
$$\lfloor x + x' \rceil_P - \lfloor x \rceil_P - \lfloor x' \rceil_P \in \{0, 1\}.$$

Proposition 2 (Properties of the rounding functions over bit strings). *Let q, p be integers such that $q > p$. Then, the following properties hold.*

1. (MSB notation) *For all $x \in \mathbb{Z}_{2^q}$,*
$$\lfloor x \rceil_{2^p} = x[(q-p)..q].$$

2. (LSB notation) *For all* $x \in \mathbb{Z}_{2^q}$,

$$2^{q-p} \lfloor x \rfloor_{2^p} = x - x[0..(q-p)],$$

or, equivalently,

$$x = 2^{q-p} \lfloor x \rfloor_{2^p} + x[0..(q-p)].$$

3. (Slices of slices equivalence) *Let* $i, j, k, h \in \mathbb{N}$ *be such that* $k < h \leq q$ *and* $i < j \leq h - k$. *Then, for all* $x \in \mathbb{Z}_{2^q}$,

$$x[k..h][i] = x[k+i]$$

and

$$x[k..h][i..j] = x[(k+i)..(k+j)].$$

4. (Monotonicity of rounding) *Let* $i, j \in \mathbb{N}$ *be such that* $i < j \leq q$. *Then, for all* $u, v \in \mathbb{Z}_{2^q}$,
   - *If* $u[i..j] < v[i..j]$, *then* $u[0..j] < v[0..j]$.
   - *If* $u[i..j] > v[i..j]$, *then* $u[0..j] > v[0..j]$.

In what follows, we report the proofs of both propositions.

Proof (Proposition 1). We start with the proof of Item 1. We first show the inequality $\lfloor \lfloor x \rfloor_P \rfloor_{P'} \leq \lfloor x \rfloor_{P'}$, and then the same in the opposite direction, *i.e.* $\lfloor x \rfloor_{P'} \leq \lfloor \lfloor x \rfloor_P \rfloor_{P'}$, which implies that the equality holds. We start with the first inequality:

$$\left\lfloor \left\lfloor x \cdot \frac{P}{Q} \right\rfloor \cdot \frac{P'}{P} \right\rfloor \leq \left\lfloor x \cdot \frac{P}{Q} \cdot \frac{P'}{P} \right\rfloor = \left\lfloor x \cdot \frac{P'}{Q} \right\rfloor = \lfloor x \rfloor_{P'}.$$

We now prove the second inequality. First, we remark that $P' | P$ and $P | Q$ imply that there exist c_1, c_2, c_3 such that $\frac{Q}{P'} = c_1, \frac{P}{P'} = c_2, \frac{Q}{P} = c_3$ and $c_1 = c_2 \cdot c_3$. Now, consider the Euclidean division of x by c_1, and name the quotient M_1 and the remainder R_1, *i.e.* $x = M_1 \cdot c_1 + R_1$ with $M_1, R_1 \in \mathbb{N}$ and $R_1 < c_1$. Then

$$\lfloor x \rfloor_{P'} = \left\lfloor \frac{M_1 \cdot c_1 + R_1}{c_1} \right\rfloor = M_1, \tag{15}$$

and

$$\begin{aligned} \lfloor \lfloor x \rfloor_P \rfloor_{P'} &= \left\lfloor \left\lfloor \frac{M_1 \cdot c_1 + R_1}{c_3} \right\rfloor \cdot \frac{1}{c_2} \right\rfloor \\ &\geq \left\lfloor \left\lfloor \frac{M_1 \cdot c_1}{c_3} \right\rfloor \cdot \frac{1}{c_2} \right\rfloor \\ &= \left\lfloor \lfloor M_1 \cdot c_2 \rfloor \cdot \frac{1}{c_2} \right\rfloor = \left\lfloor M_1 \cdot c_2 \cdot \frac{1}{c_2} \right\rfloor = M_1 \end{aligned}$$

$$= \lfloor x \rfloor_{P'} . \tag{16}$$

where Eq. (16) follows from Eq. (15). This proves Item 1.

We now prove Item 2. Consider the Euclidean division of x by $\frac{Q}{P}$, and name M the quotient and R the rest, *i.e.* $x = M \cdot \frac{Q}{P} + R$ with $M, R \in \mathbb{N}$ and $R < \frac{Q}{P}$. Then

$$\begin{aligned} x \mod \frac{Q}{P} &= R, \\ \lfloor x \rfloor_P &= \left\lfloor M \cdot \frac{Q}{P} \cdot \frac{P}{Q} + R \cdot \frac{P}{Q} \right\rfloor = M + \left\lfloor R \cdot \frac{P}{Q} \right\rfloor = M. \end{aligned}$$

To conclude, we prove Item 3. According to Item 2, we can rewrite

$$x + x' = \frac{Q}{P} \lfloor x \rfloor_P + \frac{Q}{P} \lfloor x' \rfloor_P + \left(x \mod \frac{Q}{P} \right) + \left(x' \mod \frac{Q}{P} \right).$$

By applying the rounding on both sides, we get

$$\begin{aligned} \lfloor x + x' \rfloor_P &= \left\lfloor \frac{Q}{P} \lfloor x \rfloor_P + \frac{Q}{P} \lfloor x' \rfloor_P + \left(x \mod \frac{Q}{P} \right) + \left(x' \mod \frac{Q}{P} \right) \right\rfloor_P \\ &= \lfloor x \rfloor_P + \lfloor x' \rfloor_P + \left\lfloor \left(x \mod \frac{Q}{P} \right) + \left(x' \mod \frac{Q}{P} \right) \right\rfloor_P \end{aligned} \tag{17}$$

where Eq. (17) follows from the definition of rounding and from the fact that $\lfloor x \rfloor_P$ and $\lfloor x' \rfloor_P$ are integers. To conclude the proof, it suffices to show that

$$\left\lfloor \left(x \mod \frac{Q}{P} \right) + \left(x' \mod \frac{Q}{P} \right) \right\rfloor_P \in \{0, 1\}.$$

This is true because every $x, x' \in \mathbb{Z}_Q$ satisfies

$$0 \leq \left(x \mod \frac{Q}{P} \right), \left(x' \mod \frac{Q}{P} \right) \leq \frac{Q}{P} - 1,$$

and therefore

$$0 \leq \left(x \mod \frac{Q}{P} \right) + \left(x' \mod \frac{Q}{P} \right) \leq 2\frac{Q}{P} - 2. \tag{18}$$

Moving to the rounding, Eq. (18) yields

$$0 \leq \left\lfloor \left(x \mod \frac{Q}{P} \right) + \left(x' \mod \frac{Q}{P} \right) \right\rfloor_P \leq \left\lfloor 2\frac{Q}{P} - 2 \right\rfloor_P \leq 1. \tag{19}$$

This concludes the proof.

Proof (Proposition 2). We start with Item 1. We rewrite $\lfloor x \rfloor_{2^p}$ as

$$\lfloor x \rfloor_{2^p} = \left\lfloor \sum_{i=0}^{q-1} 2^i x[i] \right\rfloor_{2^p} \qquad \text{(binary representation)},$$

$$
\begin{aligned}
&= \left\lfloor \sum_{i=0}^{q-1} 2^i x[i] \cdot \frac{2^p}{2^q} \right\rfloor && \text{(by definition of rounding)},\\
&= \left\lfloor \sum_{i=0}^{q-1} 2^{i-(q-p)} x[i] \right\rfloor && \text{(simplifying the fraction)},\\
&= \sum_{i=q-p}^{q-1} 2^{i-(q-p)} x[i] && \text{(rounding in } \mathbb{Z}),\\
&= x[(q-p)..q] && \text{(binary representation)},
\end{aligned}
$$

where the rounding in $\mathbb{Z}$ happens because

$$\sum_{i=q-p}^{q-1} 2^{i-(q-p)} x[i]$$

is an integer and

$$\sum_{i=0}^{q-p-1} 2^{i-(q-p)} x[i] < 1.$$

We now move to Item 2. We rewrite $x - x[0..(q-p)]$ as

$$
\begin{aligned}
x - x[0..(q-p)] &= \sum_{i=0}^{q-1} 2^i x[i] - \sum_{i=0}^{q-p-1} 2^i x[i] && \text{(binary representation)},\\
&= \sum_{i=q-p}^{q-1} 2^i x[i] && \text{(simplifying the sum)},\\
&= 2^{q-p} \sum_{i=q-p}^{q-1} 2^{i-q-p} x[i] && \text{(moving } 2^{q-p} \text{ outside)},\\
&= 2^{q-p} x[(q-p)..q] && \text{(binary representation)},\\
&= 2^{q-p} \lfloor x \rfloor_{2^p} && \text{(by Item 1)}.
\end{aligned}
$$

Next, we prove Item 3. We can rewrite $x[k..h][i]$ as

$$x[k..h][i] = \left(\sum_{\eta=k}^{h-1} 2^{\eta-k} x[\eta] \right)[i] = x[\eta^*]$$

for η^* such that $\eta^* - k = i$, hence $x[\eta^*] = x[k+i]$, which proves the first equality. Now we can use this to prove the second equality:

$$x[k..h][i..j] = \sum_{\eta=i}^{j-1} 2^{\eta-i} x[k..h][\eta] \qquad \text{(binary representation)},$$

$$
\begin{aligned}
&= \sum_{\eta=i}^{j-1} 2^{\eta-i} x[k+\eta] && \text{(by the first equality)},\\
&= \sum_{\eta=k+i}^{k+j-1} 2^{\eta-k-i} x[\eta] && \text{(reparametrization of the sum index)},\\
&= x[(k+i)..(k+j)] && \text{(binary representation)}.
\end{aligned}
$$

We now conclude with the proof of Item 4. We only prove the first statement, since the proof for the second one is identical. First, we rewrite $2^i u[i..j]$ as

$$
\begin{aligned}
2^i u[i..j] &= 2^i u[0..j][i..j] && \text{(by Item 3)},\\
&= 2^i \lfloor u[0..j] \rfloor_{2^{j-i}} && \text{(by Item 1)},\\
&= u[0..j] - u[0..i] && \text{(by Item 2)},
\end{aligned}
$$

and, similarly, $2^i v[i..j] = v[0..j] - v[0..i]$. Now, assume that $u[i..j] < v[i..j]$ or, since both are integers, $u[i..j] + 1 \leq 2^i v[i..j]$. Then,

$$
\begin{aligned}
u[0..j] &= u[0..j] - u[0..i] + u[0..i] && \text{(by adding and subtracting } u[0..i])\\
&= 2^i u[i..j] + u[0..i] && \text{(by applying the above)}\\
&< 2^i u[i..j] + 2^i && \text{(by } u[0..i] < 2^i)\\
&= 2^i \left(u[i..j] + 1\right) && \text{(distributive property)}\\
&\leq 2^i v[i..j] && \text{(by } u[i..j] + 1 \leq 2^i v[i..j])\\
&= v[0..j] - v[0..i] && \text{(by applying the above again)}\\
&\leq v[0..j] && \text{(by } v[0..i] \geq 0).
\end{aligned}
$$

E Security of the Sigma Protocol in Section 5

In this section, we formally prove all the properties of the sigma protocol. Before that, we need to introduce a preliminary result.

Theorem 14. *Let Σ be the Sigma protocol from Fig. 3, and $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$ be the distribution described in Fig. 4. Assume that decisional LWR holds with parameters $(2^q, 2^p, n, m)$ and advantage ϵ_{LWR}, and that $G_{\mathcal{K}}$ is a $\epsilon_{\mathsf{PRG},\mathcal{K}}$-secure PRG. Then, the probability $\epsilon_{\mathsf{HAbort}}$ that* Abort *outputs* 1 *in an honest execution of Σ satisfies*

$$\epsilon_{\mathsf{HAbort}} \leq tm \cdot 2^{-\alpha} + t \cdot \epsilon_{\mathsf{PRG},\mathcal{K}} + \epsilon_{\mathsf{LWR}}.$$

We prove the above via a hybrid argument. We report the experiments of interest in Fig. 14, and shortly summarize them below.

HonestAbort**:** This experiment describes the event where Abort outputs 1 during an honest execution of the protocol. Notably, this experiment ignores the response and verification, as we are only interested in the output of Abort.

HonestAbort

$((\boldsymbol{X}, \boldsymbol{y}), \boldsymbol{k}) \leftarrow\!\!\$\ \mathcal{D}_{\mathcal{X}\times\mathcal{W}}$
$(\mathit{firstmes}, \mathit{state}) \leftarrow\!\!\$\ \Sigma.\mathsf{Prove}_1((\boldsymbol{X}, \boldsymbol{y}), \boldsymbol{k})$
$ch \leftarrow\!\!\$\ \mathcal{N}_{\mathsf{ch}}$
if $\mathsf{Abort}(\mathit{state}, ch)$
  return 1
return 0

$\mathsf{Hyb}^{(1)}$

$((\boldsymbol{X}, \boldsymbol{y}), \boldsymbol{k}) \leftarrow\!\!\$\ \mathcal{D}_{\mathcal{X}\times\mathcal{W}}$
$ch \leftarrow\!\!\$\ \mathcal{N}_{\mathsf{ch}}$
$\mathit{state}' \leftarrow\!\!\$\ \Sigma.\mathsf{Prove}'_1((\boldsymbol{X}, \boldsymbol{y}), \boldsymbol{k}, ch)$
if $\mathsf{Abort}'(\mathit{state}')$
  return 1
return 0

$\mathsf{Hyb}^{(2)}_{j^*}$

$((\boldsymbol{X}, \boldsymbol{y}), \boldsymbol{k}) \leftarrow\!\!\$\ \mathcal{D}_{\mathcal{X}\times\mathcal{W}}$
$ch \leftarrow\!\!\$\ \mathcal{N}_{\mathsf{ch}}$
$\mathit{state}' \leftarrow\!\!\$\ \Sigma.\mathsf{Prove}''_{1,j^*}((\boldsymbol{X}, \boldsymbol{y}), \boldsymbol{k}, ch)$
if $\mathsf{Abort}'(\mathit{state}')$
  return 1
return 0

$\mathsf{Hyb}^{(3)}$

$((\boldsymbol{X}, \boldsymbol{y}), \boldsymbol{k}) \leftarrow\!\!\$\ \mathcal{D}'_{\mathcal{X}\times\mathcal{W}}$
$ch \leftarrow\!\!\$\ \mathcal{N}_{\mathsf{ch}}$
$\mathit{state}' \leftarrow\!\!\$\ \Sigma.\mathsf{Prove}''_{1,t}((\boldsymbol{X}, \boldsymbol{y}), \boldsymbol{k}, ch)$
if $\mathsf{Abort}'(\mathit{state}')$
  return 1
return 0

$\Sigma.\mathsf{Prove}'_1((\boldsymbol{X}, \boldsymbol{y}), \boldsymbol{k}, ch)$

$(\mathcal{I}^{(j)})_{j\in[t]} \leftarrow ch$
for $j \in [t]$
  $(\boldsymbol{k}_1^{(j)}, \ldots, \boldsymbol{k}_d^{(j)}) \leftarrow\!\!\$\ \mathsf{Share}(\boldsymbol{k})$
  for $i \in [d]$
    $\boldsymbol{z}_i^{(j)} \leftarrow \mathsf{Output}(\boldsymbol{X}, \boldsymbol{k}_i^{(j)}, i)$
  $\mathit{sum}^{(j)} \leftarrow \sum_{i\in\mathcal{I}^{(j)}} \boldsymbol{z}_i^{(j)}$
$\boldsymbol{y}_{ext} \leftarrow \lfloor \boldsymbol{X}\boldsymbol{k} \rfloor_{2^p}$
$\mathit{state}' \leftarrow (\boldsymbol{y}_{ext}, (\mathit{sum}^{(j)})_{j\in[t]})$
return state'

$\Sigma.\mathsf{Prove}''_{1,j^*}((\boldsymbol{X}, \boldsymbol{y}), \boldsymbol{k}, ch)$

$(\mathcal{I}^{(j)})_{j\in[t]} \leftarrow ch$
for $j \in [t]$
  if $j \le j^*$
    $\tilde{\boldsymbol{k}}^{(j)} \leftarrow\!\!\$\ \mathbb{Z}_Q^n$
    $\mathit{sum}^{(j)} \leftarrow \boldsymbol{X}\tilde{\boldsymbol{k}}^{(j)}$
  else
    $(\boldsymbol{k}_1^{(j)}, \ldots, \boldsymbol{k}_d^{(j)}) \leftarrow\!\!\$\ \mathsf{Share}(\boldsymbol{k})$
    for $i \in [d]$
      $\boldsymbol{z}_i^{(j)} \leftarrow \mathsf{Output}(\boldsymbol{X}, \boldsymbol{k}_i^{(j)}, i)$
    $\mathit{sum}^{(j)} \leftarrow \sum_{i\in\mathcal{I}^{(j)}} \boldsymbol{z}_i^{(j)}$
$\boldsymbol{y}_{ext} \leftarrow \lfloor \boldsymbol{X}\boldsymbol{k} \rfloor_{2^p}$
$\mathit{state}' \leftarrow (\boldsymbol{y}_{ext}, (\mathit{sum}^{(j)})_{j\in[t]})$
return state'

$\mathsf{Abort}'(\mathit{state}')$

$(\boldsymbol{y}_{ext}, (\mathit{sum}^{(j)})_{j\in[t]}) \leftarrow \mathit{state}'$
for $j \in [t]$
  if $0 \in \mathit{sum}^{(j)}[r_S..r_E] - \boldsymbol{y}_{ext}[0..\alpha]$
    return 1
return 0

$\mathcal{D}'_{\mathcal{X}\times\mathcal{W}}$

$\boldsymbol{k} \leftarrow\!\!\$\ \mathbb{Z}_{2q}^n$
$\boldsymbol{X} \leftarrow\!\!\$\ \mathbb{Z}_Q^{m\times n}$
$\boldsymbol{y} \leftarrow\!\!\$\ \mathbb{Z}_{2^{p-\alpha}}^m$
return $((\boldsymbol{X}, \boldsymbol{y}), \boldsymbol{k})$

Fig. 14. The hybrid experiments for the proof of Theorem 14.

