SDitH in the QROM

Carlos Aguilar-Melchor[1], Andreas Hülsing[2]*, David Joseph[1], Christian Majenz[3]**, Eyal Ronen[4]***, and Dongze Yue[1]

[1] SandboxAQ, Palo Alto, USA, `firstname.lastname@sandboxaq.com`
[2] Eindhoven University of Technology, The Netherlands, `andreas@huelsing.net`
[3] Technical University of Denmark, Kgs. Lyngby, Denmark, `chmaj@dtu.dk`
[4] Tel Aviv University, Tel Aviv, Israel, `eyal.ronen@cs.tau.ac.il`

Abstract. The MPC in the Head (MPCitH) paradigm has recently led to significant improvements for signatures in the code-based setting. In this paper we consider some modifications to a recent twist of MPCitH, called Hypercube-MPCitH, that in the code-based setting provides the currently best known signature sizes. By compressing the Hypercube-MPCitH five-round code-based identification scheme into three-rounds we obtain two main benefits. On the one hand, it allows us to further develop recent techniques to provide a tight security proof in the quantum-accessible random oracle model (QROM), avoiding the catastrophic reduction losses incurred using generic QROM-results for Fiat-Shamir. On the other hand, we can reduce the already low-cost online part of the signature even further. In addition, we propose the use of proof-of-work techniques that allow to reduce the signature size. On the technical side, we develop generalizations of several QROM proof techniques and introduce a variant of the recently proposed extractable QROM.

Keywords: Post-quantum cryptography, code-based signatures, provable security, SDitH, MPCitH, QROM, QROM+, Fiat-Shamir.

1 Introduction

The advent of large scale quantum computers will render the security of virtually all public-key cryptography that is deployed today obsolete [28]. While it is an ongoing debate if and when such devices will be built (c.f., [18]) the potential impact would be so catastrophic, that betting on this never happening is not an option. For that reason, NIST initiated a competition to select future cryptographic standards for post-quantum secure signatures and key encapsulation, in 2016 [30]. In 2022, NIST selected of one KEM (Kyber) and three digital signature systems (Dilithium, Falcon, SPHINCS+) as the end of the third round of the competition [29]. However, the competition is not over, yet. While NIST is still about to select another KEM, there also seem to be good candidates from coding-theory [3, 1, 4]. On the other hand, the situation is worse for signatures. Dilithium [26] and Falcon [31] are both based on lattice-assumptions, and SPHINCS+ [20] while solely relying on the security of a cryptographic hash function, has significantly worse performance. Just before the selection, the last remaining candidates from multivariate cryptography were fatally attacked [6]. Consequently, there is currently a lack of signature proposals that are not based on lattice-assumptions and have good overall performance. For that reason, NIST started an "on-ramp" process for new signature proposals.

A promising area for new signature proposals is code-based cryptography which dates back to the work of McEliece [27]. Code-based cryptography grounds the security of construction in the hardness of decoding problems, like the *general decoding problem* or the *syndrome decoding problem.* Traditionally, code-based cryptography is rather well-known for public key encryption schemes. Proposals for signature schemes have also been known for a long time [32] but have

* Andreas Hülsing is supported by an NWO VIDI grant (Project No. VI.Vidi.193.066).
** Christian Majenz is supported by a NWO VENI grant (Project No. VI.Veni.192.159).
*** Eyal Ronen is partially supported by Len Blavatnik and the Blavatnik Family foundation, the Blavatnik ICRC, and Robert Bosch Technologies Israel Ltd.

never really been competitive. However, in recent years this area has received new interest with several new schemes proposed, like WAVE [10], and, most recently, Syndrome-Decoding in the Head (SDitH) [14]. SDitH is a new approach to code-based signatures that applies the MPCitH approach [22] to the Syndrome Decoding Problem to build an identification scheme (IDS). The latter is then turned into a signature scheme using the Fiat-Shamir heuristic [15]. MPCitH is a well known approach in post-quantum cryptography (PQC). Picnic [33], one of the long-standing contenders in the NIST competition was built on this approach. The SDitH authors manage to show that applying the MPCitH concept to a coding theory problem enables one to achieve better performance for the overall protocol. This performance has further been improved by a recent work [2] that proposes what they call Hypercube-MPCitH, to amplify the soundness of MPCitH in an efficient way, and apply it to the SDitH signature. We will call the resulting scheme the *Hypercube-SDitH* scheme.

The works proposing SDitH and Hypercube-SDitH come with security proofs. However, these security arguments only consider classical adversaries. This does not give a formal post-quantum security guarantee, especially because they use the Random Oracle Model (ROM) which is insufficient in that setting. An oracle modeling a hash function, a public primitive, needs to permit quantum queries, as an attacker can implement a hash function on a quantum computer. Hence the Quantum-accessible Random Oracle Model (QROM) was introduced [7]. It is now common practice to provide a QROM proof for post-quantum security.

Our contribution. In this work, we present a security proof for (a minor modification of) Hypercube-SDitH in the QROM. Our proof establishes the security of previously used parameters against quantum attacks at NIST security level 1 (the only parameter set considered for Hypercube-SDitH so far). For our proof we revist the Hypercube-SDitH and SDitH constructions. They build a 5-round IDS and turn this into a signature scheme. 5-round IDS are not that well understood and results about, e.g., the Fiat-Shamir transform are often only given for the canonical 3-round IDS. We notice that the IDS in both proposals can actually be viewed as 3-round IDS *in the (Q)ROM*.

On the one hand, this change in point of view increases the conceptual complexity of the scheme in two ways: i) The 3-round IDS needs to be constructed to readily include any parallel repetitions of the 5-round IDS. ii) While the 5-round IDS has statistical special soundness, the 3-round IDS only has *computational* special soundness, requiring additional work to prove security.

On the other hand, the change in point of view has several benefits. First of all, it allows making use of results for three-round IDS. In particular, a recent result about the security of commit-and-open IDS in the QROM [11], which is only given for three-round schemes, applies after a mild generalization. Second, for Hypercube-SDitH it was noticed that a huge part of the signing cost is caused by operations that do not depend on the message. This enables an online-offline trade-off in the sense of [13], where precomputation can be done during an offline phase to speed up signing during the online phase when messages to be signed become available. That way, it becomes easier feasible to deal with traffic peaks. With our observation, the balance shifts even more and the online phase can be reduced to a mere assembly of a signature from previously computed values. Finally, this enables a more modular proof than in previous approaches which hopefully makes the result more accessible.

Why is the reduction to three rounds possible? The previous proposals need two challenge rounds (and thereby five rounds total): one for a polynomial zero test that is used to probabilistically verify that the syndrome known by the prover / signer has low weight, and one for MPCitH. However, the first challenge is not necessary to achieve zero-knowledge. One indication for this is that the proofs in [14, 2] allow to extract a syndrome from two valid transcripts that agree in the initial three messages but differ in the fourth (the second challenge). In our analysis we proceed in two Fiat-Shamir steps. First, we make the polynomial zero test non-interactive. This step is secure unless an adversary can solve a certain random oracle search problem that we characterize in the ROM as well as in the QROM. This step leads to the advertised 3-round IDS. More precisely we prove a reduction from a family of computational special soundness properties of the 3-round IDS to a family of QROM oracle search problems. The second step constructs a digital signature. We thus analyse (some form of special) soundness and honest-verifier zero knowledge (HVZK) of the three round IDS. Based on these properties, we prove UF-CMA-security of the Signature scheme.