$\mathsf{Hyb}^{(1)}$**:** This is a variant of $\mathsf{HonestAbort}$ where we tailor the prover's description to only compute the values determining the abort condition. Specifically, we consider an alternative prover $\Sigma.\mathsf{Prove}'_1$ that, instead of passing to the abort algorithm a state *state* including all the key shares $(\boldsymbol{k}_i^{(j)})_{i,j}$ and the matrix $\boldsymbol{X}$, only passes a state *state*$'$ containing the sums $(sum^{(j)})_{j\in[t]}$. Since this value depends on the challenge *ch*, $\Sigma.\mathsf{Prove}'_1$ takes it as an extra input. Accordingly, we define an alternative algorithm, Abort', which no longer computes the values $(sum^{(j)})_{j\in[t]}$ internally, but instead takes them directly as part of the input.

$(\mathsf{Hyb}^{(2)}_{j^*})_{j^*\in[t]}$**:** These experiments are identical to $\mathsf{Hyb}^{(1)}$, except that the first j^* partial sums are generated in a key-independent way.

$\mathsf{Hyb}^{(3)}$**:** This is a variant of $\mathsf{Hyb}^{(2)}$ where statement-witness pair is generated with an alternative algorithm $\mathcal{D}'_{\mathcal{X}\times\mathcal{W}}$. The only difference between $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$ and $\mathcal{D}'_{\mathcal{X}\times\mathcal{W}}$ is that the latter samples $\boldsymbol{y}$ at random, independently of $\boldsymbol{X}$ and $\boldsymbol{k}$.

The proof proceeds via a sequence of hybrid arguments, and follows from the claims below.

Claim 4 $\mathsf{HonestAbort} \equiv \mathsf{Hyb}^{(1)}$.

Proof (Claim 4). The main observation of this proof is that the output of Abort only depends on the extended output $\boldsymbol{y}_{ext}$ and the partial sums $(sum^{(j)})_j$. Therefore, omitting the computation of the rest does not affect the output of Abort. Aside from this, the only difference between $\mathsf{HonestAbort}$ and $\mathsf{Hyb}^{(1)}$ is the order of computation. However, since the information used by Abort stays the same, so does the output. The claim follows.

Claim 5 $\mathsf{Hyb}^{(1)} \equiv \mathsf{Hyb}^{(2)}_0$.

Proof (Claim 5). This is true because $\Sigma.\mathsf{Prove}'_1$ and $\Sigma.\mathsf{Prove}''_{1,j^*}$ are the same algorithm when $j^* = 0$.

Claim 6 *Assume that $G_{\mathcal{K}}$ is a $\epsilon_{\mathsf{PRG},\mathcal{K}}$-secure pseudorandom generator. Then*

$$\mathbb{P}\left[\mathsf{Hyb}^{(2)}_{j^*-1} = 1\right] - \mathbb{P}\left[\mathsf{Hyb}^{(2)}_{j^*} = 1\right] \leq \epsilon_{\mathsf{PRG},\mathcal{K}}.$$

Proof (Claim 6). This proof works via a reduction to the pseudorandomness of $G_{\mathcal{K}}$. The only difference between the two experiments concerns the partial sum $sum^{(j^*)}$. In the first case, it is computed as

$$sum^{(j^*)} = \sum_{i\in\mathcal{I}^{(j)}} \boldsymbol{z}_i^{(j)} = \boldsymbol{X}\left(\sum_{i\in[d-1]\cap\mathcal{I}^{(j^*)}} G_{\mathcal{K}}(\boldsymbol{k}_i) + \sum_{i\in\{d\}\cap\mathcal{I}^{(j^*)}} \boldsymbol{k}_i\right), \tag{20}$$

whereas in the second, it's just the multiplication of $\boldsymbol{X}$ with a fresh random vector $\tilde{\boldsymbol{k}}^{(j^*)}$. The main remark here is that $sum^{(j^*)}$ in $\mathsf{Hyb}^{(2)}_{j^*-1}$ is distributed

either as the shifted multiplication of $\boldsymbol{X}$ with a pseudorandom value, or as the multiplication of $\boldsymbol{X}$ with a sum of pseudorandom values. Moreover, this can be determined based on $\mathcal{I}^{(j^*)}$. In the next few lines, we make this observation formal. First, assume that $d \in \mathcal{I}^{(j^*)}$. Then we can use the description of $\boldsymbol{k}_d^{(j)}$ in Eq. (20) and get

$$\begin{aligned} sum^{(j^*)} &= \boldsymbol{X}\left(\sum_{i\in[d-1]\cap\mathcal{I}^{(j^*)}} G_{\mathcal{K}}(\boldsymbol{k}_i) + \boldsymbol{k}_d\right) \\ &= \boldsymbol{X}\left(\sum_{i\in[d-1]\cap\mathcal{I}^{(j^*)}} G_{\mathcal{K}}(\boldsymbol{k}_i) + \boldsymbol{k} - \sum_{i\in[d-1]} G_{\mathcal{K}}(\boldsymbol{k}_i)\right) \\ &= -\boldsymbol{X}G_{\mathcal{K}}(\boldsymbol{k}_{i^*}) + \boldsymbol{X}\boldsymbol{k}, \end{aligned} \tag{21}$$

where i^* is defined as $[t] \setminus \mathcal{I}^{(j)}$. Now, consider the case where $d \notin \mathcal{I}^{(j)}$. Then

$$sum^{(j^*)} = \boldsymbol{X}\sum_{i\in[d-1]} G_{\mathcal{K}}(\boldsymbol{k}_i). \tag{22}$$

With this intuition in mind, we can construct a reduction to the pseudorandomness of $G_{\mathcal{K}}$. We consider a reduction $\mathsf{A}_{\mathsf{PRG}}$ that takes as input a value $\tilde{\boldsymbol{k}} \in \mathbb{Z}_Q^n$, and behaves as in Fig. 15. In a nutshell, $\mathsf{A}_{\mathsf{PRG}}$ does the same as both hybrids in every iteration $j \neq j^*$, and the key point is the way it samples $sum^{(j^*)}$. If $d \in \mathcal{I}^{(j^*)}$, then $sum^{(j)} = \boldsymbol{X}\tilde{\boldsymbol{k}}$, and otherwise $sum^{(j)} = \boldsymbol{X}\left(\tilde{\boldsymbol{k}} + \sum_{i\in[d-2]} G_{\mathcal{K}}(\boldsymbol{k}_i^{(j)})\right)$. Notably, if $\tilde{\boldsymbol{k}}$ is pseudorandom, this yields a perfect simulation of $\mathsf{Hyb}_{j^*-1}^{(2)}$ according to Eq. (21) and Eq. (22). On the other hand, if $\tilde{\boldsymbol{k}}$ is purely random, then both ways of computing $sum^{(j^*)}$ correspond to the multiplication of $\boldsymbol{X}$ with a random vector. This is because in both cases $sum^{(j)} = \boldsymbol{X}\tilde{\boldsymbol{k}}'$, where $\tilde{\boldsymbol{k}}'$ is the sum of a random vector with independent objects, and therefore is still random. This proves that $\mathsf{A}_{\mathsf{PRG}}$ perfectly simulates both hybrids, and proves the claim.

Claim 7 *Assume that decisional LWR holds with parameters* $(2^q, 2^p, n, m)$ *and advantage* ϵ_{LWR}. *Then*

$$\mathbb{P}\left[\mathsf{Hyb}_t^{(2)} = 1\right] - \mathbb{P}\left[\mathsf{Hyb}^{(3)} = 1\right] \leq \epsilon_{\mathsf{LWR}}.$$

Proof (Claim 7). This proof is a reduction to the one-time pseudorandomness of the wPRF F defined in *Section* 4.1. We can reduce to the wPRF F for three main reasons. First, because $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$ and $\mathcal{D}'_{\mathcal{X}\times\mathcal{W}}$ generate the statement and witness, respectively, as the real and ideal experiments for F generate their input, output, and key. Second, because the Abort' condition does not depend on the key $\boldsymbol{k}$ directly, but only on the extended wPRF output $\boldsymbol{y}_{ext} \in \mathbb{Z}_{2^p}^m$ and the partial sums $(sum^{(j)})_j$. Finally, because $\Sigma.\mathsf{Prove}''_{1,t}$ generates the partial sums independently of $\boldsymbol{k}$. In other words, the only input to Abort' which could be related to $\boldsymbol{k}$ is $\boldsymbol{y}_{ext}$, which either equals $\lfloor \boldsymbol{X}\boldsymbol{k} \rfloor_{2^p}$, or is an independent random

$\mathsf{A}_{\mathsf{PRG}}(\tilde{\boldsymbol{k}})$

$((\boldsymbol{X},\boldsymbol{y}),\boldsymbol{k}) \leftarrow\!\!\$\ \mathcal{D}_{\mathcal{X}\times\mathcal{W}}$,
for $j \in [t]$
  $i^* \leftarrow\!\!\$\ [d]$
  $\mathcal{I}^{(j)} \leftarrow [d] \setminus \{i^*\}$
  if $j \leq j^* - 1$
    $\tilde{\boldsymbol{k}}^{(j)} \leftarrow\!\!\$\ \mathbb{Z}_Q^n$
    $sum^{(j)} \leftarrow \boldsymbol{X}\tilde{\boldsymbol{k}}^{(j)}$
  if $j = j^*$
    if $d \in \mathcal{I}^{(j)}$
      $sum^{(j)} \leftarrow \boldsymbol{X}(\boldsymbol{k} - \tilde{\boldsymbol{k}})$
    else
      $$sum^{(j)} \leftarrow \boldsymbol{X}\left(\tilde{\boldsymbol{k}} + \sum_{i\in[d-2]} G_{\mathcal{K}}(\boldsymbol{k}_i^{(j)})\right)$$
  if $j \geq j^* + 1$
    $(\boldsymbol{k}_1^{(j)},\ldots,\boldsymbol{k}_d^{(j)}) \leftarrow\!\!\$\ \mathsf{Share}(\boldsymbol{k})$
    for $i \in [d]$
      $\boldsymbol{z}_i^{(j)} \leftarrow \mathsf{Output}(\boldsymbol{X}, \boldsymbol{k}_i^{(j)}, i)$
    $sum^{(j)} \leftarrow \sum_{i\in\mathcal{I}^{(j)}} \boldsymbol{z}_i^{(j)}$
$\boldsymbol{y}_{ext} \leftarrow \lfloor \boldsymbol{X}\boldsymbol{k} \rfloor_{2^p}$
$state' \leftarrow (\boldsymbol{y}_{ext}, (sum^{(j)})_{j\in[t]})$
if $\mathsf{Abort}'(state')$
  return 1
return 0

Fig. 15. The reduction to the PRG $G_{\mathcal{K}}$ of Claim 6

string. For these reasons, even though the statement includes a wPRF output $\boldsymbol{y}$ with $p - \alpha$ bits, we reduce to a wPRF with a p-bits output. Our reduction $\mathsf{A}_{\mathsf{wPRF}}$ to the one-time security of F takes as input a matrix $\boldsymbol{X}$ (*i.e.* the input to the PRF) and a p-bits output $\boldsymbol{y}_{ext}$, and behaves as in Fig. 16. In short, $\mathsf{A}_{\mathsf{wPRF}}$

$\mathsf{A}_{\mathsf{wPRF}}(\boldsymbol{X}, \boldsymbol{y}_{ext})$

for $j \in [t]$
  $i^* \leftarrow\!\!\$\ [d]$
  $\mathcal{I}^{(j)} \leftarrow [d] \setminus \{i^*\}$
  $\tilde{\boldsymbol{k}}^{(j)} \leftarrow\!\!\$\ \mathbb{Z}_Q^n$
  $sum^{(j)} \leftarrow \boldsymbol{X}\tilde{\boldsymbol{k}}^{(j)}$
$ch \leftarrow (\mathcal{I}^{(j)})_{j\in[t]}$
$state' \leftarrow (\boldsymbol{y}_{ext}, (sum^{(j)})_{j\in[t]})$
if $\mathsf{Abort}'(state')$
  return 1
return 0

Fig. 16. The reduction to the wPRF F of Claim 7

samples $(sum^{(j)})_{j\in[t]}$ as in $\mathsf{Hyb}_t^{(2)}$ and $\mathsf{Hyb}^{(3)}$, and feeds its input $\boldsymbol{y}_{ext}$ as input to Abort' too. This guarantees that when $\boldsymbol{y}_{ext}$ equals $\lfloor \boldsymbol{X}\boldsymbol{k} \rceil_{2^p} = F_{\boldsymbol{k}}(\boldsymbol{X})$, then $\mathsf{A}_{\mathsf{wPRF}}(\boldsymbol{X}, \boldsymbol{y}_{ext})$ perfectly simulates $\mathsf{Hyb}_t^{(2)}$; whereas when $\boldsymbol{y}_{ext}$ is random, then $\mathsf{A}_{\mathsf{wPRF}}(\boldsymbol{X}, \boldsymbol{y}_{ext})$ perfectly simulates $\mathsf{Hyb}^{(3)}$. More formally, this proves that

$$\mathbb{P}\left[\mathsf{wPRF.Real}_F(\mathsf{A}_{\mathsf{wPRF}}) = 1\right] = \mathbb{P}\left[\mathsf{Hyb}_t^{(2)} = 1\right]$$
$$\mathbb{P}\left[\mathsf{wPRF.Rand}_F(\mathsf{A}_{\mathsf{wPRF}}) = 1\right] = \mathbb{P}\left[\mathsf{Hyb}^{(3)} = 1\right].$$

This concludes the proof.