As mentioned, the QROM proof we obtain is clean and modular. We analyze HVZK in the multi-transcript setting, necessary when considering computational in place of statistical HVZK as is the case in Hypercube-SDitH. We prove security of the now non-interactive polynomial test. For this, we apply recent QROM lower bound methods by Chung, Fehr, Hsuan and Liao [9] based on Zhandry's compressed oracle technique [34]. We then prove a computational version of special-soundness in the QROM. Next, we develop a generalization of the recent result of [11] to the case of computational special soundness, and apply them towards security for Fiat-Shamir transformed IDS under no-message attacks. For this last step we introduce the QROM+, a model similar to the extractable QROM as recently defined in [19] (which maybe of independent interest), and develop an extension of the techniques from [9] to the QROM+. The QROM+ serves as a proof tool: it allows us to generalize a common, modular proof strategy, where intermediate algorithms require to learn the preimages of certain queries to the QROM. To eventually obtain a security bound that does not refer to the QROM+, we prove an explicit bound for the adversarial advantage against computational special soundness in the QROM+. Finally, we extend the result to security under chosen message attacks using the adaptive reprogramming technique from [17].

Besides the change in point of view that allows for improved analysis including a QROM proof, we also make some actual modifications to the scheme. We note that we can optimize the computation cost without increasing the signature size. This is done in a counter-intuitive way: it turns out that by increasing the communication cost of the IDS sending certain data in the clear instead of just a commitment, we can reduce the signing cost. The considered data are the communication transcripts and final outputs of the MPC parties which will all be revealed eventually. We assume that this information was previously sent in committed form to optimize communication cost of the IDS. However, as the data is recomputable from the opening information provided in the last message, it does not have to be included in this first message. As a side effect, this simplifies the structure of the protocol and the security proof.

Finally, we present performance numbers of our proposed signature scheme. The total signature times are comparable to the original Hypercube-SDitH, but most of the computational cost is moved from online time to offline time. In addition, we also show that it is possible to make use of a proof-of-work (PoW) technique similar to the recently proposed SPHINCS+C in [24] to decrease signature sizes by minimally increasing signing and verification times.

Outline. We discuss the identification scheme in Sec. 2, including necessary background, a summary of the ideas behind SDitH, and Hypercube-SDitH (A table of symbols is available in the Supp.Mat. A). We analyze the security of the IDS in Sec. 3. In Sec. 4, we discuss the signature scheme and its security. Finally, in Sec. 5 we provide performance results and the PoW trick.

2 SDitH as a 3-Round Identification Scheme

In this section we present SDitH and the hypercube variant thereof as a three round, public coin, commit and open identification scheme (IDS). We first provide background on the cryptographic tools used in our construction. Afterwards, we give a high level intuition of the scheme, before we end with a detailed, modular description of the IDS.

2.1 Preliminaries

In the following we provide the definitions for a PRG, a TreePRG, commitments, and identification schemes. At the end of the section we introduce the syndrome decoding problem we use as hardness assumption.

PRG. A pseudorandom generator (PRG) is an efficiently computable function $\mathsf{PRG} : \{0,1\}^n \to \{0,1\}^{en}$ where e is the expansion factor. Security of a PRG is defined in terms of a real-or-random game. The advantage of a possibly quantum adversary A is defined as

$$\mathrm{Adv}^{\mathsf{ror}}_{\mathsf{PRG}}(\mathsf{A}) := |\Pr[x \leftarrow \{0,1\}^{en} : 1 \leftarrow \mathsf{A}(x)] - \Pr[x \leftarrow \{0,1\}^n : 1 \leftarrow \mathsf{A}(\mathsf{PRG}(x))]| .$$

TreePRG. In this work we make use of a specific PRG called $\mathsf{TreePRG}$, initially proposed by Goldreich, Goldwasser, and Micali [16]. $\mathsf{TreePRG}$ makes use of a standard PRG with expansion factor $e = 2$ and reaches $e = 2^\lambda$ building a binary tree of height λ. The root of the tree is the input and the leaves are the outputs. To build the tree, every inner node is fed to PRG to generate its two child nodes. Let Outi denote the ith leaf / output block of $\mathsf{TreePRG}$. We define as $\mathsf{TP.extract}$ the function that given a seed x and an index i returns the sibling path for $\mathsf{Out_i}$, i.e., the minimal set of inner nodes that allows to compute all Out values except $\mathsf{Out_i}$. For our construction we require that $\mathsf{Out_i}$ is pseudorandom even when given the output of $\mathsf{TP.extract}(x, i)$. We define an even stronger notion as it is easily achievable: For a possibly quantum adversary A we define the advantage against $\mathsf{TreePRG}$ as

$$\begin{aligned}\mathrm{Adv}^{\mathsf{ror}}_{\mathsf{TreePRG}}(\mathsf{A}) := \Big|&\Pr[\{x_j\}_{j=0}^{\lambda} \leftarrow (\{0,1\}^n)^\lambda : 1 \leftarrow \mathsf{A}(\{x_j\}_{j=0}^{\lambda})] \\ &- \Pr[x, y \leftarrow \{0,1\}^n : 1 \leftarrow \mathsf{A}(\mathsf{TP.extract}(x,i), \mathsf{Out_i}))]\Big| .\end{aligned}$$

A standard hybrid argument can be used to show that $\mathrm{Adv}^{\mathsf{ror}}_{\mathsf{TreePRG}}(\mathsf{A}) \le (\lambda - 1)\mathrm{Adv}^{\mathsf{ror}}_{\mathsf{PRG}}(\mathsf{B})$ where $\mathsf{TIME}(\mathsf{B}) \le \mathsf{TIME}(\mathsf{A}) + (\lambda - 1)\mathsf{TIME}(\mathsf{PRG})$: One replaces the outputs on the path to leaf i by random values, one by one. The beginning is the real case (right probability above). Once all outputs on the pathare replaced, we get the random case (left probability above). The computational distance between any two consecutive hybrids is bounded by $\mathrm{Adv}^{\mathsf{ror}}_{\mathsf{PRG}}(\mathsf{B})$ where B replaces the outputs where the two hybrids differ by its input and then runs A.

Com. In this work we consider only hash-based commitments. Hence, we define commitment scheme as an algorithm Com that given an input x and randomness $\rho \in \{0,1\}^r$ produces a commitment $\mathsf{com} = \mathsf{Com}(x; \rho) \in \{0,1\}^c$. We make the randomness explicit as given (com, x, ρ) everybody can check that indeed $\mathsf{com} = \mathsf{Com}(x; \rho)$. From our commitment schemes we require two properties: binding and hiding.

We define the advantage of a possibly quantum adversary A against the computational binding property of Com as

$$\mathrm{Adv}^{\mathsf{bind}}_{\mathsf{Com}}(\mathsf{A}) := \Pr[((x_1, \rho_1), (x_2, \rho_2)) \leftarrow \mathsf{A} : \mathsf{Com}(x_1; \rho_1) = \mathsf{Com}(x_2; \rho_2)].$$

We define the advantage of a possibly quantum adversary A against the computational hiding property of Com as

$$\begin{aligned}\mathrm{Adv}^{\mathsf{hide}}_{\mathsf{Com}}(\mathsf{A}) := \Big|&\Pr[((x_1,x_2) \leftarrow \mathsf{A}; \rho \leftarrow \{0,1\}^k : 1 \leftarrow \mathsf{A}(\mathsf{Com}(x_1;\rho)] \\ &- \Pr[((x_1,x_2) \leftarrow \mathsf{A}; \rho \leftarrow \{0,1\}^k : 1 \leftarrow \mathsf{A}(\mathsf{Com}(x_2;\rho)]\Big| .\end{aligned}$$

2.1.1 Identification Schemes.

In this work we are concerned with 3-round, public coin, commit and open identification schemes which we will denote by IDS. An IDS is an interactive protocol between a prover P and a verifier V. It is defined by a tuple of algorithms $(\mathsf{Keygen}, \mathsf{Commit}, \mathsf{Resp}, \mathsf{Vrf})$ and a challenge space $\mathcal{C}$. Prior to any interaction, Keygen is run and outputs a key pair $(\mathsf{pk}, \mathsf{sk})$. A protocol run starts with P running $(\mathsf{st}, \mathsf{w}) \leftarrow \mathsf{Commit}(\mathsf{sk})$. The commitment message w is sent to V which samples a challenge c from the uniform distribution over $\mathcal{C}$ and sends it to P. Upon receiving c, the prover P runs $\mathsf{z} \leftarrow \mathsf{Resp}(\mathsf{st}, \mathsf{c})$ and sends z to V. The verifier accepts if $\mathsf{Vrf}(\mathsf{pk}, \mathsf{w}, \mathsf{c}, \mathsf{z}) = 1$ and rejects otherwise.