Claim 8 $\mathbb{P}\left[\mathsf{Hyb}^{(3)} = 1\right] \leq tm \cdot 2^{-\alpha}$

Proof (Claim 8). Define Event as the event where Abort outputs 1 in $\mathsf{Hyb}^{(3)}$. Additionally, define Event_j as the event where the j-th iteration triggers the abort, meaning that $sum^{(j)}[r_s..r_E]$ and $\boldsymbol{y}_{ext}[0..\alpha]$ have at least one equal component. Thus, we have $\mathsf{Event} = \bigcup_{i\in[t]} \mathsf{Event}_j$. Moreover, $\mathbb{P}\left[\mathsf{Event}_j\right] \leq m \cdot 2^{\alpha}$, as in every iteration j of $\mathsf{Hyb}^{(3)}$, Abort compares component-wise two independent α-bit strings, one of which is uniformly random. By union bound,

$$\mathbb{P}\left[\mathsf{Hyb}^{(3)} = 1\right] = \mathbb{P}\left[\mathsf{Event}\right] \leq \sum_{i\in[t]} \mathbb{P}\left[\mathsf{Event}_j\right] \leq tm \cdot 2^{\alpha}.$$

This proves the claim.

To conclude the proof of Theorem 14, we now combine Claim 4, 5, 6, 7, and 8. Adding and subtracting the terms corresponding to all the hybrids, we get

$$\begin{aligned}
\mathbb{P}\left[\mathsf{HonestAbort}=1\right] &= \left(\mathbb{P}\left[\mathsf{HonestAbort}=1\right]-\mathbb{P}\left[\mathsf{Hyb}^{(1)}=1\right]\right)\\
&+\left(\mathbb{P}\left[\mathsf{Hyb}^{(1)}=1\right]-\mathbb{P}\left[\mathsf{Hyb}_0^{(2)}=1\right]\right)\\
&+\sum_{j^*\in[t]}\left(\mathbb{P}\left[\mathsf{Hyb}_{j^*}^{(2)}=1\right]-\mathbb{P}\left[\mathsf{Hyb}_{j^*-1}^{(2)}=1\right]\right)\\
&+\left(\mathbb{P}\left[\mathsf{Hyb}_t^{(2)}=1\right]-\mathbb{P}\left[\mathsf{Hyb}^{(3)}=1\right]\right)\\
&+\mathbb{P}\left[\mathsf{Hyb}^{(3)}=1\right]\\
&\leq t\cdot\epsilon_{\mathsf{PRG},\mathcal{K}}+\epsilon_{\mathsf{LWR}}+tm\cdot 2^{-\alpha}.
\end{aligned}$$

E.1 Completeness (Theorem 4)

In the first part of the proof, we use Theorem 14 to upper bound the probability that Abort outputs 1. Then, we use the almost homomorphic properties of the rounding function to prove that, if Abort outputs 0, then honestly generated proofs are always accepted. Consider the event $\mathsf{Event}=\{\mathsf{Abort}(state,ch)=0\}$ in $\Sigma.\mathsf{Comp}_\Sigma$. Then,

$$\begin{aligned}
&\mathbb{P}\left[\Sigma.\mathsf{Comp}_\Sigma=0\right]\\
&=\mathbb{P}\left[\Sigma.\mathsf{Comp}_\Sigma=0,\mathsf{Event}\right]+\mathbb{P}\left[\Sigma.\mathsf{Comp}_\Sigma=0,\neg\mathsf{Event}\right] &(23)\\
&\leq\mathbb{P}\left[\Sigma.\mathsf{Comp}_\Sigma=0,\mathsf{Event}\right]+\mathbb{P}\left[\neg\mathsf{Event}\right] &(24)\\
&\leq\mathbb{P}\left[\Sigma.\mathsf{Comp}_\Sigma=0,\mathsf{Event}\right]+tm\cdot 2^{-\alpha}+t\cdot\epsilon_{\mathsf{PRG},\mathcal{K}}+\epsilon_{\mathsf{LWR}}, &(25)
\end{aligned}$$

where Eq. (23) follows from the law of total probabilities, Eq. (24) follows from the observation that the joint probability is always smaller than the corresponding marginals, and Eq. (25) is true by Theorem 14. In what follows, we prove that

$$\mathbb{P}\left[\Sigma.\mathsf{Comp}_\Sigma=0,\mathsf{Abort}(state,ch)=0\right]=0. \tag{26}$$

According to Eq. (25), this implies the theorem. If $\mathbb{P}\left[\mathsf{Abort}(state,ch)=0\right]=0$, then Eq. (26) is true. Therefore, it suffices to prove Eq. (26) under the assumption that $\mathbb{P}\left[\mathsf{Abort}(state,ch)=0\right]>0$. Via Bayes' Theorem, we get

$$\begin{aligned}
&\mathbb{P}\left[\Sigma.\mathsf{Comp}_\Sigma=0,\mathsf{Abort}(state,ch)=0\right]\\
&=\mathbb{P}\left[\Sigma.\mathsf{Comp}_\Sigma=0|\mathsf{Abort}(state,ch)=0\right]\cdot\mathbb{P}\left[\mathsf{Abort}(state,ch)=0\right]\\
&\leq\mathbb{P}\left[\Sigma.\mathsf{Comp}_\Sigma=0|\mathsf{Abort}(state,ch)=0\right], &(27)
\end{aligned}$$

where the latter probability is well defined because $\mathbb{P}\left[\mathsf{Abort}(state,ch)=0\right]>0$. In what follows, we prove that Eq. (27) equals 0. The experiment $\Sigma.\mathsf{Comp}_\Sigma$ outputs 0 only if at least one of these conditions is met:

1. $((\boldsymbol{X}, \boldsymbol{y}), \boldsymbol{k}) \notin \mathcal{R}$.
2. $resp = \perp$.
3. There exists $j \in [t], i \in \mathcal{I}^{(j)}$ such that $\mathsf{H}(\boldsymbol{k}_i^{(j)}) \neq h_i^{(j)}$.
4. There exists $j \in [t], i \in \mathcal{I}^{(j)}$ such that $\left\lfloor \boldsymbol{z}_i^{(j)} \right\rfloor_{2^{p-\alpha}} \neq \boldsymbol{y}_i^{(j)}$
5. There exists $j \in [t]$ such that $\boldsymbol{y} - \tilde{\boldsymbol{y}}^{(j)} \notin \{0,1\}^m$, where $\tilde{\boldsymbol{y}}^{(j)} = \lfloor sum^{(j)} \rfloor_{2^{p-\alpha}} + \boldsymbol{y}_{i^*}^{(j)}$

Item 1 is never true because the experiment $\Sigma.\mathsf{Comp}_{\Sigma}$ generates $((\boldsymbol{X}, \boldsymbol{y}), \boldsymbol{k})$ with $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$, which only outputs statement-witness pairs in $\mathcal{R}$. Item 2 cannot happen either, as Eq. (27) is under the condition that the prover does not abort. Item 3 and Item 4 are impossible as well in $\Sigma.\mathsf{Comp}_{\Sigma}$, because $\Sigma.\mathsf{Prove}_1$ generates $h_i^{(j)}$ and $\boldsymbol{y}_i^{(j)}$ precisely as $\mathsf{H}(\boldsymbol{k}_i^{(j)})$ and $\left\lfloor \boldsymbol{z}_i^{(j)} \right\rfloor_{2^{p-\alpha}}$, respectively. To conclude the proof, it remains to prove Item 5. The latter argument combines the fact that the key is (pseudorandomly) additively shared with the almost homomorphic behaviour of the rounding function (Proposition 1, Item 3). For every $j \in [t]$, the experiment $\Sigma.\mathsf{Comp}_{\Sigma}$ computes $\boldsymbol{y} - \tilde{\boldsymbol{y}}^{(j)}$ as follows

$$\begin{aligned}
&\boldsymbol{y} - \tilde{\boldsymbol{y}}^{(j)} \\
&= \boldsymbol{y} - \left\lfloor sum^{(j)} \right\rfloor_{2^{p-\alpha}} - \boldsymbol{y}_{i^*}^{(j)} \\
&= \lfloor \boldsymbol{X}\boldsymbol{k} \rfloor_{2^{p-\alpha}} - \left\lfloor \sum_{i \in \mathcal{I}^{(j)}} \mathsf{Output}(\boldsymbol{X}, \boldsymbol{k}_i, i) \right\rfloor_{2^{p-\alpha}} - \lfloor \mathsf{Output}(\boldsymbol{X}, \boldsymbol{k}_{i^*}, i^*) \rfloor_{2^{p-\alpha}} . \quad (28)
\end{aligned}$$

We now want to use the almost homomorphic properties of the rounding, and rewrite Eq. (28) as $\lfloor \boldsymbol{x} + \boldsymbol{x}' \rfloor_{2^{p-\alpha}} - \lfloor \boldsymbol{x} \rfloor_{2^{p-\alpha}} - \lfloor \boldsymbol{x}' \rfloor_{2^{p-\alpha}}$. This would conclude the proof, as Proposition 1, Item 3 guarantees that the latter always lies in $\{0,1\}^m$. We define

$$\boldsymbol{x} = \sum_{i \in \mathcal{I}^{(j)}} \mathsf{Output}(\boldsymbol{X}, \boldsymbol{k}_i, i), \quad \boldsymbol{x}' = \mathsf{Output}(\boldsymbol{X}, \boldsymbol{k}_{i^*}, i^*).$$

Therefore, the lemma follows if we show that

$$\boldsymbol{x} + \boldsymbol{x}' = \boldsymbol{X}\boldsymbol{k}. \quad (29)$$

Here, we use the fact that the key is (pseudorandomly) additively shared. More formally, since $(\boldsymbol{k}_i)_{i\in[d]}$ is generated with $\mathsf{Share}(\boldsymbol{k})$, we know that

$$\boldsymbol{k} = \sum_{i \in [d-1]} G_{\mathcal{K}}(\boldsymbol{k}_i) + \boldsymbol{k}_d. \quad (30)$$

Therefore, we can rewrite the right-hand side of Eq. (29) as

$$\boldsymbol{X}\boldsymbol{k} = \sum_{i \in [d-1]} \boldsymbol{X} G_{\mathcal{K}}(\boldsymbol{k}_i) + \boldsymbol{X}\boldsymbol{k}_d \quad (31)$$

$$= \sum_{i\in[d]} \mathsf{Output}(\boldsymbol{X}, \boldsymbol{k}_i, i) \tag{32}$$

$$= \sum_{i\in\mathcal{I}^{(j)}} \mathsf{Output}(\boldsymbol{X}, \boldsymbol{k}_i, i) + \mathsf{Output}(\boldsymbol{X}, \boldsymbol{k}_{i^*}, i^*) \tag{33}$$

$$= \boldsymbol{x} + \boldsymbol{x}'. \tag{34}$$

where Eq. (31) uses Eq. (30) and the linearity of the matrix product, Eq. (32) uses the definition of Output and groups the summands together, Eq. (33) splits the summands isolating the index i^*, and Eq. (34) uses the definition of $\boldsymbol{x}$ and $\boldsymbol{x}'$. This proves Eq. (29), and therefore the theorem.

E.2 Average-Case Accepting Multi-HVZK (Theorem 5)

The proof follows by a sequence of hybrid experiments, moving from the real output to the simulated output. We denote by **Success** the event of a successful computation (i.e., the algorithm Abort outputs 0), and we denote by **Abort** the opposite event.

Before proceeding with the actual proof, we define some auxiliary tools. For pairs $(\beta, j), (\beta', j') \in [N] \times [t]$, we define the lexicographic relation $\leq_{\mathsf{lex}}$, defined as

$$(\beta, j) \leq_{\mathsf{lex}} (\beta', j') \iff \beta < \beta^* \vee (\beta = \beta^* \wedge j \leq j^*)\,.$$

We define in Fig. 17 some helper functions that will make the proof easier to follow, and we summarize them below.

- A "batch" hash function $\mathsf{H}\langle\!\langle \beta^*, j^* \rangle\!\rangle$ that, intuitively, uses H to compute the hash values of the list of keys provided as input. Additionally, given a set of indices i^*, $\mathsf{H}\langle\!\langle \beta^*, j^* \rangle\!\rangle$ replaces some of the keys (depending on the parameters β^* and j^*) indexed by i^* and produces hash values of random keys instead.
- A "batch" sharing algorithm $\mathsf{Share}\langle\!\langle \beta^*, j^* \rangle\!\rangle$ that runs Share Nt times and produces a secret sharing of either the input value $\boldsymbol{k}$ or of a uniformly random key $\boldsymbol{k}^* \leftarrow\!\!\$\, \mathbb{Z}_Q^n$, depending on the parameters β^* and j^*.
- A "batch" output algorithm $\mathsf{Output}\langle\!\langle b \rangle\!\rangle$ that computes Output on every key share, except, when $b = 1$, for the ones indexed by i^*, for which the output value is computed by a special function Calc, described below.
- A function Calc that computes the share $\boldsymbol{y}_{i^*}$ given the computation of $\boldsymbol{z}_i \leftarrow \mathsf{Output}(\boldsymbol{X}, \boldsymbol{k}_i, i)$ for all the $i \neq i^*$, and the extended output $\boldsymbol{y}_{ext}$. Looking ahead, this computation does not always output the same as computing $\boldsymbol{y}_{i^*}$ directly as $\lfloor \boldsymbol{z}_i \rfloor_{2^{p-\alpha}}$; however, the cases in which the two outputs differ are the cases in which the **Abort** happens, which is never the case if we condition to **Success**.

For the sake of completeness, the descriptions of both the real experiment and the simulator are depicted in Fig. 18, in which we made explicit the steps of $\Sigma.\mathsf{Prove}_1$. In particular, the view of the adversary in the real setting is given by the distribution

$$\mathsf{Real}(\mathcal{D}_{\mathcal{X}\times\mathcal{W}}),$$

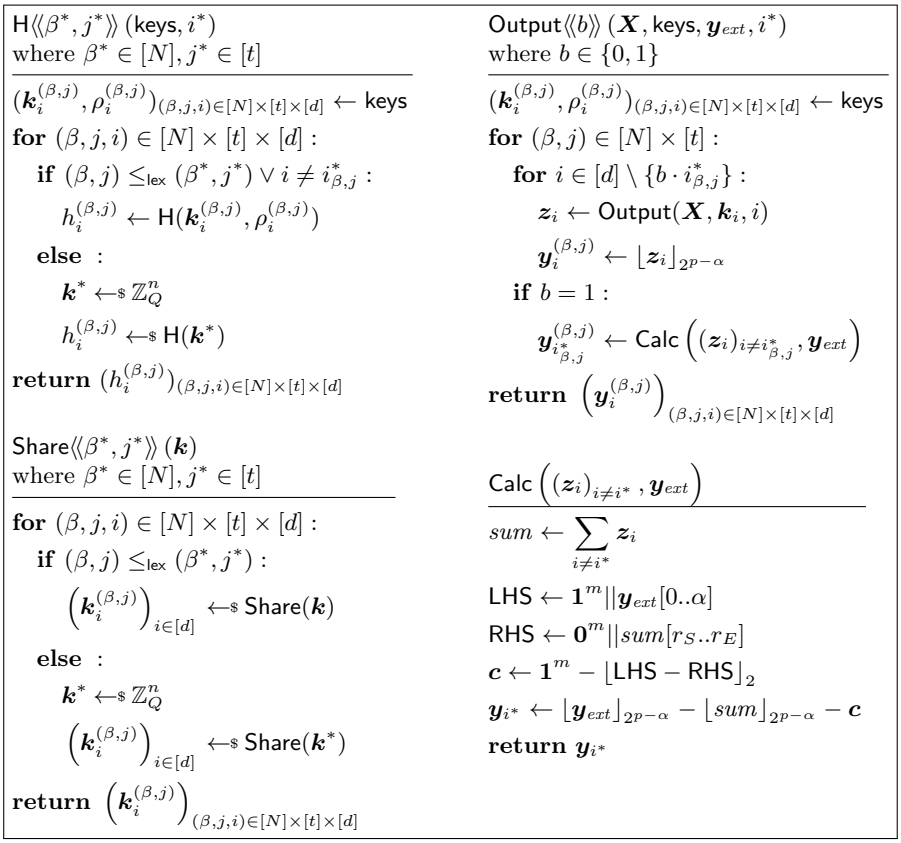

Fig. 17. The helper functions. The order relation $\leq_{\mathsf{lex}}$ is defined as $(\beta, j) \leq_{\mathsf{lex}} (\beta^*, j^*)$ whenever $\beta < \beta^*$ or $\beta = \beta^*$ and $j \leq j^*$.

while the simulated view is output by

$$\mathsf{Sim}_\omega(\mathcal{D}_{\mathcal{X}\times\mathcal{W}}).$$

The simulator tries ω times to produce a list of valid simulated transcripts, after which gives up and outputs $\perp$. In particular, for $\omega = 1$, only a single attempt is executed, which makes Sim_ω computationally close to Real only when conditioning on the event **Success** for both of them. We will show that, by increasing ω, the probability that Sim_ω outputs $\perp$ rapidly decreases, making the simulator efficient in expected runtime. Furthermore, for $\omega \to \infty$, the probability of $\perp$ goes to zero.