The *transcript* of a run of the IDS is the tuple $(\mathsf{w}, \mathsf{c}, \mathsf{z})$ of messages exchanged. We are only interested in IDS that are *correct*, i.e., for any key pair output by Keygen, we want that the execution of IDS between honest P and V always accepts. A property that can be handy when turning IDS into signatures is that of *commitment-recoverable* IDS. An IDS is commitment recoverable if there exists an algorithm Rcvr, such that for any valid transcript $(\mathsf{w}, \mathsf{c}, \mathsf{z})$, we have $\mathsf{Rcvr}(\mathsf{c}, \mathsf{z}) = \mathsf{w}$.

We expect IDS to provide two security propertes which are defined below.

HVZK. The most commonly used version of honest-verifer zero-knowledge (HVZK) is the statistical version. This version has the advantage that it trivially also gives a bound for multiple transcripts. However, in our setting where we use hash-based commitments the amount of commitment randomness required to achieve statistical HVZK in place of computational HVZK is greater by a factor 2.5 as shown in [25]. This has a huge impact on signature size and so we aim only at computational HVZK. As pointed out in [17], deriving a bound for HVZK of multiple transcripts is not straight-forward when in the computational setting. Hence, we directly prove multiple transcript, computational HVZK below. To define this property, we first have to define an honest transcript generator Trans and an HVZK simulator Sim. In our definitions we closely follow [17] as we later use the HVZK property in a result of that work.

Definition 1 (HVZK simulator and honest transcript generator). *An* HVZK simulator *for* IDS *is an algorithm* Sim *that takes as input the public key* pk *and outputs a transcript* $(\mathsf{w}, \mathsf{c}, \mathsf{z})$*. An* honest transcript generator *for* IDS *is an algorithm* Trans *that takes as input the secret key* sk *and outputs a transcript* $(\mathsf{w}, \mathsf{c}, \mathsf{z})$ *by means of an honest execution of* IDS*.*

Based on this definition we can define computational t-HVZK of an IDS as follows:

Definition 2 (Computational t-HVZK). *We define the advantage of a possibly quantum adversary* A *against the computational* t-HVZK *of* IDS *with simulator* Sim*, making no more than* t *queries to its (transcript-)oracle as*

$$\begin{aligned}\mathrm{Adv}^{t-\mathsf{HVZK}}_{\mathsf{IDS},\mathsf{Sim}}(\mathsf{A}) := \Big|&\Pr[(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Keygen}() : 1 \leftarrow \mathsf{A}^{\mathsf{Sim}(\mathsf{pk})}(\mathsf{pk})] \\ &- \Pr[(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Keygen}() : 1 \leftarrow \mathsf{A}^{\mathsf{Trans}(\mathsf{sk})}(\mathsf{pk})]\Big| .\end{aligned}$$

Special Soundness. Also for special soundness (spS) we slightly deviate from the common definition. The reason is again that statistical special soundness would be too expensive in terms of signature size (requiring length preserving commitments). Moreover, as we turn the five- into a three-round protocol it becomes inherently impossible to achieve statistical special soundness: the polynomial test can now be cheated by solving a search problem for the hash function. Bounding the hardness of this search problem will be a large part of the spS-proof.

It turns out that we need an even more fine-grained notion of special soundness as we are considering a τ-fold parallel-composition of some basic IDS'. Looking ahead, in the case we are interested in, IDS is the parallel repetition of the five-round identification scheme considered in [2], with the Fiat-Shamir transform for proof systems applied to the first three rounds. As an abstraction of this parallel composition, we say IDS has a *splittable challenge* if a challenge c of IDS has form $\mathsf{c} = (\mathsf{c}_1, \ldots, \mathsf{c}_\tau)$, where c_i are challenges of IDS'. We let the distance between two IDS challenges $\mathsf{Dist}(\mathsf{c}_1, \mathsf{c}_2)$ as the number of IDS' challenges on which they disagree, i.e., the number of indices $1 \leq i \leq \tau$ for which $(\mathsf{c}_1)_i \neq (\mathsf{c}_2)_i$.

Definition 3 ((Query-bounded) distance-d special soundness for IDS with splittable challenge). *We define the advantage of a possibly quantum adversary A against the query bounded special soundness of a composed IDS with respect to extractor Ext in the (quantum-accessible) random oracle model as follows*

$$\begin{aligned}\mathrm{Adv}^{d-\mathsf{spS}}_{\mathsf{IDS},\mathsf{Ext}}(\mathsf{A}) := \Pr[&(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Keygen}(); ((\mathsf{w}_1, \mathsf{c}_1, \mathsf{z}_1), (\mathsf{w}_2, \mathsf{c}_2, \mathsf{z}_2)) \leftarrow \mathsf{A}^{\mathsf{RO}}(\mathsf{pk});\\ &\mathsf{sk}' \leftarrow \mathsf{Ext}^{\mathsf{RO}}((\mathsf{w}_1, \mathsf{c}_1, \mathsf{z}_1), (\mathsf{w}_2, \mathsf{c}_2, \mathsf{z}_2)) : \mathsf{Vrf}(\mathsf{pk}, \mathsf{w}_i, \mathsf{c}_i, \mathsf{z}_i) = 1, i \in \{1, 2\}\\ &\wedge (\mathsf{w}_1 = \mathsf{w}_2) \wedge d = \mathsf{Dist}(\mathsf{c}_1, \mathsf{c}_2) \wedge (\mathsf{sk}', \mathsf{pk}) \notin \mathsf{Keygen}()],\end{aligned}$$

where q is the maximum number of queries that A makes to RO and we consider it understood that in this case all IDS algorithms may depend on RO.

Syndrome Decoding. The hardness assumption that we use in this work is that of the Coset Weights variant of the Syndrome Decoding (SD) problem, shown to be NP-complete [5].

Definition 4 (Coset Weights Syndrome Decoding problem). *Sample a uniformly random parity check matrix $\mathbf{H} \in \mathbb{F}_{SD}^{(m-k)\times m}$, and binary vector $\mathbf{x} \in \mathbb{F}_{SD}^{m}$ with $wt(\mathbf{x}) = \omega$. Let syndrome $\mathbf{y} = \mathbf{H}\mathbf{x}$. Then given only $\mathbf{H}, \mathbf{y}$, it is difficult to find $\mathbf{x}' \in \mathbb{F}_{SD}^{m}$ such that $\mathbf{H}\mathbf{x}' = \mathbf{y}$ with $wt(\mathbf{x}') \leq \omega$.*

Furthermore, for cryptographically relevant parameters, with overwhelming likelihood there exists only one short preimage of weight $\leq \omega$, and that is the $\mathbf{x}$ sampled initially.

2.2 SDitH and the hypercube approach

In the following we summarize the previous works that we build on. We first briefly sketch the MPC in the Head (MPCitH) paradigm [22]. Then we discuss the work syndrome decoding in the head [14], and finally a recent extension to that work called the hypercube approach [2].

MPCitH. The MPCitH approach is a technique to build a zero-knowledge proof (ZKP) by simulating an MPC computation *in the head* and building on the security properties of the MPC protocol. More precisely, MPCitH can be used to prove knowledge of some x such that $F(x) = \mathtt{ACCEPT}$ for a function F that outputs either $\mathtt{ACCEPT}$ or $\mathtt{REJECT}$ in zero-knowledge. Roughly, the protocol works as follows. The input x is secret shared among all parties (we limit ourselves here to additive secret sharing over a finite field) and the MPC protocol is used to evaluate F on this shared x. For this the MPC protocol would exchange messages between parties to implement multiplications of secret shared data while linear operations can be done locally by every party on their shares. Finally, all parties output their secret share of the result which can be summed up to get the result.

To turn this into a ZKP, in MPCitH the prover P first does the secret sharing and then executes the MPC protocol for all parties to compute the communication transcript of in- and outgoing messages for each party, as well as the secret share of the result. Then, P commits to the view of all parties which contains the initial secret share, their random tape, and communication transcripts. The commitments together with all secret shares of the result are sent to the verifier V. In response, V sends a random number i between 1 and t. As last message, P then sends the openings for the views of all parties but the ith. (We limit ourselves to the case where all but one state are opened. In general, less than $t-1$ parties might be opened.) For verification, V checks the views of all opened parties, making sure that the communications agree with the initial state and both together lead to the secret share of the result for this party.

Intuitively, zero-knowledge is obtained due to the privacy of the MPC protocol and one party not being opened. Soundness is obtained by the correctness of the MPC protocol, and the observation that a P that does not know x can at most compute $t-1$ consistent views. Consequently, the view of one party has to be inconsistent which is observed by V with probability $1-t^{-1}$.

SDitH. In [14], an application of MPCitH to the syndrome decoding problem is proposed. Intuitively, it is clear that we can use MPCitH to prove knowledge of an $\mathbf{x}$ such that $\mathbf{Hx} = \mathbf{y}$ setting $F(\mathbf{x}) := \mathbf{Hx} - \mathbf{y}$ and defining 0 to indicate `ACCEPT` and any other value to indicate `REJECT`. The problem is that this does not guarantee that $wt(\mathbf{x}) \leq \omega$.

The crucial novelty in [14] is to overcome this problem by proposing a weight check routine which we describe in detail later. Roughly, this routine computes a polynomial S from $\mathbf{x}$ as well as some other polynomials Q, P, and F such that $S \cdot Q - P \cdot F = 0$ iff $wt(\mathbf{x}) \leq \omega$. This equation is then probabilistically checked on a set of random points, chosen by the verifier (which makes their protocol five-round). To avoid running two MPCitH instances, the authors link the two as follows. They consider only $\mathbf{H} = (\mathbf{H}'|\mathbf{I})$ given in standard form. This means, one can split $\mathbf{x} = (\mathbf{x}_A|\mathbf{x}_B)$ such that $\mathbf{y} = \mathbf{H}'\mathbf{x}_A + \mathbf{x}_B$. Exploiting this, they store only $\mathbf{x}_A$ as secret and start the weight check routine by recomputing $\mathbf{x}_B = \mathbf{y} - \mathbf{H}'\mathbf{x}_A$, and then deriving S from the recombined $\mathbf{x} = (\mathbf{x}_A|\mathbf{x}_B)$. Thereby, this extended weight check also verifies that $\mathbf{Hx} = \mathbf{y}$.

The final protocol is then obtained applying MPCitH to F being the extended weight check function that starts with $\mathbf{x}_A$. The protocol deviates from the basic MPCitH recipe as it also obtains the random evaluation points from V. For this, P generates the initial secret shares of $\mathbf{x}_A$, Q, and P (F is public and the secret share of S is derived from the secret share of $\mathbf{x}_A$). It commits to all of this and sends it to the verifier that responds with the random evaluation points (and some values necessary to deal with multiplication of shares). Then P can simulate the MPC computation of F and the protocol from there on follows the standard MPCitH receipt.

A standard optimization. One way to reduce the size of the opening information above is based on the properties of additive secret sharing. Namely, the initial state of all parties consists of a secret sharing of the secret x and in case of SDitH also of some secret sharings of further values (the polynomials Q, and P, as well as values needed to do multiplication of shares). Generation of the secret shares in additive secret sharing can be done by picking the first $t-1$ shares at random and then computing the final share as the difference of the shared value and the sum of the $t-1$ shares. The first $t-1$ shares can hence be replaced by short random seeds which are expanded to the full shares using a PRG. These seeds can be bitstrings of the length of the security parameter while the secret shares in the above protocols are significantly longer.

Given that we commonly send the shares together with the commitment randomness as opening information, this massively reduces communication cost as we now only have to send the seeds to open parties 1 to $t-1$. Only for party t we are unable to compress the opening information. We call this the auxiliary state aux.

A further way of optimizing communication cost can be achieved using $\mathsf{TreePRG}$. Instead of sampling $t-1$ random seeds for the initial secret sharing, these seeds are generated using $\mathsf{TreePRG}$. This allows to open all seeds with $\log_2(t-1)$ values and if $i \neq t$ (using $\mathsf{TP.extract}$) and with just a single value in case $i = t$ (in the former case we still have to send along the full auxiliary state aux to open that one). This reduces the biggest part of the communication cost from linear in the number of parties to logarithmic.

Hypercubes. In a recent work [2], an improvement to the SDitH protocol is proposed that allows to boost soundness in a size efficient way. The protocols above have a soundness error of $1/t$. To achieve a negligible soundness error, we require amplification. There are two common ways to go for a soundness error of $t^{-\tau}$: First, we can simply increase the number of parties to t^{τ} which comes with an exponential increase in runtime and communication cost but the number of aux states remains the same. Second, we can run τ iterations of the protocol in parallel at the cost of a τ fold increase in communication, especially, we get τ aux states, but also only a τ fold increase in runtime (compared to the exponential increase).

The improvement proposed in [2] is the hypercube approach. The idea is to generate an N^D secret sharing of the initial state values, i.e., all the values that are secret shared for the initial states of parties. This means we get a single auxiliary state. Then, these shares are used to create D instances of the MPC protocol with N parties each. For each of these D instances, they partition the N^D shares into N subsets of D shares each and recombine the secret shares in each subset by summing them up. This recombination results in N secret shares of the shared values, i.e., the initial states of N parties necessary for the MPC protocol. This is related to a hypercube as the partitioning is done by arranging the original N^D shares in a hypercube and for each of the D partitions we recombine by projection onto one of the D dimensions. The protocol essentially then runs the SDitH protocol for D instances with a few little differences. First, P commits to all the N^D initial shares independently instead of committing to the shares of the parties in the actual MPC protocol. This is intuitively fine as P thereby still commits to all the information. Second, V still only picks one index i now between 1 and N^D. Then P opens all secret shares of the original N^D shares except share i (which is possible because of the independent commitments). Due to the properties of the partitioning and the secret sharing scheme, this means that in each protocol instance, there is one party for which the initial state remains unopened as one share is lacking for the partial recombination.

All in all, this approach allows us to achieve the best of both worlds: We get the soundness error N^{-D} at computational and communication cost of parallel composition (D parallel repetitions of the N party protocol), while we get just one auxiliary state as if we had increased the number of protocol parties to N^D (as pointed out in [2], there are more computational improvements possible when looking at the details, like balancing the party preparation phase and the MPC phase; for those we refer to [2]).

2.3 Polynomial zero test

In the identification schemes presented here, the prover P gives a zero knowledge proof (ZKP) that he knows a solution $\mathbf{x} \in \mathbb{F}_{SD}^m$ to the syndrome decoding problem,i.e., such that $\mathbf{Hx} = \mathbf{y}$ with $wt(\mathbf{x}) \leq \omega$. In order to do this, P constructs four polynomials S, Q, P, F in $\mathbb{F}_{\text{poly}}[X]$ which should satisfy the relation $S \cdot Q = P \cdot F$, and the ZKP proceeds by checking the relation is true at various points in a space $\mathbb{F}_{\text{points}} \supseteq \mathbb{F}_{\text{poly}}$.