$\mathsf{Real}((\boldsymbol{X},\boldsymbol{y}),\boldsymbol{k})$

$\boldsymbol{y}_{ext} \leftarrow \lfloor \boldsymbol{X}\boldsymbol{k} \rfloor_{2^p}$
for $\beta \in [N]$:
  for $j \in [t]$:
    $(\boldsymbol{k}_1^{(\beta,j)},\ldots,\boldsymbol{k}_d^{(\beta,j)}) \leftarrow\$ \mathsf{Share}(\boldsymbol{k})$
    for $i \in [d]$
      $\boldsymbol{z}_i^{(\beta,j)} \leftarrow \mathsf{Output}(\boldsymbol{X}, \boldsymbol{k}_i^{(\beta,j)}, i)$
      $\boldsymbol{y}_i^{(\beta,j)} \leftarrow \left\lfloor \boldsymbol{z}_i^{(\beta,j)} \right\rfloor_{2^{p-\alpha}}$
      $\rho_i^{(\beta,j)} \leftarrow\$ \mathcal{P}_{\mathsf{H}}$
      $h_i^{(\beta,j)} \leftarrow \mathsf{H}(\boldsymbol{k}_i^{(\beta,j)}, \rho_i^{(\beta,j)})$
  $\mathit{firstmes}_\beta \leftarrow (\boldsymbol{y}_i^{(\beta,j)}, h_i^{(\beta,j)})_{i\in[d],j\in[t]}$
  $ch_\beta := (\mathcal{I}^{(\beta,j)})_{j\in[t]} \leftarrow\$ \mathcal{N}_{\mathsf{ch}}$
  $\mathsf{keys}_\beta \leftarrow (\boldsymbol{k}_i^{(\beta,j)}, \rho_i^{(\beta,j)})_{(i\in[d],j\in[t]}$
  $\mathit{state}_\beta \leftarrow (\boldsymbol{X}, \boldsymbol{y}_{ext}, \mathsf{keys}_\beta)$
  $\mathit{resp}_\beta \leftarrow \Sigma.\mathsf{Prove}_2(\mathit{state}, ch)$
  $\mathsf{tr}_\beta = (\mathit{firstmes}_\beta, ch_\beta, \mathit{resp}_\beta)$
return $\left(\boldsymbol{X}, \boldsymbol{y}, (\mathsf{tr}_\beta)_{\beta\in[N]}\right)$

$\mathsf{Sim}_\omega(\boldsymbol{X},\boldsymbol{y})$

for $_ \in [\omega]$:
  $\boldsymbol{r} \leftarrow\$ \mathbb{Z}_{2^\alpha}^n$
  $\boldsymbol{y}_{ext} \leftarrow 2^\alpha \boldsymbol{y} + \boldsymbol{r}$
  $i^* \leftarrow\$ [d]^{N\times t}$
  $(ch_\beta)_{\beta\in[N]} := ([d] \setminus \{i^*_{\beta,j}\})_{\beta\in[N],j\in[t]}$
  $\mathsf{keys} \leftarrow\$ \mathsf{Share}\langle\langle 0,0\rangle\rangle(\boldsymbol{k})$
  $\mathsf{outs} \leftarrow\$ \mathsf{Output}\langle\langle 1\rangle\rangle(\boldsymbol{X}, \mathsf{keys}, \boldsymbol{y}_{ext}, i^*)$
  $\mathsf{hashes} \leftarrow\$ \mathsf{H}\langle\langle 0,0\rangle\rangle(\mathsf{keys}, i^*)$
  for $\beta \in [N]$:
    $\mathit{firstmes}_\beta \leftarrow (\mathsf{outs}_\beta, \mathsf{hashes}_\beta)$
    $\mathit{state}_\beta \leftarrow (\boldsymbol{X}, \boldsymbol{y}_{ext}, \mathsf{keys}_\beta)$
    $\mathit{resp}_\beta \leftarrow \Sigma.\mathsf{Prove}_2(\mathit{state}_\beta, ch_\beta)$
    $\mathsf{tr}_\beta \leftarrow (\mathit{firstmes}_\beta, ch_\beta, \mathit{resp}_\beta)$
  if $\forall\beta \in [N], \mathit{resp}_\beta \neq \perp$:
    return $\left(\boldsymbol{X}, \boldsymbol{y}, (\mathsf{tr}_\beta)_{\beta\in[N]}\right)$
return $\perp$

Fig. 18. The real view and the simulated view, appearing in the proof of Theorem 5.

The hybrid experiments are all depicted in Figs. 19 and 20, in which we also highlight in a yellow box what changes from the previous experiment. We give an overview of the experiments below, along with an intuition of how to move from one of them to the next one.

$\mathsf{Real}(\mathcal{D}_{\mathcal{X}\times\mathcal{W}})$**:** As stated before, this is the real experiment, in which we made explicit the steps of $\Sigma.\mathsf{Prove}_1$.

$\mathsf{Hyb}_0(\mathcal{D}_{\mathcal{X}\times\mathcal{W}})$**:** This is just a syntactic modification of the real experiment, in which we move some computation at the beginning and use the above "batch" functions instead of computing everything in sequence in a loop.

$\mathsf{Hyb}_1(\mathcal{D}_{\mathcal{X}\times\mathcal{W}})$**:** Here we replace the "batch" function $\mathsf{H}\langle\langle N, t\rangle\rangle$ with $\mathsf{H}\langle\langle 0, 0\rangle\rangle$, thus changing the distribution of the hash values for the indices i^*. Namely, for those indices, instead of computing the correct hash value, we compute the hash value for a uniform, unrelated key. By a hybrid argument via a simple reduction to the inner hash function H, we show that this is computationally close to the previous hybrid.

$\mathsf{Hyb}_2(\mathcal{D}_{\mathcal{X}\times\mathcal{W}})$**:** Here we replace the "batch" function $\mathsf{Output}\langle\langle 0\rangle\rangle$ with $\mathsf{Output}\langle\langle 1\rangle\rangle$, thus changing the computation of the output values $\boldsymbol{y}^{(\beta,j)}$ for the indices i^*. Namely, instead of computing such values by the usual rounding function, we use the Calc function, that simulates the value of the carry and uses it to compute $\boldsymbol{y}^{(\beta,j)}_{i^*_{\beta,j}}$. We will show that, whenever **Abort** does not happen, the simulated carry is computed in the same way as the real one, and, therefore, the reconstructed output value $\boldsymbol{y}^{(\beta,j)}_{i^*_{\beta,j}}$ is identically distributed to the real one as well. For this reason, when conditioning on **Success**, this experiment is identical to the previous one.

$\mathsf{Hyb}_3(\mathcal{D}_{\mathcal{X}\times\mathcal{W}})$**:** Here we replace the "batch" function $\mathsf{Share}\langle\langle N, t\rangle\rangle$ with $\mathsf{Share}\langle\langle 0, 0\rangle\rangle$, thus changing the distribution of the key shares $\boldsymbol{k}_i^{(\beta,j)}$. Namely, now the key shares do not depend anymore on the original key $\boldsymbol{k}$, but are sampled using a uniform, unrelated key. By a hybrid argument via a reduction to the inner secret sharing scheme Share (which, in turns, requires a reduction to the inner PRG), we show that this is computationally close to the previous hybrid.

$\mathsf{Hyb}_4(\mathsf{LWR.Real}(\boldsymbol{k}))$**:** This is a syntactic modification of the previous experiment. Namely, in the previous experiment, the witness $\boldsymbol{k}$ was just used to compute $\boldsymbol{y}_{ext}$; therefore, here we replace the input $\mathcal{D}_{\mathcal{X}\times\mathcal{W}}$ with an LWR instance $(\boldsymbol{X}, \boldsymbol{y}_{ext})$ in which $\boldsymbol{y}_{ext}$ is given.

$\mathsf{Hyb}_4(\mathsf{LWR.Rand}())$**:** Here we apply a simple reduction to LWR to move from the real instance to a random one in which $\boldsymbol{y}_{ext}$ is a uniform vector.

$\mathsf{Sim}_1(\mathsf{LWR.Rand}())$**:** Finally, this is the simulator for $\omega = 1$, which, conditioning on **Success**, can be seen as a syntactic modification of the previous experiment. The only non-syntactic modification consists of how the value $\boldsymbol{y}_{ext}$ is sampled: in the previous case, $\boldsymbol{y}_{ext}$ is just a uniform vector, while, in this experiment, $\boldsymbol{y}$ is a uniform vector and $\boldsymbol{y}_{ext}$ is computed starting from $\boldsymbol{y}$ and uniformly sampling the missing bits; however, the distribution of $\boldsymbol{y}_{ext}$ turns out to be identical in the two experiments.

$\mathsf{Sim}_1(\mathsf{LWR.Real}(\boldsymbol{k}))$**:** This is the last experiment of the chain of hybrid experiments, in which, by a simple reduction to LWR, we return back to the situation in which $(\boldsymbol{X}, \boldsymbol{y})$ comes from the real distribution.

The final step consists in switching from Sim_1 to Sim_∞, where the simulator only generates successful views. The full proof of Theorem 5 follows.

We start by proving the indistinguishability of the inner components.

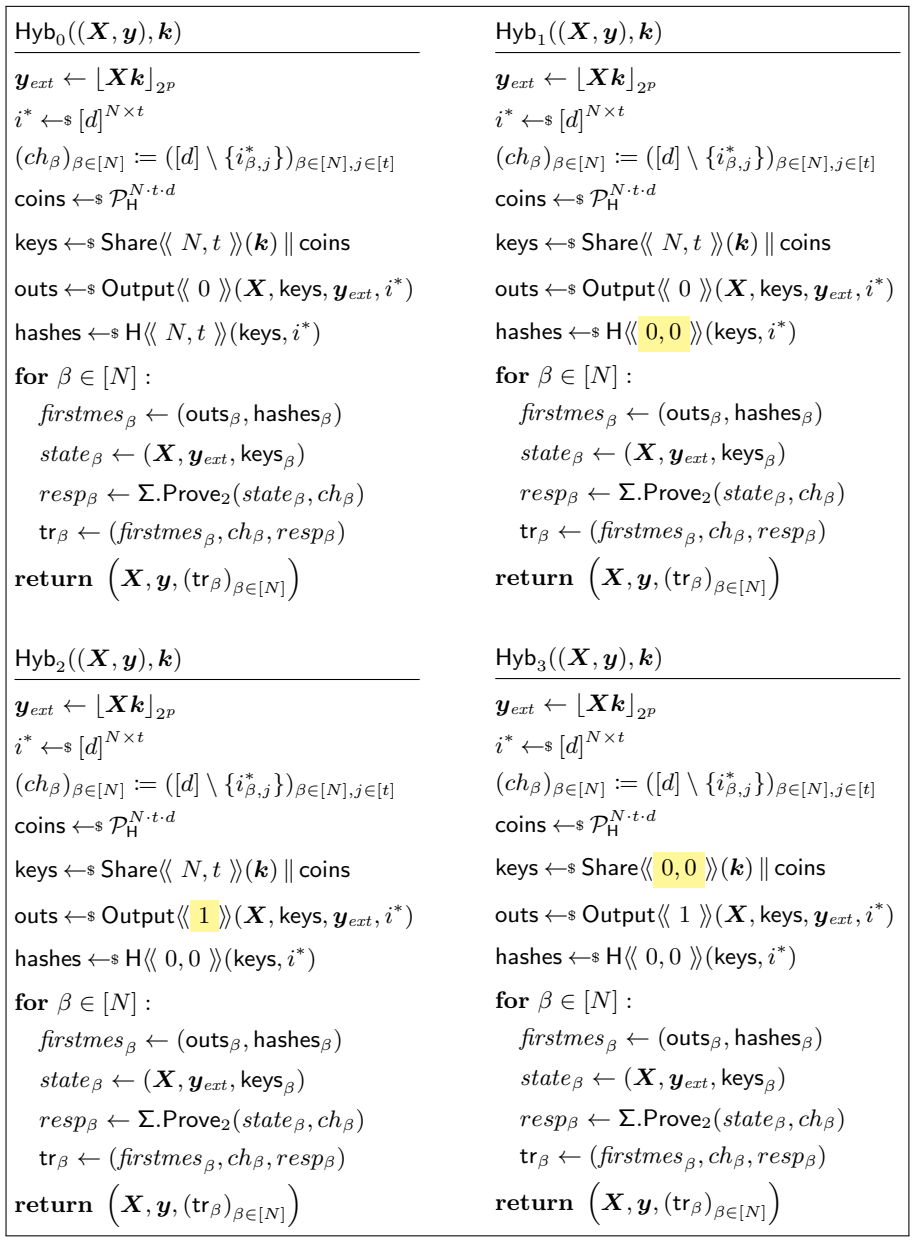

Fig. 19. The first four hybrid experiments in the proof of Theorem 5.

$\mathsf{Hyb}_4(\boldsymbol{X}, \boldsymbol{y}_{ext})$

$i^* \leftarrow\!\!\$\ [d]^{N\times t}$

$(ch_\beta)_{\beta\in[N]} \coloneqq ([d] \setminus \{i^*_{\beta,j}\})_{\beta\in[N],j\in[t]}$

$\mathsf{coins} \leftarrow\!\!\$\ \mathcal{P}_{\mathsf{H}}^{N\cdot t\cdot d}$

$\mathsf{keys} \leftarrow\!\!\$\ \mathsf{Share}\langle\langle 0,0\rangle\rangle(\ \bot\ ) \parallel \mathsf{coins}$

$\mathsf{outs} \leftarrow\!\!\$\ \mathsf{Output}\langle\langle 1\rangle\rangle(\boldsymbol{X}, \mathsf{keys}, \boldsymbol{y}_{ext}, i^*)$

$\mathsf{hashes} \leftarrow\!\!\$\ \mathsf{H}\langle\langle 0,0\rangle\rangle(\mathsf{keys}, i^*)$

for $\beta \in [N]$:

$\quad firstmes_\beta \leftarrow (\mathsf{outs}_\beta, \mathsf{hashes}_\beta)$

$\quad state_\beta \leftarrow (\boldsymbol{X}, \boldsymbol{y}_{ext}, \mathsf{keys}_\beta)$

$\quad resp_\beta \leftarrow \Sigma.\mathsf{Prove}_2(state_\beta, ch_\beta)$

$\quad \mathsf{tr}_\beta \leftarrow (firstmes_\beta, ch_\beta, resp_\beta)$

return $\left(\boldsymbol{X}, \boldsymbol{y}, (\mathsf{tr}_\beta)_{\beta\in[N]}\right)$

Fig. 20. The last hybrid experiment in the proof of Theorem 5.