Let $\phi \colon \mathbb{F}_{SD} \to \mathbb{F}_{\text{poly}}$ be the canonical embedding. Then S is computed by interpolating over the coordinates of $\mathbf{x}$. That is, $S(f_i) = \phi(x_i)$ where f_i runs over the first m elements of $\mathbb{F}_{\text{poly}}$, so $deg(S) \leq m-1$ as S is the interpolation over the m coordinates of $\mathbf{x}$. Next, Q is $\prod_{f_i \in E}(X - f_i)$, where E is a set of order ω which contains the nonzero coordinates of S. Thus, the nonzero points of S are all roots of $Q[X]$ which has degree ω. Polynomial F is public, and is $F[X] = \prod_{[m]}(X - f_i)$, meaning it has roots everywhere in the first m coordinates of $\mathbb{F}_{\text{poly}}$. And finally, P is defined as $S \cdot Q/F$, in order to ensure that both sides of the relation have the same degree, which is $\leq m + \omega - 1$.

Checking the polynomial is not done by directly checking the polynomials, but implicitly by checking that the polynomial relation is true at several points $r \in \mathbb{F}_{\text{points}}$. This is because if two polynomials are equal, then they will be equal at every point at which they are evaluated, however if they are not equal, then it becomes increasingly unlikely that they will be equal if we check them at an increasing number t of randomly selected points, by the Schwarz-Zippel lemma [14]. When selecting points at which to evaluate the polynomials, we draw from a larger domain $\mathbb{F}_{\text{points}} \supseteq \mathbb{F}_{\text{poly}}$, in order to make it harder to find points at random where non-equivalent polynomials coincide.

In summary, when evaluated on the first m coordinates of $\mathbb{F}_{\text{poly}}$, S has zeros everywhere except the ω nonzero coordinates of $\mathbf{x}$; Q has zeroes everywhere that S does not by construction, F has zeroes everywhere, and P serves to make left and right hand sides equal. Any party that knows a valid solution to the Coset Weights SD problem can therefore build polynomials S, Q, P that satisfy this relation. Note that a party who can solve the SD problem and finds $\mathbf{x}'$ such that $wt(\mathbf{x}') < \omega$ would also be able to construct a valid but different set of S, Q, P.

2.4 Protocol formulation

SDitH and Hypercube-SDitH are presented as five round IDS. Here we give a description as three-round IDS. The advantage of observing that they can be turned into three-round IDS, is threefold. First, it reduces the number of interactions between parties. Second, when turning it into a signature scheme using the Fiat-Shamir transform, we can apply the tight QROM proof recently introduced in [11] which only applies to three round IDS. Third, more of the computation done during signature generation is independent of the message, thus can be precomputed. Indeed, the required online computation consists merely of computing a hash and assembling a message from the local state.

In the following we describe the protocol in terms of the different steps it encompasses. For a full picture of the protocol see Algorithms 1 and 2. We give our description for $\tau = 1$ and explain a detailed change to the standard parallel compoisition for $\tau > 1$ afterwards.

Parameters. Hypercube-SDitH has the following building blocks and parameters. The seed length for the used PRG is n. We assume that PRG can produce an arbitrary number of n byte output blocks and we truncate to the required amount. Commitments take r bits of randomness and produce commitments of length c. It uses a hypercube of dimension D and N parties per MPC computation. We use parallel composition of τ instances to reduce the soundness error. Finally, the parameters of the syndrome decoding problem are m, k, and ω.

Key Generation. Prover P samples $\mathbf{H}' \xleftarrow{\$} \mathbb{F}_{\text{SD}}^{(m-k)\times k}$ and $\mathbf{x} \xleftarrow{\$} \mathbb{F}_{\text{SD},\omega}^{m}$ where $\mathbb{F}_{\text{SD},\omega}^{m}$ is the set of all elements $\mathbf{a} \in \mathbb{F}_{\text{SD}}^{m}$ with $wt(\mathbf{a}) = \omega$. It splits $\mathbf{x} = (\mathbf{x}_A|\mathbf{x}_B)$ with $\mathbf{x}_A \in \mathbb{F}_{\text{SD},\omega}^{m-k}$ and sets $\mathsf{sk} = \mathbf{x}_A$. Then it computes $\mathbf{y} = (\mathbf{H}'|\mathbf{I}_{m-k})\mathbf{x}$ and sets $\mathsf{pk} = (\mathbf{H}', \mathbf{y})$.

Algorithm 1 3-round Hypercube-SDitH – Part 1: $\mathsf{P.Commit}$

```
    Algorithm P.Commit:
Input: Secret key $\mathsf{sk} = \mathbf{x}_A \in \mathbb{F}_{\mathrm{SD}}^{m-k}$.
Output: Commitment message $\mathsf{w}$ (For simplicity we keep $\mathsf{st}$ implicit).
    Set-up:
 1: Choose $E \subset [m]$ such that $|E| = w$ and the non-zero coordinates of $x$ are in $E$.
 2: Compute $Q(X) = \prod_{i \in E}(X - \gamma_i) \in \mathbb{F}_{\mathrm{poly}}(X)$.
 3: Compute $S(X) \in \mathbb{F}_{\mathrm{poly}}(X)$ by interpolation over the coordinates of $\mathbf{x}$.
 4: Compute $P(X) = S(X) \cdot Q(X)/F(X)$ with $F(X) \in \mathbb{F}_{\mathrm{poly}}(X)$ s.t. $F(X) = \prod_{i=1}^{m}(X - \gamma_i)$.
 5: Sample a root seed: $\mathsf{seed} \leftarrow \{0,1\}^{\lambda}$.
 6: Expand root seed $\mathsf{seed}$ recursively using TreePRG to obtain $N^D$ leafs $\mathsf{seed}'_i$ which are further expanded to
    $(\mathsf{seed}_i, \rho_i) \leftarrow \mathsf{PRG}(\mathsf{seed}'_i), 0 \leq i < N^D$
 7: The index of a main party is $(k, j) \in \{0, \ldots, D-1\} \times \{0, \ldots, N-1\}$ and contains all leaf parties $i$ whose $k$-th
    coordinate is $j$ when $i$ is represented as radix $N$ integer.
 8: for each party $(k, j) \in \{0, \ldots, D-1\} \times \{0, \ldots, N-1\}$ do
 9:     Set $[\mathbf{x}_A]_{(k,j)}$, $[Q]_{(k,j)}$, $[P]_{(k,j)}$, $[a]_{(k,j)}$, $[b]_{(k,j)}$, and $[c]_{(k,j)}$ to zero.
    Expand leaf party seeds and commit:
10: for each leaf $i \in \{0, \ldots, N^D - 1\}$ do
11:     if $i \neq N^D - 1$ then
12:         $([\![a]\!]_i, [\![b]\!]_i, [\![c]\!]_i, [\![\mathbf{x}_A]\!]_i, [\![Q]\!]_i, [\![P]\!]_i) \leftarrow \mathrm{PRG}(\mathsf{seed}_i)$
13:         $\mathrm{state}_i = \mathsf{seed}_i$
14:     else
15:         $[\![a]\!]_{N^D-1}, [\![b]\!]_{N^D-1} \leftarrow \mathrm{PRG}(\mathsf{seed}_{N^D-1})$,    $[\![c]\!]_{N^D-1} = \langle a, b \rangle - \sum_{i \neq N^D-1} [\![c]\!]_i$
16:         $[\![\mathbf{x}_A]\!]_{N^D-1} = \mathbf{x}_A - \sum_{i \neq N^D-1} [\![\mathbf{x}_A]\!]_i$
17:         $[\![Q]\!]_{N^D-1} = Q - \sum_{i \neq N^D-1} [\![Q]\!]_i$,    $[\![P]\!]_{N^D-1} = P - \sum_{i \neq N^D-1} [\![P]\!]_i$,
18:         $aux = ([\![\mathbf{x}_A]\!]_{N^D-1}, [\![Q]\!]_{N^D-1}, [\![P]\!]_{N^D-1}, [\![c]\!]_{N^D-1})$, and $\mathrm{state}_{N^D-1} = \mathsf{seed}_{N^D-1} || aux$
19:     Leaf parties commit to their state $\mathsf{com}_i = \mathsf{Com}(state_i, \rho_i)$.
20: Compute $\mathsf{w}_1 = \mathsf{Hash}(\mathsf{com}_0, \ldots, \mathsf{com}_{N^D-1})$.
    Derive evaluation points and masks:
21: $\mathsf{P}$ derives $t$ challenge points $r \in \mathbb{F}_{\mathrm{points}}$ and masks $\epsilon \in \mathbb{F}_{\mathrm{points}}$ from commitment hash: $\{r, \varepsilon\}_0^{t-1} = \mathsf{PRG}(\mathsf{w}_1)$.
    Build main parties:
22: for Dimension $k \in \{0, \ldots, D-1\}$ do
23:     for Main party $j \in \{0, \ldots, N-1\}$ do
24:         Let $(i_1, \ldots, i_D)$ be the radix $N$ representation of $i$.
25:         Let $\mathcal{S}$ be the set of leaf parties with $i_k = j$.
26:         $[\mathbf{x}_A]_{(k,j)} = \sum_{i \in \mathcal{S}} [\![\mathbf{x}_A]\!]_i$,   $[Q]_{(k,j)} = \sum_{i \in \mathcal{S}} [\![Q]\!]_i$,   $[P]_{(k,j)} = \sum_{i \in \mathcal{S}} [\![P]\!]_i$
27:         $[a]_{(k,j)} = \sum_{i \in \mathcal{S}} [\![a]\!]_i$,   $[b]_{(k,j)} = \sum_{i \in \mathcal{S}} [\![b]\!]_i$,   $[c]_{(k,j)} = \sum_{i \in \mathcal{S}} [\![c]\!]_i$
    Execute MPC protocol:
28: for Dimension $k \in \{0, \ldots, D-1\}$ do
29:     Execute MPC protocol (Algorithm 6) between the main parties $(k, 1), \ldots, (k, N)$ to
        compute communication and result shares $\{[\alpha]_{(k,j)}, [\beta]_{(k,j)}, [v]_{(k,j)}\}_{j=0}^{N-1}$.
30: Set $\mathsf{w}_2 = \left\{ \left\{ [\alpha]_{(k,j)}, [\beta]_{(k,j)}, [v]_{(k,j)} \right\}_{j=0}^{N-1} \right\}_{k=0}^{D-1}$, and send $\mathsf{w} = (\mathsf{w}_1, \mathsf{w}_2)$ to $\mathsf{V}$.
```

Algorithm 2 3-round Hypercube-SDitH – Part 2: $\mathsf{V.Challenge}$, $\mathsf{P.Resp}$, $\mathsf{V.Vrf}$

```
    Algorithm V.Challenge:
Input: Commitment message $\mathsf{w}$.
Output: Challenge $\mathsf{c}$.
 1: $\mathsf{V}$ samples $\mathsf{c} \xleftarrow{\$} \{0, \dots, N^D - 1\}$ and sends it to $\mathsf{P}$.
    Algorithm P.Resp:
Input: Commitment message $\mathsf{w}$, challenge $\mathsf{c}$ (and internal state $\mathsf{st}$ that we left implicit).
Output: Response $\mathsf{z}$.
 2: Run the local computations of Algorithm 6 using the shares of leaf party $\mathsf{c}$ to obtain its contribution to the
    overall communication $([\![\alpha]\!]_c, [\![\beta]\!]_c)$.
 3: $\mathsf{P}$ sets $\mathsf{z} = (\mathsf{TP.extract}(\mathsf{seed}, \mathsf{c}), \mathsf{com_c}, ([\![\alpha]\!]_c, [\![\beta]\!]_c)$, adds $aux$ if $\mathsf{c} \neq N^D - 1$ and sends it to $\mathsf{V}$.
    Algorithm V.Vrf:
Input: Public key $\mathsf{pk} = (\mathbf{H}', \mathbf{y}) \in \mathbb{F}_{\mathrm{SD}}^{(m-k)\times k} \times \mathbb{F}_{\mathrm{SD}}^{(m-k)}$, commitment message $\mathsf{w}$, challenge $\mathsf{c}$ and response $\mathsf{z}$.
Output: Decision (ACCEPT/REJECT).
 4: for $i \in (\{0, \dots, N^D - 1\} \setminus \mathsf{c})$ do
 5:     Compute $(state_i, \rho_i)$ from $\mathsf{z}$ using $\mathsf{TreePRG}$.
 6:     Compute $\mathsf{com}'_i = \mathsf{Com}(state_i, \rho_i)$.
 7:     Compute $\mathsf{w}'_1 = \mathsf{Hash}(\mathsf{com}'_0, \dots, \mathsf{com_c}, \dots \mathsf{com}'_{N^D-1})$.
 8: for $(k \in \{0, \dots, D - 1\})$ do
 9:     Run Alg. 7 on inputs derived from $\{state_i\}_{i \neq \mathsf{c}, i=0}^{N^D-1}$, $([\![\alpha]\!]_c, [\![\beta]\!]_c)$,
         and $\{r, \varepsilon\}_0^{t-1} = \mathsf{PRG}(\mathsf{w}_1)$ to get $\left\{[\alpha']_{(k,j)}, [\beta']_{(k,j)}, [v']_{(k,j)}\right\}_{j=0}^{N-1}$.
10: if $(\mathsf{w}'_1, \mathsf{w}'_2) \neq \mathsf{w}$ where $\mathsf{w}'_2 = \left\{\left\{[\alpha']_{(k,j)}, [\beta']_{(k,j)}, [v']_{(k,j)}\right\}_{j=0}^{N-1}\right\}_{k=0}^{D-1}$ then return REJECT.
11: return ACCEPT.
```

Generating leaf parties. The prover P first generates the polynomials S, Q, P as explained above. Then it creates a secret sharing for each of them as follows. P first picks a fresh random seed and generates shares for all N^D leaf parties using $\mathsf{TreePRG}$ to generate $(state_i, \rho_i)$ which are the leaf's seed, and its commitment randomness. From $state_i$, the prover then derives the i^{th} share of each of the polynomials $[\![S]\!]_i, [\![P]\!]_i, [\![Q]\!]_i$, as well as its share of the Beaver triple $[\![a]\!]_i, [\![b]\!]_i, [\![c]\!]_i$ using PRG to expand $state_i$. For the auxiliary party $(i = N^D - 1)$, the secret share is then computed such that the shares sum up to the right values. This share is then appended to the auxiliary party's $state_i$. Then P commits to each state: $\mathsf{com}_i = \mathsf{Com}(state_i, \rho_i)$.

Building main parties. Next the prover builds the main parties for the MPC computations. The prover runs D MPC computations. For this, [2] aggregates the secret shares of the N^D leaf parties into an N party protocol in D different ways. This is done using D different partitions of the N^D leaf parties.

The partitions are computed as follows. First the index i of a leave party is turned into a vector of D values, the *hypercube representation*, taking its radix N representation $i = (i_0, \dots, i_{D-1})$. The leaf parties that are summed up to form the share of the j-th main party of the k-th MPC instance are those parties for which $i_k = j$. For $K = 1$, i.e., considering the first hypercube index, one obtains an N party MPCitH protocol, where the first party is the aggregation of all leave parties of the form $(0, i_1, \dots, i_{D-1})$, the second contains leaves of the form $(1, i_1, \dots, i_{D-1})$, and so on. This process is repeated for each of the D dimensions of the hypercube, giving D independent N-party MPCitH protocols.