Lemma 11. *For every vector of keys* $\mathsf{keys} \coloneqq \left(\boldsymbol{k}_i^{(\beta,j)}\right)_{(\beta,j,i)\in[N]\times[t]\times[d]}$*, every vector of indices* $i^* \coloneqq \left(i^*_{\beta,j}\right)_{(\beta,j)\in[N]\times[t]}$ *and every stateful quantum polynomial-time adversary* A*,*

$$|\mathbb{P}\left[\mathsf{A}\left(\mathsf{H}\langle\langle N,t\rangle\rangle(\mathsf{keys}, i^*)\right)\right] - \mathbb{P}\left[\mathsf{A}\left(\mathsf{H}\langle\langle 0,0\rangle\rangle(\mathsf{keys}, i^*)\right)\right]| \leq Nt\epsilon_{\mathsf{Hide}}.$$

Furthermore, it is well defined $\mathsf{H}\langle\langle N,t\rangle\rangle(\mathsf{keys})$ *such that, for all* i^**,*

$$\mathsf{H}\langle\langle N,t\rangle\rangle(\mathsf{keys}) = \mathsf{H}\langle\langle N,t\rangle\rangle(\mathsf{keys}, i^*)\,.$$

Proof. The proof proceeds by hybrid argument. Notice that, for $\beta, \beta^* \in [N], j \in [t]$, the condition $(\beta, j) \leq_{\mathsf{lex}} (\beta^*, 0)$ implies that $\beta < \beta^*$ (since $j \geq 1$), or, equivalently, $\beta \leq \beta^* - 1$. Since $j \leq t$, this also implies $(\beta, j) \leq_{\mathsf{lex}} (\beta^* - 1, t)$, which means that the condition in $\mathsf{H}\langle\langle\beta^*, 0\rangle\rangle$ and in $\mathsf{H}\langle\langle\beta^* - 1, t\rangle\rangle$ is actually the same and the two experiments are identical. Furthermore, the condition $(\beta, j) \leq_{\mathsf{lex}} (\beta^*, j^*)$ is never true for $\beta^* = 0$, making all the experiments $\mathsf{H}\langle\langle 0, j^*\rangle\rangle$ for $j^* \in [t] \cup \{0\}$ identical. We now show that for $(\beta^*, j^*) \in [N] \times [t]$, the computational distance between $\mathsf{H}\langle\langle\beta^*, j^*\rangle\rangle$ and $\mathsf{H}\langle\langle\beta^*, j^* - 1\rangle\rangle$ is bounded by ϵ_{Hide}; the first part of the lemma will then follow by a nested hybrid argument: indeed, for every $\beta^* \in [N]$ and every stateful quantum polynomial-time adversary A, an hybrid argument of t experiments shows that

$$|\mathbb{P}\left[\mathsf{A}\left(\mathsf{H}\langle\langle\beta^*, t\rangle\rangle(\mathsf{keys}, i^*)\right)\right] - \mathbb{P}\left[\mathsf{A}\left(\mathsf{H}\langle\langle\beta^*, 0\rangle\rangle(\mathsf{keys}, i^*)\right)\right]| \leq t\epsilon_{\mathsf{Hide}},$$

and, by $\mathsf{H}\langle\langle\beta^*, 0\rangle\rangle \equiv \mathsf{H}\langle\langle\beta^* - 1, t\rangle\rangle$ we again have an hybrid arguments of N experiments.

Fix $(\beta^*, j^*) \in [N] \times [t]$. Then, the only difference between $\mathsf{H}\langle\langle \beta^*, j^* \rangle\rangle$ and $\mathsf{H}\langle\langle \beta^*, j^* - 1 \rangle\rangle$ is how the hash $h^{(\beta^*,j^*)}_{i^*_{\beta^*,j^*}}$ is computed. Namely, $h^{(\beta^*,j^*)}_{i^*_{\beta^*,j^*}}$ is a commitment to $\boldsymbol{k}^{(\beta^*,j^*)}_{i^*_{\beta^*,j^*}}$ in $\mathsf{H}\langle\langle \beta^*, j^* \rangle\rangle$, and it is a commitment to an unrelated random value in $\mathsf{H}\langle\langle \beta^*, j^* - 1 \rangle\rangle$. The first part of the lemma follows by a simple reduction to the hiding property of H. The second part follows by realizing that all the hash values are computed from the real keys in $\mathsf{H}\langle\langle N, t \rangle\rangle$ and, therefore, the function does not depend on i^*. □

Lemma 12. *For every* $\mathsf{keys} \coloneqq (\boldsymbol{k}_i^{(\beta,j)})_{(\beta,j,i)\in[N]\times[t]\times[d]}, i^* \coloneqq \left(i^*_{\beta,j}\right)_{(\beta,j)\in[N]\times[t]}$, $\boldsymbol{X}, \boldsymbol{y}_{ext}$ *and every stateful quantum polynomial-time adversary* A*,*

$$\begin{aligned}&\mathbb{P}\left[\mathsf{A}\left(\mathsf{Output}\langle\langle 0\rangle\rangle\left(\boldsymbol{X}, \mathsf{keys}, \boldsymbol{y}_{ext}, i^*\right)\right) \mid \mathbf{Success}\right]\\ &= \mathbb{P}\left[\mathsf{A}\left(\mathsf{Output}\langle\langle 1\rangle\rangle\left(\boldsymbol{X}, \mathsf{keys}, \boldsymbol{y}_{ext}, i^*\right)\right) \mid \mathbf{Success}\right]\end{aligned}$$

Proof. First of all, notice that the only difference concerns the computation of the values $\boldsymbol{y}^{(\beta,j)}_{i^*_{\beta,j}}$ for $(\beta, j) \in [N] \times [t]$. To make notation easier, we focus on a single iteration, and we drop the indices β and j; therefore, we focus on the computation $\boldsymbol{y}_{i^*}$.

In $\mathsf{Output}\langle\langle 0\rangle\rangle$, all the values $\boldsymbol{y}_i$ for $i \in [d]$ are computed by using the function Output to get $\boldsymbol{z}_i$ and then rounding, while, in $\mathsf{Output}\langle\langle 1\rangle\rangle$, the values $\boldsymbol{y}_{i^*}$ are computed using the special function Calc. Expanding the definition of Calc, we get that, in $\mathsf{Output}\langle\langle 0\rangle\rangle$, the values $\boldsymbol{y}$ and $\boldsymbol{y}_{i^*}$ are computed as

$$\boldsymbol{y} = \lfloor \boldsymbol{X}\boldsymbol{k} \rfloor_{2^{p-\alpha}} \qquad \text{and} \qquad \boldsymbol{y}_{i^*} = \lfloor \boldsymbol{z}_{i^*} \rfloor_{2^{p-\alpha}}$$

respectively. Notice that $\boldsymbol{X}\boldsymbol{k} = \sum_{i=1}^{d} \boldsymbol{z}_i$ and, therefore, by recalling that $r_E = q - p + \alpha$,

$$\begin{aligned}\boldsymbol{y} &= \left\lfloor \sum_{i=1}^{d} \boldsymbol{z}_i \right\rfloor_{2^{p-\alpha}}\\ &= \left\lfloor \boldsymbol{z}_{i^*} + \sum_{i \neq i^*} \boldsymbol{z}_i \right\rfloor_{2^{p-\alpha}}\\ &= \lfloor \boldsymbol{z}_{i^*} + sum \rfloor_{2^{p-\alpha}}\\ &= \boldsymbol{y}_{i^*} + \lfloor sum \rfloor_{2^{p-\alpha}} + \lfloor \boldsymbol{z}_{i^*}[0..r_E] + sum[0..r_E] \rfloor_{2^{p-\alpha}}\,,\end{aligned}$$

where the last equality follows by Item 2 of Proposition 2. By letting

$$\boldsymbol{c}' \coloneqq \lfloor \boldsymbol{z}_{i^*}[0..r_E] + sum[0..r_E] \rfloor_{2^{p-\alpha}}\,,$$

and recalling that $\boldsymbol{y} = \lfloor \boldsymbol{y}_{ext} \rfloor_{2^{p-\alpha}}$, we get another way to compute $\boldsymbol{y}_{i^*}$, that is,

$$\boldsymbol{y}_{i^*} = \lfloor \boldsymbol{y}_{ext} \rfloor_{2^{p-\alpha}} - \lfloor sum \rfloor_{2^{p-\alpha}} - \boldsymbol{c}'.$$

On the other side, in $\mathsf{Output}\langle\langle 1\rangle\rangle$, the value $\boldsymbol{y}_{i^*}$ is computed directly as

$$\boldsymbol{y}_{i^*} = \lfloor \boldsymbol{y}_{ext} \rfloor_{2^{p-\alpha}} - \lfloor sum \rfloor_{2^{p-\alpha}} - \boldsymbol{c},$$

and, therefore, we only need to show that, when **Success** happens, the two ways of computing the carry are equivalent, i.e.,

$$\boldsymbol{c} = \boldsymbol{c}'.$$

Since $\lfloor \cdot \rfloor$ acts component-wise, we show that this holds for every component of the m-sized vectors that we are considering; namely, assuming that $\boldsymbol{y}_{ext}, sum, \boldsymbol{y}$, $\boldsymbol{z}_{i^*}$ are all scalars, we want to show that

$$1 - \lfloor 2^\alpha + \boldsymbol{y}_{ext}[0..\alpha] - sum[r_S..r_E] \rfloor_2 = \lfloor \boldsymbol{z}_{i^*}[0..r_E] + sum[0..r_E] \rfloor_{2^{p-\alpha}}, \quad (35)$$

where the left side of the above is how $\boldsymbol{c}$ is computed and the right side is exactly $\boldsymbol{c}'$. We split the proof in three cases.

- *Case* $\boldsymbol{y}_{ext}[0..\alpha] - sum[r_S..r_E] > 0$. In this case,

$$\begin{aligned} 2^\alpha &< 2^\alpha + \boldsymbol{y}_{ext}[0..\alpha] - sum[r_S..r_E] \\ &\leq 2^\alpha + \boldsymbol{y}_{ext}[0..\alpha] \\ &< 2^\alpha + 2^\alpha, \end{aligned}$$

  hence

$$\lfloor 2^\alpha + \boldsymbol{y}_{ext}[0..\alpha] - sum[r_S..r_E] \rfloor_2 = 1$$

  and the left side of Eq. (35) is 0. On the other side, by $\boldsymbol{y}_{ext}[0..\alpha] = \boldsymbol{z}[r_S..r_E]$ and Item 4 of Proposition 2,

$$\boldsymbol{y}_{ext}[0..\alpha] > sum[r_S..r_E] \quad \Longrightarrow \quad \boldsymbol{z}[0..r_E] > sum[0..r_E].$$

  In this case, $\boldsymbol{z}_{i^*} = \boldsymbol{z} - sum$, and the same happens when only considering the r_E LSBs, which means that the right side of the equality simplifies as

$$\begin{aligned} \lfloor \boldsymbol{z}_{i^*}[0..r_E] + sum[0..r_E] \rfloor_{2^{p-\alpha}} &= \lfloor \boldsymbol{z}[0..r_E] - sum[0..r_E] + sum[0..r_E] \rfloor_{2^{p-\alpha}} \\ &= \lfloor \boldsymbol{z}[0..r_E] \rfloor_{2^{p-\alpha}} \\ &= 0. \end{aligned}$$

- *Case* $\boldsymbol{y}_{ext}[0..\alpha] - sum[r_S..r_E] < 0$. This case is very similar to the previous one, except that the carry assumes value 1. Namely, we get that

$$0 < 2^\alpha + \boldsymbol{y}_{ext}[0..\alpha] - sum[r_S..r_E] < 2^\alpha,$$

  and the left side of Eq. (35) simplifies as

$$1 - \lfloor 2^\alpha + \boldsymbol{y}_{ext}[0..\alpha] - sum[r_S..r_E] \rfloor_2 = 1 - 0 = 1.$$

On the other side, by $\boldsymbol{y}_{ext}[0..\alpha] = \boldsymbol{z}[r_S..r_E]$ and Item 4 of Proposition 2, again,

$$\boldsymbol{y}_{ext}[0..\alpha] < sum[r_S..r_E] \quad \Longrightarrow \quad \boldsymbol{z}[0..r_E] < sum[0..r_E].$$

In this case, $\boldsymbol{z}_{i^*} = 2^{p-\alpha} + \boldsymbol{z} - sum$, the same happens when only considering the r_E LSBs, which means that the right side of the equality simplifies as

$$\begin{aligned}
&\lfloor \boldsymbol{z}_{i^*}[0..r_E] + sum[0..r_E] \rfloor_{2^{p-\alpha}} \\
&= \left\lfloor 2^{p-\alpha} + \boldsymbol{z}[0..r_E] - sum[0..r_E] + sum[0..r_E] \right\rfloor_{2^{p-\alpha}} \\
&= \left\lfloor 2^{p-\alpha} + \boldsymbol{z}[0..r_E] \right\rfloor_{2^{p-\alpha}} \\
&= 1.
\end{aligned}$$

– *Case* $\boldsymbol{y}_{ext}[0..\alpha] = sum[r_S..r_E]$. In this case, it may happen that $\boldsymbol{c} \neq \boldsymbol{c}'$ with non-negligible probability. However, this is also the case that triggers $\mathsf{Abort} = 1$ and, therefore, we don't need the condition $\boldsymbol{c} = \boldsymbol{c}'$.

Finally, the values *sum* and $\boldsymbol{y}_{ext}$ are computed in the same way in both experiments and, since they are the only ones influencing the event **Abort**, the probability of **Success** is the same in both experiments. The claim follows by extending the above reasoning to all components of the involved vectors. □

Now we show, via a sequence of hybrid experiments, that $\mathsf{Hyb}_2(\mathcal{D}_{\mathcal{X}\times\mathcal{W}})$ and $\mathsf{Sim}_1(\mathcal{D}_{\mathcal{X}})$ are computationally close. Looking forward, we will use this fact, with an additional lemma, to show that, when **Abort** does not happen, Real and Sim_1 are computationally close.

Lemma 13. *For every* $\boldsymbol{X} \in \mathbb{Z}_Q^{m\times n}, \boldsymbol{k} \in \mathbb{Z}_Q^n, \boldsymbol{y} = \lfloor \boldsymbol{X}\boldsymbol{k} \rfloor$*, and every stateful quantum polynomial-time adversary* A*,*

$$|\mathbb{P}\left[\mathsf{A}\left(\mathsf{Hyb}_2\left((\boldsymbol{X},\boldsymbol{y}),\boldsymbol{k}\right)\right)\right] - \mathbb{P}\left[\mathsf{A}\left(\mathsf{Hyb}_3\left((\boldsymbol{X},\boldsymbol{y}),\boldsymbol{k}\right)\right)\right]| \leq Nt\epsilon_{\mathsf{PRG}}.$$

Proof. For any $(\beta^*, j^*) \in [N] \times [t]$, let $\mathsf{Hyb}^*\langle\langle \beta^*, j^* \rangle\rangle$ be the same as Hyb_2 (resp., Hyb_3) except for the fact that we replace $\mathsf{Share}\langle\langle N, t\rangle\rangle$ (resp., $\mathsf{Share}\langle\langle 0, 0\rangle\rangle$) with $\mathsf{Share}\langle\langle \beta^*, j^*\rangle\rangle$. In particular, we get that $\mathsf{Hyb}^*\langle\langle N, t\rangle\rangle \equiv \mathsf{Hyb}_2$ and $\mathsf{Hyb}^*\langle\langle 0, 0\rangle\rangle \equiv \mathsf{Hyb}_3$, since the batch parameters of Share are the only difference between the two experiments. As in the proof for Lemma 11, we only need to prove that, for any $(\beta^*, j^*) \in [N] \times [t]$,

$$|\mathbb{P}\left[\mathsf{A}\left(\mathsf{Hyb}^*\langle\langle \beta^*, j^*\rangle\rangle\left((\boldsymbol{X},\boldsymbol{y}),\boldsymbol{k}\right)\right)\right] - \mathbb{P}\left[\mathsf{A}\left(\mathsf{Hyb}^*\langle\langle \beta^*, j^*-1\rangle\rangle\left((\boldsymbol{X},\boldsymbol{y}),\boldsymbol{k}\right)\right)\right]| \leq \epsilon_{\mathsf{PRG}}$$

and then conclude by nested hybrid argument. We proceed by reduction to the PRG $G_{\mathcal{K}}$. Notice that, in both experiments, the key share $\boldsymbol{k}_{i^*_{\beta^*,j^*}}^{(\beta^*,j^*)}$ is never revealed, nor used anywhere. Indeed, $\boldsymbol{y}_{i^*_{\beta^*,j^*}}^{(\beta^*,j^*)}$ is computed by using all the other shares, $h_{i^*_{\beta^*,j^*}}^{(\beta^*,j^*)}$ is the commitment to a uniformly sampled and independent value $\boldsymbol{k}^*$, and, being $ch_{\beta^*,j^*} = [d] \setminus \{i^*_{\beta^*,j^*}\}$, share $\boldsymbol{k}_{i^*_{\beta^*,j^*}}^{(\beta^*,j^*)}$ will never make it into *resp.* We divide the proof in two cases.

Case $i^*_{\beta^*,j^*} = d$**.** In this case, $\boldsymbol{k}_1^{(\beta^*,j^*)}, \ldots, \boldsymbol{k}_{d-1}^{(\beta^*,j^*)}$ are generated in the same way in both experiments: in particular, they are randomly sampled, independently of the key $\boldsymbol{k}$. Therefore, in this case, the two experiments are identical.

Case $i^*_{\beta^*,j^*} < d$**.** This case is slightly more complicated, because we now give to the adversary the value

$$\boldsymbol{k}_d^{(\beta^*,j^*)} = \boldsymbol{k} - \sum_{i=1}^{d-1} G_{\mathcal{K}}\left(\boldsymbol{k}_i^{(\beta^*,j^*)}\right),$$

which depends on $\boldsymbol{k}$. However, in this case, one of the seeds $\boldsymbol{k}_{i^*_{\beta^*,j^*}}^{(\beta^*,j^*)}$ is never revealed, and a simple reduction to $G_{\mathcal{K}}$ shows that we can replace the PRG generated share $G_{\mathcal{K}}(\boldsymbol{k}_{i^*}^{(\beta^*,j^*)})$ with a uniformly sampled one $\tilde{\boldsymbol{k}}_{i^*_{\beta^*,j^*}}^{(\beta^*,j^*)}$. More in detail, we define a reduction R that, upon input a value $\hat{\boldsymbol{k}}$, runs exactly as $\mathsf{Hyb}^*\langle\langle \beta^*, j^* \rangle\rangle$, with the only difference of setting $\boldsymbol{k}_{i^*_{\beta^*,j^*}}^{(\beta^*,j^*)} \leftarrow \hat{\boldsymbol{k}}$. Then, R perfectly simulates $\mathsf{Hyb}^*\langle\langle \beta^*, j^*-1 \rangle\rangle$ if $\hat{\boldsymbol{k}}$ comes from the PRG, and perfectly simulates $\mathsf{Hyb}^*\langle\langle \beta^*, j^* \rangle\rangle$ if $\hat{\boldsymbol{k}}$ is uniformly sampled. Then, the advantage of any stateful quantum polynomial-time adversary A' distinguishing between R with PRG input and R with real input is bounded by ϵ_{PRG}. A final step is missing, to move from the situation where the shares are sampled from the original key $\boldsymbol{k}$ to the situation where they are sampled by $\mathsf{Sample}()$. Namely, after the reduction R, the shares are still sampled by sampling $\boldsymbol{k}_1^{(\beta^*,j^*)}, \ldots, \boldsymbol{k}_{d-1}^{(\beta^*,j^*)}$ first and then letting

$$\boldsymbol{k}_d^{(\beta^*,j^*)} = \boldsymbol{k} - \boldsymbol{k}_{i^*_{\beta^*,j^*}}^{(\beta^*,j^*)} - \sum_{i \in [d-1] \setminus \{i^*_{\beta^*,j^*}\}} G_{\mathcal{K}}\left(\boldsymbol{k}_i^{(\beta^*,j^*)}\right).$$

It is easy to see that this is equivalent to sampling all the shares except for $\boldsymbol{k}_{i^*_{\beta^*,j^*}}^{(\beta^*,j^*)}$, and then letting

$$\boldsymbol{k}_{i^*_{\beta^*,j^*}}^{(\beta^*,j^*)} = \boldsymbol{k} - \boldsymbol{k}_d^{(\beta^*,j^*)} - \sum_{i \in [d-1] \setminus \{i^*_{\beta^*,j^*}\}} G_{\mathcal{K}}\left(\boldsymbol{k}_i^{(\beta^*,j^*)}\right).$$

However, now we can use the same reasoning of the case $i^*_{\beta^*,j^*} = d$ since we don't reveal $\boldsymbol{k}_{i^*_{\beta^*,j^*}}^{(\beta^*,j^*)}$ anywhere and this is the only share that functionally depends on the key $\boldsymbol{k}$.

□

Lemma 14. *For uniform* $\boldsymbol{k}$*,*

$$\mathsf{Hyb}_3\left(\mathcal{D}_{\mathcal{X}\times\mathcal{W}}\right) \equiv \mathsf{Hyb}_4\left(\mathsf{LWR.Real}(\boldsymbol{k})\right).$$

Proof. The only difference between Hyb_3 and Hyb_4 is what is given as input and how $\boldsymbol{y}_{ext}$ is generated. Namely, the only two places where Hyb_3 uses the

input $\boldsymbol{k}$ are when $\boldsymbol{y}_{ext}$ is computed and when keys is computed. However, in the computation of keys, $\boldsymbol{k}$ is never used, since $\mathsf{Share}\langle\!\langle 0, 0\rangle\!\rangle$ never uses its input, while the value $\boldsymbol{y}_{ext}$ is computed exactly as in the real case. Therefore, if we give $\boldsymbol{y}_{ext}$ as input directly, we don't need $\boldsymbol{k}$ anymore, which is what happens in Hyb_4. □

Lemma 15. *For uniform $\boldsymbol{k}$, and every stateful quantum polynomial-time adversary* A*,*

$$|\mathbb{P}\left[\mathsf{A}\left(\mathsf{Hyb}_4\left(\mathsf{LWR.Real}(\boldsymbol{k})\right)\right)\right] - \mathbb{P}\left[\mathsf{A}\left(\mathsf{Hyb}_4\left(\mathsf{LWR.Rand}()\right)\right)\right]| \leq \epsilon_{\mathsf{LWR}}.$$

Proof. Assume not. Then, there exists A such that

$$|\mathbb{P}\left[\mathsf{A}\left(\mathsf{Hyb}_4\left(\mathsf{LWR.Real}(\boldsymbol{k})\right)\right)\right] - \mathbb{P}\left[\mathsf{A}\left(\mathsf{Hyb}_4\left(\mathsf{LWR.Rand}()\right)\right)\right]| > \epsilon_{\mathsf{LWR}},$$

which, in turns, means that there exists $\mathsf{A}' \coloneqq \mathsf{A} \circ \mathsf{Hyb}_4$ such that

$$|\mathbb{P}\left[\mathsf{A}'\left(\mathsf{LWR.Real}(\boldsymbol{k})\right)\right] - \mathbb{P}\left[\mathsf{A}'\left(\mathsf{LWR.Rand}()\right)\right]| > \epsilon_{\mathsf{LWR}},$$

thus breaking the LWR assumption. □

Lemma 16.

$$\mathsf{Hyb}_4\left(\mathsf{LWR.Rand}()\right) \equiv \mathsf{Sim}_1\left(\mathsf{LWR.Rand}()\right).$$

Proof. The only modification is that Sim_1, instead of receiving $(\boldsymbol{X}, \boldsymbol{y}_{ext})$, receives $(\boldsymbol{X}, \boldsymbol{y})$ and uniformly samples the missing α bits per component to get $\boldsymbol{y}_{ext}$. Since both samples $(\boldsymbol{X}, \boldsymbol{y}_{ext})$ and $(\boldsymbol{X}, \boldsymbol{y})$ come from the random distributions of the corresponding LWR instances, the value of $\boldsymbol{y}_{ext}$ in the two experiments is identically distributed. □

Lemma 17. *For uniform $\boldsymbol{k}$, and every stateful quantum polynomial-time adversary* A*,*

$$|\mathbb{P}\left[\mathsf{A}\left(\mathsf{Sim}_1\left(\mathsf{LWR.Rand}()\right)\right)\right] - \mathbb{P}\left[\mathsf{A}\left(\mathsf{Sim}_1\left(\mathsf{LWR.Real}(\boldsymbol{k})\right)\right)\right]| \leq \epsilon_{\mathsf{LWR}}.$$

Proof. Assume not. Then, there exists A such that

$$|\mathbb{P}\left[\mathsf{A}\left(\mathsf{Sim}_1\left(\mathsf{LWR.Rand}()\right)\right)\right] - \mathbb{P}\left[\mathsf{A}\left(\mathsf{Sim}_1\left(\mathsf{LWR.Real}(\boldsymbol{k})\right)\right)\right]| > \epsilon_{\mathsf{LWR}},$$

which, in turns, means that there exists $\mathsf{A}' \coloneqq \mathsf{A} \circ \mathsf{Sim}_1$ such that

$$|\mathbb{P}\left[\mathsf{A}'\left(\mathsf{LWR.Rand}()\right)\right] - \mathbb{P}\left[\mathsf{A}'\left(\mathsf{LWR.Real}(\boldsymbol{k})\right)\right]| > \epsilon_{\mathsf{LWR}},$$

thus breaking the LWR assumption. □

The following lemma is needed in order to step from the general indistinguishability of experiments to the indistinguishability of experiments conditioning on some non-negligible event happening.

Lemma 18. *Let $\mathcal{D}, \mathcal{D}'$ be two distributions over a set $\mathcal{X}$, let $\mathcal{S} \subseteq \mathcal{X}$, and let $\mathbf{E}$ (resp. $\mathbf{E}'$) be the event that $x \in \mathcal{S}$ when $x \leftarrow\!\!\$\, \mathcal{D}$ (resp. $x \leftarrow\!\!\$\, \mathcal{D}'$). Assume that $\mathbb{P}[\mathbf{E}] > 0$, $\mathbb{P}[\mathbf{E}'] > 0$, and, for some $\epsilon \in [0,1]$ and for every stateful quantum polynomial-time adversary A,*

$$|\mathbb{P}[\mathsf{A}(\mathcal{D}) = 1] - \mathbb{P}[\mathsf{A}(\mathcal{D}') = 1]| \leq \epsilon.$$

Finally, assume that $x \in \mathcal{S}$ is an efficiently verifiable statement. Then, for every stateful quantum polynomial-time adversary A',

$$|\mathbb{P}[\mathsf{A}'(\mathcal{D}) = 1 \mid \mathbf{E}] - \mathbb{P}[\mathsf{A}'(\mathcal{D}') = 1 \mid \mathbf{E}']| \leq \min\left\{\frac{2\epsilon}{\mathbb{P}[\mathbf{E}]}, \frac{2\epsilon}{\mathbb{P}[\mathbf{E}']}\right\}.$$

Furthermore, $|\mathbb{P}[\mathbf{E}] - \mathbb{P}[\mathbf{E}']| \leq \epsilon$.

Proof. First of all, we show that

$$|\mathbb{P}[\mathbf{E}] - \mathbb{P}[\mathbf{E}']| \leq \epsilon.$$

Indeed, assume that this is not the case, and consider an adversary A'' that outputs 1 if $x \in \mathcal{S}$ and 0 otherwise (this adversary exists because the statement $x \in \mathcal{S}$ is efficiently verifiable). Then,

$$\begin{aligned}
&|\mathbb{P}[\mathbf{E}] - \mathbb{P}[\mathbf{E}']| \\
&= |\mathbb{P}[\mathsf{A}''(\mathcal{D}) = 1] - \mathbb{P}[\mathsf{A}''(\mathcal{D}') = 1]| \\
&\leq \epsilon,
\end{aligned}$$

where the last step comes directly from the hypotesis.

From the above, we have that $\mathbb{P}[\mathbf{E}'] \leq \mathbb{P}[\mathbf{E}] + \epsilon$. Let $\mathsf{B} : \mathcal{X} \to \mathcal{X} \cup \{\perp\}$ be the function that maps every $x \in \mathcal{S}$ to $\perp$ and behaves like the identity otherwise. Then, for every stateful quantum polynomial-time adversary A',

$$\begin{aligned}
&|\mathbb{P}[\mathsf{A}'(\mathcal{D}) = 1] - \mathbb{P}[\mathsf{A}'(\mathcal{D}') = 1]| \\
&\geq |\mathbb{P}[\mathsf{A}'(\mathsf{B}(\mathcal{D})) = 1] - \mathbb{P}[\mathsf{A}'(\mathsf{B}(\mathcal{D}')) = 1]| \\
&= |\mathbb{P}[\mathsf{A}'(\mathsf{B}(\mathcal{D})) = 1 \wedge \mathbf{E}] - \mathbb{P}[\mathsf{A}'(\mathsf{B}(\mathcal{D}')) = 1 \wedge \mathbf{E}']| \\
&= |\mathbb{P}[\mathbf{E}]\,\mathbb{P}[\mathsf{A}'(\mathsf{B}(\mathcal{D})) = 1 \mid \mathbf{E}] - \mathbb{P}[\mathbf{E}']\,\mathbb{P}[\mathsf{A}'(\mathsf{B}(\mathcal{D}')) = 1 \mid \mathbf{E}']| \\
&= |\mathbb{P}[\mathbf{E}]\,\mathbb{P}[\mathsf{A}'(\mathcal{D}) = 1 \mid \mathbf{E}] - \mathbb{P}[\mathbf{E}']\,\mathbb{P}[\mathsf{A}'(\mathcal{D}') = 1 \mid \mathbf{E}']| \\
&\geq \mathbb{P}[\mathbf{E}]\,|\mathbb{P}[\mathsf{A}'(\mathcal{D}) = 1 \mid \mathbf{E}] - \mathbb{P}[\mathsf{A}'(\mathcal{D}') = 1 \mid \mathbf{E}']| - \epsilon,
\end{aligned}$$

and, by turning around the inequality and letting $\mathsf{A} = \mathsf{A}'$,

$$\begin{aligned}
&|\mathbb{P}[\mathsf{A}'(\mathcal{D}) = 1 \mid \mathbf{E}] - \mathbb{P}[\mathsf{A}'(\mathcal{D}') = 1 \mid \mathbf{E}']| \\
&\leq \frac{|\mathbb{P}[\mathsf{A}(\mathcal{D}) = 1] - \mathbb{P}[\mathsf{A}(\mathcal{D}') = 1]| + \epsilon}{\mathbb{P}[\mathbf{E}]} \\
&\leq \frac{2\epsilon}{\mathbb{P}[\mathbf{E}]}.
\end{aligned}$$

The proof for

$$|\mathbb{P}\left[\mathsf{A}'\left(\mathcal{D}\right)=1\,|\,\mathbf{E}\right]-\mathbb{P}\left[\mathsf{A}'\left(\mathcal{D}'\right)=1\,|\,\mathbf{E}'\right]|\leq\frac{2\epsilon}{\mathbb{P}\left[\mathbf{E}'\right]}$$

is analogous. □

The following lemma puts everything together, proving that Real and Sim_1 are computationally close when **Abort** does not happen.