Evaluation points. The next step is generating the points for validating the polynomial relation $S \cdot Q = P \cdot F$, the objective of which is for the prover to demonstrate that the preimage $\mathbf{x}$ they know for the syndrome decoding problem $\mathbf{y} = \mathbf{Hx}$ has *low weight*, i.e. $wt(\mathbf{x}) \leq \omega$. To that end, points $r_j \in \mathbb{F}_{\text{points}}$ and masks ϵ_j for $j = 0, \ldots, t-1$ are sampled. Then for each j, $S(r_j), Q(r_j), P(r_j), F(r_j)$ are computed via MPC and the identity $S(r_j) \cdot Q(r_j) = (P \cdot F)(r_j)$ is checked probabilistically via an MPC protocol using the mask ϵ_j.

In the five-round IDS of [2] and [14], the evaluation points and masks are selected at random by the verifier as the first challenge. However in the three round scheme we present here, the evaluation points are derived from the transcript of the previous steps that have occurred up to that point, i.e. they are generated by expanding the hash of the commitments w_1 using PRG.

MPC operations. At this stage P has all the information required to perform the MPC operations - the inputs being the shared main party polynomials evaluated at the challenge points to give $[s], [q], [pf]$. Beaver multiplication is then performed to verify the triple s, q, pf by sacrificing the Beaver triple a, b, c, as defined in Figure 3. This creates the communication shares $[\alpha], [\beta], [v]$. In each of the D dimensions $k \in D$ the prover runs Γ on each set of main party inputs, resulting in communication and output shares $\{[\alpha]_{(k,j)}, [\beta]_{(k,j)}, [v]_{(k,j)}\}_{j=0}^{N-1}$. This is repeated for each of the D dimensions of the hypercube, and all communications. Note that for honest P, $v_k = 0$ for all $k \in \{0, \ldots, D-1\}$.

Challenge. P sends the commitment hash w_1 together with all the communication and main party sharings $\mathsf{w}_2 = \left\{ \left\{ [\alpha]_{(k,j)}, [\beta]_{(k,j)}, [v]_{(k,j)} \right\}_{j=0}^{N-1} \right\}_{k=0}^{D-1}$ to V. The MPCitH challenge is then randomly sampled by the verifier and returned to the prover. This challenge is interpreted as an index c of one of the N^D leaf parties that does not need to be opened.

Response. P opens the views of all leaf parties except for c, by sending $(state_i, \rho_i)$. This is done more efficiently using $\mathsf{TP.extract}(\mathsf{seed}, \mathsf{c})$ to extract the sibling path path for leaf c from TreePRG and sending this instead. The prover also sends the initial commitment com_c and communications $(\llbracket\alpha\rrbracket_\mathsf{c}, \llbracket\beta\rrbracket_\mathsf{c})$ for the hidden (leaf) party, and aux in case that $\mathsf{c} \neq N^D - 1$. Note that the communication shares would not have to be sent for the IDS as they are already part of w. However, we send them as we want Hypercube-SDitH to be commitment-recoverable.

Verification. The verifier first recomputes the commitment hash w_1 by computing commitments $\{\mathsf{com}'_i\}_{i \neq \mathsf{c}}$ for each of the states of the $N^D - 1$ leaf parties that have been revealed, and then combining with com_c to compute w'_1.

Next the verifier expands the commitment hash to get the evaluation points and masks, and compiles the polynomial shares for each of the main parties from the given state information. Once this is done, they execute the MPCitH protocol on main parties as in the original SDitH proposal [14] for each of the D dimensions k using $\llbracket\alpha\rrbracket_\mathsf{c}, \llbracket\beta\rrbracket_\mathsf{c}$. Here we exploit that by linearity of the calculations of $[\alpha]$ and $[\beta]$ the communications of the main party k, i'_k that contains the unopened leaf party c can be computed by assembling the respective leaf party shares, and that $v = 0$ when determine $[v]_{k,\mathsf{c}_k}$. The final main party communications and output shares $\{[\alpha]_{k,j}, [\beta]_{k,j}, [v]_{k,j}\}_{j=0}^{N-1}$ for each dimension $k \in \{0, \ldots, D-1\}$ are then assembled to obtain w'_2. This part also represents the commitment recovery algorithm Rcvr for Hypercube-SDitH. The final output is the result of the comparison $(\mathsf{w}'_1, \mathsf{w}'_2) \stackrel{?}{=} \mathsf{w}$.

Parallel composition (Π). In the above, we did describe the routines performed in the atomic three round IDS ($\tau = 1$), which takes soundness error $\simeq 1/N^D$. In order to reach negligible soundness error of 2^{-n} one can repeat the IDS many times independently in parallel such that $(1/N^D)^\tau \leq 2^{-n}$.

However, we note here that since the evaluation points are generated offline by the prover, it is possible to make the polynomial test harder to cheat by deriving the challenge points from a hash of the commitments com from all τ parallel repetitions. Denote the τ-fold parallel IDS as Π. Then to generate the challenge points/masking point pairs $\left\{\{r_i^j, \varepsilon_i^j\}_{i=0}^{t-1}\right\}_{j=0}^{\tau-1}$ we take $\mathsf{w}_1 = \mathsf{PRG} \circ \mathsf{Hash}(\mathsf{com}_1, \ldots, \mathsf{com}_{\tau N^D-1})$, therefore the evaluation points for all τ repetitions depend on the state commitments of all leaves in the entire τ-fold protocol Π.

3 Security of the 3-Round IDS

In this section we discuss the security of our IDS. We prove that the IDS is multi-transcript honest-verifier zero-knowledge (HVZK) and has special-soundness. We begin with HVZK proving the following theorem:

Theorem 1 (Honest-Verifier Zero Knowledge (HVZK)). *The algorithm* Sim_Π *shown in Algorithm 3 is an HVZK simulator for* Π *such that for any quantum algorithm* A *in distinguishing* Trans_Π *from* Sim_Π *making at most* q_{zk} *queries to its oracle there exist algorithms* B– *distinguishing the outputs of* $\mathsf{TreePRG}$ *from random – and* C– *breaking the hiding property of* Com– *which fulfill*

$$\begin{aligned}\mathrm{Adv}^{\mathsf{hvzk}}_{\Pi,\mathsf{Sim}}(\mathsf{A}) &:= \left|\Pr[1 \leftarrow \mathsf{A}^{\mathsf{Sim}_\Pi}] - \Pr[1 \leftarrow \mathsf{A}^{\mathsf{Trans}_\Pi}]\right| \\ &\leq q_{\mathsf{zk}}\tau(\mathrm{Adv}^{\mathsf{hide}}_{\mathsf{Com}}(\mathsf{C}) + \mathrm{Adv}^{\mathsf{ror}}_{\mathsf{TreePRG}}(\mathsf{B})),\end{aligned}$$

where B *and* C *run in time* $\mathsf{TIME}(\mathsf{B}) = \mathsf{TIME}(\mathsf{C}) = \mathsf{TIME}(\mathsf{A}) + \mathsf{TIME}(\mathsf{Trans})$ *respectively.*

On a high level, our proof follows a sequence of game hops, where we slowly change the oracle given to the adversary. We start with Trans, i.e., the honest execution of the protocol, in GAME_0. In the first hop, we switch the order of operations and sample the challenges first. This defines GAME_1. In GAME_2, we replace the seed seed_c and the commitment pseudorandomness ρ_c for the commitments that remain unopened by truly random bits. To be consistent with $\mathsf{TreePRG}$, we also sample a random sibling path path which we use to derive the values for the opened commitments. This whole change is only detectable up to a τ-fold distinguishing advantage against $\mathsf{TreePRG}$ per oracle query. Next, we replace the state of the unopened parties by truly random bits in GAME_3. This is undetectable up to a τ-fold advantage against the hiding property of the commitment scheme per oracle query. At this point, the distribution of the auxiliary state (of party $N^D - 1$) is independent of the sum of the other shares. Hence, in GAME_4 we sample that state uniformly at random. To preserve consistency of the communications, we compute the communications of all opened parties using the original algorithm. Then we compute the communication of the unopened parties to agree with these. At this point we don't need the secret key anymore and observe that the oracle in GAME_4 corresponds to Sim. The full proof can be found in Supp.Mat. D

We now move on to prove soundness of Π. Maybe not surprisingly, this is based on the binding property of the used commitment and the soundness of the non-interactive polynomial test which we prove first.