Lemma 19. *For every stateful quantum polynomial-time adversary* A*,*

$$\begin{aligned}&|\mathbb{P}\left[\mathsf{A}\left(\mathsf{Real}\left(\mathcal{D}_{\mathcal{X}\times\mathcal{W}}\right)\right)\,|\,\mathbf{Success}\right]-\mathbb{P}\left[\mathsf{A}\left(\mathsf{Sim}_1\left(\mathcal{D}_{\mathcal{X}}\right)\right)\,|\,\mathbf{Success}\right]|\\&\leq Nt\epsilon_{\mathsf{Hide}}+\frac{2Nt\epsilon_{\mathsf{PRG}}+4\epsilon_{\mathsf{LWR}}}{1-mNt2^{-\alpha}}.\end{aligned}$$

Proof. First of all, notice that the only difference between Real and Hyb_0 is where the the keys, outputs, hash values, and challenges are computed. Namely, the real experiment computes each of the above in its own loop, while Hyb_0 computes everything in batch at the beginning of the experiment. Furthermore, the challenge i^* is given as input to $\mathsf{Output}\langle\!\langle 0\rangle\!\rangle$ and $\mathsf{H}\langle\!\langle N,t\rangle\!\rangle$, but none of them actually uses the indices. Therefore, the two experiments are identical:

$$\mathsf{Real}\left(\mathcal{D}_{\mathcal{X}\times\mathcal{W}}\right)\equiv\mathsf{Hyb}_0\left(\mathcal{D}_{\mathcal{X}\times\mathcal{W}}\right).\tag{36}$$

Moving on, the event **Success** does not depend nor affects the distribution of hashes and, therefore,

$$\begin{aligned}&|\mathbb{P}\left[\mathsf{A}\left(\mathsf{Hyb}_0\left(\mathcal{D}_{\mathcal{X}\times\mathcal{W}}\right)\right)\,|\,\mathbf{Success}\right]-\mathbb{P}\left[\mathsf{A}\left(\mathsf{Hyb}_1\left(\mathcal{D}_{\mathcal{X}\times\mathcal{W}}\right)\right)\,|\,\mathbf{Success}\right]|\\&=|\mathbb{P}\left[\mathsf{A}\left(\mathsf{Hyb}_0\left(\mathcal{D}_{\mathcal{X}\times\mathcal{W}}\right)\right)\right]-\mathbb{P}\left[\mathsf{A}\left(\mathsf{Hyb}_1\left(\mathcal{D}_{\mathcal{X}\times\mathcal{W}}\right)\right)\right]|\\&\leq Nt\epsilon_{\mathsf{Hide}},\end{aligned}\tag{37}$$

where the last inequality is a consequence of Lemma 11. Furthermore, by applying Lemmas 13 to 17 and the triangular inequality, we have that

$$|\mathbb{P}\left[\mathsf{A}\left(\mathsf{Hyb}_2\left(\mathcal{D}_{\mathcal{X}\times\mathcal{W}}\right)\right)\right]-\mathbb{P}\left[\mathsf{A}\left(\mathsf{Sim}_1\left(\mathcal{D}_{\mathcal{X}}\right)\right)\right]|\leq Nt\epsilon_{\mathsf{PRG}}+2\epsilon_{\mathsf{LWR}}.$$

Let $\mathbf{Success}'$ be the the event that **Abort** does not happen in $\mathsf{Sim}_1\left(\mathcal{D}_{\mathcal{X}}\right)$. Then, by Lemma 18, the above inequality becomes

$$\begin{aligned}&\left|\mathbb{P}\left[\mathsf{A}\left(\mathsf{Hyb}_2\left(\mathcal{D}_{\mathcal{X}\times\mathcal{W}}\right)\right)\,|\,\mathbf{Success}\right]-\mathbb{P}\left[\mathsf{A}\left(\mathsf{Sim}_1\left(\mathcal{D}_{\mathcal{X}}\right)\right)\,\middle|\,\mathbf{Success}'\right]\right|\\&\leq\frac{2Nt\epsilon_{\mathsf{PRG}}+4\epsilon_{\mathsf{LWR}}}{\mathbb{P}\left[\mathbf{Success}'\right]}.\end{aligned}\tag{38}$$

Notice that the event **Abort** happens whenever the first α bits of one of the components of the vector $\boldsymbol{y}_{ext}$ match the same bits of one of the components of any of the *sum* vectors generated in any of the Nt iterations. Let $\mathbf{Abort}_i$ be the event where **Abort** happens and is caused by the i-th component of the above

vectors. Since these α bits of $\boldsymbol{y}_{ext}$ are uniform in $\mathsf{Sim}_1(\mathcal{D}_{\mathcal{X}})$, the probability that the α bits of a single component cause $\mathbf{Abort}_i$ is

$$\mathbb{P}[\mathbf{Abort}_i] = \frac{Nt}{2^\alpha},$$

and, by the union bound,

$$\begin{aligned}\mathbb{P}[\mathbf{Abort}] &= \mathbb{P}\left[\bigvee_{i\in[m]} \mathbf{Abort}_i\right] \\ &\leq \sum_{i\in[m]} \mathbb{P}[\mathbf{Abort}_i] \\ &= m\frac{Nt}{2^\alpha}.\end{aligned}$$

Therefore, by complementing the above event, we get that

$$\mathbb{P}\left[\mathbf{Success}'\right] \geq 1 - mNt2^{-\alpha}. \tag{39}$$

Finally, the lemma follows by plugging together Eqs. (36) to (39) and Lemma 12 and observing that, since $\mathsf{Output}\langle\!\langle 0\rangle\!\rangle \equiv \mathsf{Output}\langle\!\langle 1\rangle\!\rangle$ when **Abort** does not happen, $\mathsf{Hyb}_1 \equiv \mathsf{Hyb}_2$ under the same condition. □

To conclude the proof, we just replace Sim_1 with Sim_ω, which tries to sample valid transcripts for ω before giving up, thus amplifying the probability of success.

Proof (of Theorem 5). It suffices to show that, for every $\omega > 0$ and every stateful quantum polynomial-time adversary A,

$$\left|\mathbb{P}[\mathsf{A}(\mathsf{Sim}_1(\mathcal{D}_{\mathcal{X}})) \mid \neg\mathbf{Abort}] - \mathbb{P}[\mathsf{A}(\mathsf{Sim}_\omega(\mathcal{D}_{\mathcal{X}}))]\right| \leq 2\left(mNt2^{-\alpha}\right)^\omega,$$

and the proof will follow by the triangular inequality and by the fact that

$$\lim_{\omega\to\infty} 2\left(mNt2^{-\alpha}\right)^\omega = 0.$$

In each attempt, the simulator Sim_ω discards everything and samples everything from scratch, causing the event **Abort** with probability at most $mNt2^{-\alpha}$. Therefore, the probability that, for ω consecutive attempts, **Abort** happens is upper bounded by

$$\left(mNt2^{-\alpha}\right)^\omega.$$

If the event **Abort** does not happen, Sim_1 and Sim_ω output a sample from the same distribution. The lemma follows by

$$\begin{aligned}&\left|\mathbb{P}[\mathsf{A}(\mathsf{Sim}_1(\mathcal{D}_{\mathcal{X}})) \mid \neg\mathbf{Abort}] - \mathbb{P}[\mathsf{A}(\mathsf{Sim}_\omega(\mathcal{D}_{\mathcal{X}}))]\right| \\ &\leq \left|\mathbb{P}[\mathsf{A}(\mathsf{Sim}_1(\mathcal{D}_{\mathcal{X}})) \mid \neg\mathbf{Abort}] - \mathbb{P}[\mathsf{A}(\mathsf{Sim}_\omega(\mathcal{D}_{\mathcal{X}})) \mid \neg\mathbf{Abort}]\,\mathbb{P}[\neg\mathbf{Abort}]\right| \\ &\quad + \mathbb{P}[\mathbf{Abort}]\end{aligned}$$

$$\leq |\mathbb{P}\left[\mathsf{A}\left(\mathsf{Sim}_1\left(\mathcal{D}_{\mathcal{X}}\right)\right) \mid \neg\mathbf{Abort}\right] - \mathbb{P}\left[\mathsf{A}\left(\mathsf{Sim}_\omega\left(\mathcal{D}_{\mathcal{X}}\right)\right) \mid \neg\mathbf{Abort}\right](1 - \mathbb{P}\left[\mathbf{Abort}\right])|$$
$$+ \mathbb{P}\left[\mathbf{Abort}\right]$$
$$\leq 2\mathbb{P}\left[\mathbf{Abort}\right] \leq 2\left(mNt2^{-\alpha}\right)^{\omega}.$$

This concludes the proof. □

E.3 Special Soundness (Theorem 6)

We begin with an outline of the proof intuition, and later provide a formal treatment. First of all, we isolate the event where the adversary succeeds in producing $s = 2$ accepting proofs for the same commitment and different challenges. Otherwise, there would be nothing to extract. Next, we observe that having 2 such transcripts provides the extractor with a complete sharing of a candidate witness $\boldsymbol{k}$. Therefore, extraction can be finalized by reconstructing $\boldsymbol{k}$ from its sharing. Finally, we prove that this extraction strategy always provides a witness for the given statement. To do it, we rely on the validity of each transcript with respect to the same statement and first message, and on the almost homomorphism of the rounding function.

Our extractor always reconstructs the sharing in the same way: $d-1$ shares from the first transcript, and the remaining share from the second. Notably, there can be many different ways to reconstruct the sharing: we could have also considered an extractor taking $d - i$ shares from the first transcript, and the remaining i shares from the second. In principle, all of them could result in a different witness candidate. Interestingly, all such cases would lead to a valid witness, for two main reasons. First, despite the opened key shares $\boldsymbol{k}_{i,1}$ and $\boldsymbol{k}_{i,2}$ can be distinct, they all verify with respect to the same first message, and in particular they all correspond to the same output shares $\boldsymbol{y}_i$. Moreover, our witness relation is defined at the output level, and therefore the specific choice of the reconstruction is not relevant as long as the output shares stay the same. For the formal analysis, we consider the extractor E defining special soundness described in Fig. 21.

In order to prove $(s, \epsilon_{\Sigma.\mathsf{SS}})$-special soundness for Σ, we need to upper bound with $\epsilon_{\Sigma.\mathsf{SS}}$ the quantity

$$\mathbb{P}\left[\Sigma.\mathsf{SS}(\mathsf{A}, \mathsf{E}) = 0\right].$$

where $\Sigma.\mathsf{SS}(\mathsf{A}, \mathsf{E})$ is the experiment defining special soundness (*cf.* Fig. 1). Let ASucc be the event where the adversary succeeds in creating s transcripts having distinct challenges and correctly verifying, *i.e.*

$$\mathsf{ASucc} = \left\{ \begin{array}{l} ch_1 \neq ch_2 \\ \Sigma.\mathsf{Verify}((\boldsymbol{X}, \boldsymbol{y}), \mathsf{tr}_c) = 1 \ \forall c \in [2] \end{array} \middle| \begin{array}{l} ((\boldsymbol{X}, \boldsymbol{y}), \mathsf{mtr}) \leftarrow\!\!\$\ \mathsf{A}(), \\ \boldsymbol{k} \leftarrow\!\!\$\ \mathsf{E}((\boldsymbol{X}, \boldsymbol{y}), \mathsf{mtr}) \end{array} \right\},$$

where the abbreviation mtr denotes the s transcripts produced by A, *i.e.*

$$\begin{aligned} \mathsf{mtr} &= (firstmes, (ch_c, resp_c)_{c\in\{1,2\}}) \\ &= ((\boldsymbol{y}_i^{(j)}, h_i^{(j)})_{i\in[d], j\in[t]}, ((\mathcal{I}_c^{(j)}), (\boldsymbol{k}_{i,c}^{(j)})_{i\in\mathcal{I}_c^{(j)}})_{j\in[t], c\in\{1,2\}}, \end{aligned}$$

$\mathsf{E}((\boldsymbol{X}, \boldsymbol{y}), \mathit{firstmes}, (ch_c, resp_c)_{c\in[s]})$

$j^* \leftarrow 0$
for $j \in [t]$
  for $c \in \{1, 2\}$
    $(\mathcal{I}_c^{(j)}) \leftarrow ch_c$
    $(\boldsymbol{k}_{i,c}^{(j)})_{i\in\mathcal{I}_c^{(j)}, j\in[t]} \leftarrow resp_c$
  if $\mathcal{I}_1^{(j)} \neq \mathcal{I}_2^{(j)}$
    $j^* = j$
if $j^* = 0$
  return $\perp$
for $i \in [d]$
  if $i \in \mathcal{I}_1^{(j^*)}$
    $\boldsymbol{k}_i \leftarrow \boldsymbol{k}_{i,1}^{(j^*)}$
  else
    $\boldsymbol{k}_i \leftarrow \boldsymbol{k}_{i,2}^{(j^*)}$
$\boldsymbol{k} \leftarrow \sum_{i\in[d-1]} G_{\mathcal{K}}(\boldsymbol{k}_i) + \boldsymbol{k}_d$
return $\boldsymbol{k}$

Fig. 21. The extractor E.

and tr_c denotes the c-th transcript $(\mathit{firstmes}, (ch_c, resp_c))$. Then,

$$\mathbb{P}\left[\Sigma.\mathsf{SS}(\mathsf{A},\mathsf{E})=0\right]=\mathbb{P}\left[\Sigma.\mathsf{SS}(\mathsf{A},\mathsf{E})=0,\mathsf{ASucc}\right]. \tag{40}$$

If $\mathbb{P}[\mathsf{ASucc}]=0$, then the theorem follows, as Eq. (40) is always smaller than $\mathbb{P}[\mathsf{ASucc}]$. Otherwise, we can use Bayes' Theorem and rewrite Eq. (40) as

$$\mathbb{P}\left[\Sigma.\mathsf{SS}(\mathsf{A},\mathsf{E})=0\right]=\mathbb{P}\left[\Sigma.\mathsf{SS}(\mathsf{A},\mathsf{E})=0|\mathsf{ASucc}\right]\cdot\mathbb{P}\left[\mathsf{ASucc}\right]. \tag{41}$$

In the rest of this proof, we show that

$$\mathbb{P}\left[\Sigma.\mathsf{SS}(\mathsf{A},\mathsf{E})=0|\mathsf{ASucc}\right]=0. \tag{42}$$

According to Eq. (41), this implies the theorem. Intuitively, Eq. (42) means that conditioned on the two challenges being distinct and the transcripts all accepting, then the extractor always outputs a valid witness. Eq. (42) comes from the following sequence of remarks.

1. Conditioning on ASucc guarantees that $ch_1 \neq ch_2$. Since each challenge ch_c consists of t subsets $(\mathcal{I}_c^{(j)})_{j\in[t]}$, then there must be at least one index j^* in which the subsets differ, *i.e.* $\mathcal{I}_1^{(j^*)} \neq \mathcal{I}_2^{(j^*)}$. Since those are two distincts subsets of $[d]$ with $d-1$ elements, then $\mathcal{I}_1^{(j^*)} \cup \mathcal{I}_2^{(j^*)} = [d]$. This implies that the extractor E is well defined and does not abort.
2. Using Item 1, we know that E produces d values $(\boldsymbol{k}_1,\ldots,\boldsymbol{k}_d)$ and computes $\boldsymbol{k}=\sum_{i\in[d-1]} G_{\mathcal{K}}(\boldsymbol{k}_i)+\boldsymbol{k}_d$. In other words, $(\boldsymbol{k}_1,\ldots,\boldsymbol{k}_d)$ is a sharing of the extracted key $\boldsymbol{k}$.
3. By combining Item 2, the almost key homomorphism of the rounding function (Proposition 1, Item 3), and the fact that the two first messages are equal, we derive
$$\lfloor \boldsymbol{X}\boldsymbol{k}\rfloor_{2^{q-\alpha}} - \tilde{\boldsymbol{y}}_1{}^{(j^*)} \in \{0,1\}, \tag{43}$$
where $\tilde{\boldsymbol{y}}_1{}^{(j^*)}$ denotes the value $\tilde{\boldsymbol{y}}^{(j^*)}$ reconstructed during the verification of tr_1. The latter equation follows a reasoning similar to the one made for proving Theorem 4, which we detail below. We define
$$x=\sum_{i\in\mathcal{I}^{(j^*)}} \mathsf{Output}(\boldsymbol{X},\boldsymbol{k}_{i,1}^{(j^*)},i), \quad x'=\mathsf{Output}(\boldsymbol{X},\boldsymbol{k}_{i^*,2}^{(j^*)},i^*).$$
Using 2, we can rewrite
$$\begin{aligned}\boldsymbol{X}\boldsymbol{k}&=\sum_{i\in[d-1]}\boldsymbol{X}G_{\mathcal{K}}(\boldsymbol{k}_i)+\boldsymbol{X}\boldsymbol{k}_d\\&=\sum_{i\in[d]}\mathsf{Output}(\boldsymbol{X},\boldsymbol{k}_i,i)\\&=\sum_{i\in\mathcal{I}^{(j^*)}}\mathsf{Output}(\boldsymbol{X},\boldsymbol{k}_{i,1}^{(j^*)},i)+\mathsf{Output}(\boldsymbol{X},\boldsymbol{k}_{i^*,2}^{(j^*)},i^*)\\&=\boldsymbol{x}+\boldsymbol{x}',\end{aligned}$$

where $i^* = [d] \setminus \mathcal{I}^{(j^*)}$. Therefore, the almost key homomorphism of the rounding (Proposition 1, Item 3) implies that

$$\lfloor \boldsymbol{X}\boldsymbol{k} \rfloor_{2^{q-\alpha}} - \lfloor \boldsymbol{x} \rfloor_{2^{q-\alpha}} - \lfloor \boldsymbol{x}' \rfloor_{2^{q-\alpha}} \in \{0,1\}^m. \tag{44}$$

To conclude the proof of the remark, we use two main observations. First, we denote with $sum_1^{(j^*)}$ the partial sum computed in the verification of tr_1, and rewrite

$$\lfloor \boldsymbol{x} \rfloor_{2^{q-\alpha}} = \left\lfloor \sum_{i \in \mathcal{I}^{(j^*)}} \mathsf{Output}(\boldsymbol{X}, \boldsymbol{k}_{i,1}^{(j^*)}, i) \right\rfloor_{2^{q-\alpha}} = \left\lfloor sum_1^{(j^*)} \right\rfloor_{2^{q-\alpha}}. \tag{45}$$

Next, we use the fact that the first message is the same for both transcripts, and they are both accepting. This implies that

$$\lfloor \boldsymbol{x}' \rfloor_{2^{q-\alpha}} = \left\lfloor \mathsf{Output}(\boldsymbol{X}, \boldsymbol{k}_{i^*,2}^{(j^*)}, i^*) \right\rfloor_{2^{q-\alpha}} = \boldsymbol{y}_{i^*} \tag{46}$$

where $\boldsymbol{y}_{i^*}$ is the i^*-th output share appearing in *firstmes*. Eq. (45) and Eq. (46) imply that

$$\lfloor \boldsymbol{x} \rfloor_{2^{q-\alpha}} + \lfloor \boldsymbol{x}' \rfloor_{2^{q-\alpha}} = \left\lfloor sum_1^{(j^*)} \right\rfloor_{2^{q-\alpha}} + \boldsymbol{y}_{i^*} = \tilde{\boldsymbol{y}}_1^{(j^*)}. \tag{47}$$

Combining Eq. (47) and Eq. (44), the remark follows.

4. Conditioning on ASucc guarantees that verification succeeds, and in particular that
$$\boldsymbol{y} - \tilde{\boldsymbol{y}}_1^{(j^*)} \in \{0,1\}^m.$$
5. Item 3 and 4 imply that $\boldsymbol{k}$ is a valid witness for $(\boldsymbol{X}, \boldsymbol{y})$, as
$$\boldsymbol{y} - \lfloor \boldsymbol{X}\boldsymbol{k} \rfloor_{2^{p-\alpha}} = (\boldsymbol{y} - \tilde{\boldsymbol{y}}_1^{(j^*)}) - (\lfloor \boldsymbol{X}\boldsymbol{k} \rfloor_{2^{p-\alpha}} - \tilde{\boldsymbol{y}}_1^{(j^*)}) \in \{-1, 0, +1\}^m.$$
This concludes the proof.

E.4 Quantum Computationally Unique Response (Theorem 7)

This proof is very similar to the one provided for Picnic1 in Corollary 26 of [47]. To prove quantum computationally unique responses, we must argue about the event where an adversary provides two different responses for the same first message and challenge, and about the detection of a measurement in the computational basis. Notably, the response of our sigma protocol only contains preimages of the hash function. Therefore, the QCUR property reduces to the collapsingness of H on each of the $(d-1)t$ opened values. Using the bound on the concurrent composition of collapsing functions (Lemma 4, [51]), the theorem follows.

E.5 Unpredictable Commitments (Theorem 8)

To prove commitment unpredictability, we must upper bound the following quantity

$$\mathbb{E}\left(\max_{\mathit{firstmes}^* \in \mathcal{N}_{\mathsf{com}}} \Pr[\mathit{firstmes} = \mathit{firstmes}^*]\right),$$

where the average ranges over the choice of the statement and witness pair, and the probability inside is taken over $\mathit{firstmes} \leftarrow\!\!\$\ \Sigma.\mathsf{Prove}_1(x, w)$. In what follows, we actually ignore the average and provide an upper bound for every statement and witness pair. Notably, for every statement $((\boldsymbol{X}, \boldsymbol{y}), \boldsymbol{k})$ and every first message $\mathit{firstmes}^*$, the probability that $\Sigma.\mathsf{Prove}_1$ generates exactly $\mathit{firstmes}^*$ as a commitment is upper-bounded by the probability that all the corresponding hash values are the same. In particular, this holds true for the $t(d-1)$ values generated with the PRG. Formally,

$$\Pr[\mathit{firstmes} = \mathit{firstmes}^*] \leq \Pr[h_i^{(j)} = h_i^{*(j)} \quad \forall (i,j) \in [d-1] \times [t]].$$

We now want to leverage the inherent collision resistance of collapsing hash functions. For $N \in [(d-1)t]$ and for every subset of N indices $\mathcal{S} \subseteq [d-1] \times [t]$, consider the event where only the key shares in $\mathcal{S}$ are different, *i.e.*

$$\mathsf{Event}_{\mathcal{S}} = \{\boldsymbol{k}_i^{(j)} \neq \boldsymbol{k}_i^{*(j)} \quad \forall (i,j) \in \mathcal{S} \text{ and } \boldsymbol{k}_i^{(j)} = \boldsymbol{k}_i^{*(j)} \text{ otherwise}\},$$

and let $\mathsf{Event}_{i,j}$ be the event where the specific couple (i,j) yields distinct key shares, *i.e.*

$$\mathsf{Event}_{i,j} = \{\boldsymbol{k}_i^{(j)} \neq \boldsymbol{k}_i^{*(j)}\}.$$

Using the law of total probabilities and the definition of conditional probability, we get

$$\begin{aligned}
&\Pr[\mathit{firstmes} = \mathit{firstmes}^*] \leq \Pr[h_i^{(j)} = h_i^{*(j)} \quad \forall (i,j) \in [d-1] \times [t]] \\
&= \sum_{N \in [(d-1)t]} \sum_{\substack{\mathcal{S} \subseteq [(d-1)] \times [t] \\ |\mathcal{S}| = N}} \Pr[h_i^{(j)} = h_i^{*(j)} \quad \forall (i,j) \in [d-1] \times [t], \mathsf{Event}_{\mathcal{S}}] \\
&= \sum_{N \in [(d-1)t]} \sum_{\substack{\mathcal{S} \subseteq [(d-1)] \times [t] \\ |\mathcal{S}| = N}} \Pr[h_i^{(j)} = h_i^{*(j)} \quad \forall (i,j) \in [d-1] \times [t] \mid \mathsf{Event}_{\mathcal{S}}] \\
&\qquad \cdot \Pr[\mathsf{Event}_{\mathcal{S}}]. \qquad (48)
\end{aligned}$$

In what follows, we analyze the terms of Eq. (48) separately. Starting with $\Pr[\mathsf{Event}_{\mathcal{S}}]$, we remark that $d-1$ shares per iteration correspond to $(d-1)t$ independently random sampled seeds for the PRG. Therefore, $\Pr[\mathsf{Event}_{\mathcal{S}}]$ satisfies

$$\Pr[\mathsf{Event}_{\mathcal{S}}] \leq \left(1 - \frac{1}{|\mathcal{P}_{\mathcal{K}}|}\right)^N \cdot \left(\frac{1}{|\mathcal{P}_{\mathcal{K}}|}\right)^{(d-1)t-N}.$$

We now study the term $\Pr[h_i^{(j)} = h_i^{*(j)} \quad \forall (i,j) \in [d-1] \times [t] \mid \mathsf{Event}_S]$. Using the independence of the key shares and the fact that equal key shares correspond to equal hash, we can rewrite it as

$$\begin{aligned}
&\Pr[h_i^{(j)} = h_i^{*(j)} \quad \forall (i,j) \in [d-1] \times [t] \mid \mathsf{Event}_S] \\
&= \prod_{(i,j) \in \mathcal{S}} \Pr[h_i^{(j)} = h_i^{*(j)} \mid \mathsf{Event}_{i,j}] \qquad (49) \\
&\leq (2\epsilon_{\mathsf{Coll}})^N, \qquad (50)
\end{aligned}$$

where the last inequality holds because collapsing hash functions are inherently collision-resistant (*cf.* Lemma 24 in [82]). Notably, the terms in Eq. (49) correspond to finding a collision for the hash function. According to Lemma 24 of [82], each of them can be upper bounded by $2\epsilon_{\mathsf{Coll}}$, yielding Eq. (50). Going back to Eq. (48), this yields

$$\begin{aligned}
&\Pr[\mathit{firstmes} = \mathit{firstmes}^*] \leq \Pr[h_i^{(j)} = h_i^{*(j)} \quad \forall (i,j) \in [d-1] \times [t]] \\
&= \sum_{N \in [(d-1)t]} \sum_{\substack{\mathcal{S} \subseteq [(d-1)] \times [t] \\ |\mathcal{S}| = N}} \Pr[h_i^{(j)} = h_i^{*(j)} \quad \forall (i,j) \in [d-1] \times [t] \mid \mathsf{Event}_S] \\
&\qquad \cdot \Pr[\mathsf{Event}_{\mathcal{S}}] \\
&\leq \sum_{N \in [(d-1)t]} \sum_{\substack{\mathcal{S} \subseteq [(d-1)] \times [t] \\ |\mathcal{S}| = N}} \left(1 - \frac{1}{|\mathcal{P}_\mathcal{K}|}\right)^N \cdot \left(\frac{1}{|\mathcal{P}_\mathcal{K}|}\right)^{(d-1)t-N} (2\epsilon_{\mathsf{Coll}})^N \\
&= \sum_{N \in [(d-1)t]} \binom{(d-1)t}{N} \cdot \left(1 - \frac{1}{|\mathcal{P}_\mathcal{K}|}\right)^N \cdot \left(\frac{1}{|\mathcal{P}_\mathcal{K}|}\right)^{(d-1)t-N} (2\epsilon_{\mathsf{Coll}})^N \\
&\leq \left(2\epsilon_{\mathsf{Coll}}\left(1 - \frac{1}{|\mathcal{P}_\mathcal{K}|}\right) + \frac{1}{|\mathcal{P}_\mathcal{K}|}\right)^{(d-1)t},
\end{aligned}$$

where the last inequality is true by the binomial theorem. The theorem follows.

F Albrecht et al.'s estimator security estimates

Table 9. Security estimates (in bits) against lattice attacks with $m = \infty$ using [5] (Commit `2bfb768`). Entries marked as ∞ indicate a security greater than 1000 bits.

NIST Security	q	(p', n)	coded BKW	uSVP	BDD	BDD hybrid	BDD mitm hybrid	dual	Size (Bytes) $d = 256$
L1	128	$(2, 6)$	392.60	144.01	144.88	144.88	299.20	159.01	7 817
		$(3, 9)$	521.01	138.45	138.45	138.45	292.76	154.22	6 510
		$(4, 12)$	649.33	135.67	135.67	135.67	265.81	145.74	6 919
	112	$(2, 6)$	344.41	128.73	129.60	129.60	258.52	142.62	7 164
		$(3, 10)$	456.82	135.12	135.12	135.12	283.64	150.84	6 401
		$(4, 13)$	569.14	129.62	129.84	129.84	254.05	139.00	6 692
	96	$(2, 7)$	296.19	128.73	129.60	129.60	258.52	142.62	7 164
		$(3, 11)$	392.60	128.73	128.73	128.73	265.55	143.65	6 183
		$(4, 16)$	585.19	135.67	135.67	135.67	265.81	145.74	6 919
	80	$(2, 10)$	328.34	149.01	149.88	149.88	312.95	164.02	8 035
		$(3, 14)$	408.67	135.12	135.12	135.12	283.64	150.84	6 401
		$(4, 19)$	569.15	134.56	134.56	134.56	262.48	144.30	6 874
L3	128	$(2, 9)$	521.01	205.31	206.18	206.18	469.51	222.33	17 563
		$(3, 14)$	649.34	203.36	203.39	203.39	483.65	224.59	14 949
		$(4, 19)$	905.82	202.52	201.12	201.12	461.21	219.93	16 084
	112	$(2, 10)$	456.82	200.29	200.87	200.87	454.93	217.02	17 237
		$(3, 16)$	681.41	203.36	203.39	203.39	483.65	224.59	14 949
		$(4, 21)$	∞	196.38	194.98	194.98	443.12	213.47	15 743
	96	$(2, 12)$	488.92	205.31	206.18	206.18	469.51	222.33	17 563
		$(3, 18)$	681.41	196.94	196.95	196.95	463.85	217.30	14 622
		$(4, 25)$	873.77	201.12	199.17	199.17	459.47	217.20	15 947
	80	$(2, 14)$	408.67	200.29	200.87	200.87	454.93	217.02	17 237
		$(3, 22)$	649.34	200.01	200.02	200.02	473.98	221.20	14 782
		$(4, 29)$	∞	194.15	192.75	192.75	435.72	210.74	15 609
L5	128	$(2, 12)$	649.33	266.83	267.42	267.42	648.64	284.75	31 200
		$(3, 19)$	905.82	268.51	269.38	269.38	684.09	296.82	26 842
		$(4, 25)$	∞	261.83	259.55	259.55	647.39	283.87	28 330
	112	$(2, 14)$	569.14	271.87	272.74	272.74	663.87	290.08	31 636
		$(3, 21)$	∞	260.39	261.26	261.26	658.08	288.10	26 300
		$(4, 29)$	∞	263.19	262.91	262.91	654.53	287.90	28 598
	96	$(2, 16)$	585.19	266.83	267.42	267.42	648.64	284.75	31 200
		$(3, 25)$	873.77	265.15	266.02	266.02	673.66	293.42	26 627
		$(4, 33)$	∞	259.39	257.03	257.03	640.07	281.14	28 147
	80	$(2, 19)$	569.15	264.31	264.90	264.90	641.03	282.49	30 985
		$(3, 29)$	∞	257.03	257.90	257.90	647.73	284.47	26 080
		$(4, 40)$	∞	261.83	259.55	259.55	647.39	283.87	28 330