Soundness of the non-interactive polynomial test.

Here, we prove a query lower bound on the oracle search problem of finding inputs x, P and Q that "cheat" on the polynomial test implemented as MPC computation in Commit of Π (c.f., Algorithm 1). More generally, we will show concrete query lower bounds for the *family* of search

Algorithm 3 HVZK simulator Sim_{Π} (Simplified version $[\tau = 1]$)

```
    Step 1: Sample challenge.
 1: $\mathsf{c} \leftarrow \{0, \dots, N^D - 1\}$.
    Step 2: generate $N^D$ leaf party states and witness shares.
 2: Sample sibling path $\mathsf{path} \leftarrow\$ \{0,1\}^{n \times \log_2 N^D}$ for leaf $\mathsf{c}$ and $\{(\mathsf{seed}_\mathsf{c}, \rho_\mathsf{c})\} \leftarrow\$ \{0,1\}^n$.
 3: for $i' \neq$ do
 4:     Generate $\{(\mathsf{seed}_i, \rho_i)\}$ via $\mathsf{TreePRG}(\mathsf{path})$ and $\mathsf{PRG}$.
 5:     if $i' \neq N^D - 1$ then
 6:         Set $\mathsf{state}_i = \mathsf{seed}_i$.
 7:         Expand $\mathsf{seed}_i$ into witness shares.
 8:     else
 9:         To generate $aux$ for the last leaf party, $i' = N^D - 1$, randomly draw
            $[\![\mathbf{x}_A]\!]_{N^D-1}$, $[\![Q]\!]_{N^D-1}$, $[\![P]\!]_{N^D-1}$, and $[\![c]\!]_{N^D-1}$.
10:         Set $\mathsf{state}_{N^D-1} = (\mathsf{seed}_{N^D-1} \| aux)$.
    Step 3: generate leaf party commitments
11: for $i' \neq \mathsf{c}$ do Compute $\mathsf{com}_{i'} = \mathsf{Hash}(state_{i'}, \rho_{i'})$
12: Draw $\mathsf{com}_\mathsf{c}$ at random.
13: Compute commitment hash $\mathsf{w}_1 = \mathsf{Hash}(\mathsf{com}_0, \dots, \mathsf{com}_{i*}, \dots, \mathsf{com}_{N^D-1})$.
    Step 4: compute evaluation points
14: $\{r_l, \epsilon_l\}_{l=1}^t \leftarrow \mathsf{PRG}(\mathsf{w}_1)$
    Step 5: generate party communications
15: Draw $[\![\alpha]\!]_\mathsf{c}$ and $[\![\beta]\!]_\mathsf{c}$ uniformly at random from their respective domains.
16: for $k \in \{0, \dots, D-1\}$ do
17:     Let the main party to which $\mathsf{c}$ belongs be $(k, j^*)$
18:     for $(k, j) \neq (k, j^*)$ do
19:         Compute communications $[\alpha]_{k,j^*}, [\beta]_{k,j^*}, [v]_{k,j^*}$ following Algorithms 1 and 6
20:     for $(k, j^*)$ do
21:         Compute party communication shares $[\alpha]_{k,j^*}, [\beta]_{k,j^*}, [v]_{k,j^*}$ by running $\Pi$ on the sum of the witnesses of the $N^{D-1} - 1$ *revealed* leaf parties in main party $(k, j^*)$, as described in Algorithm 1, then add on $[\![\alpha]\!]_\mathsf{c}$ and $[\![\beta]\!]_\mathsf{c}$
22:         Set $v_\mathsf{c} = -\sum_{i' \neq \mathsf{c}} [\![v]\!]$.
    Step 6: Output transcript $((\mathsf{w}_1, \mathsf{w}_2), \mathsf{c}, \mathsf{z})$:
23: $\mathsf{w}_2 = \{\{[\alpha]_{(k,j)}, [\beta]_{(k,j)}, [v]_{(k,j)}\}_{k=0}^{D-1}\}_{j=0}^{N-1}$, $\mathsf{c} = \mathsf{c}$
24: $\mathsf{z} = \mathsf{com}_\mathsf{c}, \{(\mathrm{state}_\mathsf{c}, \rho_\mathsf{c}) \, \forall \, i \neq \mathsf{c}\}$.
```

problems where the goal is to cheat ℓ out of τ parallel repetitions of the polynomial test, where the challenge points for all repetitions are generated by hashing all commitments together.

We begin by finding a more abstract formulation that is a common generalization of all the mentioned problems of cheating (some of) the polynomial zero tests. To that end, let $\mathcal{P}(P_1, ..., P_{n_p})$ be a predicate on polynomials $P_i \in \mathfrak{P}_i \subset \mathbb{F}_{\text{poly}}[X], i = 1, ..., n_p$. The domains $\mathfrak{P}_i$ can be different for every polynomial and can, e.g., reflect degree limitations (e.g. for polynomials P and Q in Π) or that a polynomial has been obtained via interpolation (e.g. for polynomial S in Π). Let $\mathbf{T} = (T_1, \dots, T_{n_t})$ be a list of test polynomials $T_i \in R[X_1, ..., X_{n_p}], i = 1, ..., n_t$ for $R = \mathbb{F}_{\text{poly}}[X]$ such that $\mathcal{P}(\mathbf{P}) = 0 \implies T_i(\mathbf{P}) = 0$ for all i, where $\mathbf{P} = (P_1, ..., P_{n_p}) \in \mathfrak{P} = \mathfrak{P}_1 \times \mathfrak{P}_2 \times \ldots \times \mathfrak{P}_{n_p}$.[5] In addition, let $\mathcal{M}$ be a randomized algorithm that takes as input a testing polynomial T, a tuple of polynomials $\mathbf{P}$, an evaluation point r and a random masking point $\epsilon \in \mathfrak{E}$, with the purpose that if $T(\mathbf{P})(r) \neq 0$ then the probability that $\mathcal{M}$ outputs 0 is small. We define the false-positive probability

$$p^{\text{fp}}_{\mathbf{T},\ell} = \max_{\mathbf{P}:\mathcal{P}(\mathbf{P})=1} \left(\Pr_{\substack{\mathbf{r} \leftarrow \mathbb{F}^{n_t}_{\text{points}} \\ \boldsymbol{\epsilon} \leftarrow \mathfrak{E}^{n_t}}} \left[|\{i \in [n_t] | \mathcal{M}(T_i, \mathbf{P}, r; \epsilon_i) = 0\}| \geq \ell \right] \right), \tag{1}$$

[5] In our application, we only need T_i with coefficients in $\mathbb{F}_{\text{poly}} \subset \mathbb{F}_{\text{poly}}[X]$.

Algorithm 4 Abstract non-interactive polynomial zero test for secret-shared polynomials $\mathcal{T}$

Input: Secret-shared polynomials $(state_i, \rho_i)_{i=1}^{n_c}$, threshold ℓ
Output: Boolean value $b \in \{0, 1\}$.

```
$state = (state_i)_{i=1}^{n_c}$
$\mathbf{P} = \mathcal{R}(state)$
$\mathsf{com}_i = \mathsf{Com}(state_i, \rho_i)$ for all $i = 1, \ldots, n_c$
$\big((r_{i,j})_{(i,j)\in[n_t]\times[t]}, (\epsilon_{i,j})_{(i,j)\in[n_t]\times[t]}\big) = G((\mathsf{com}_i)_{i=1}^{n_c})$

Perform zero checks:
count= 0
for $i \in [n_t]$ do
    for $j \in [t]$ do
        if $\mathcal{M}(T_i, \mathbf{P}, r_{i,j}; \epsilon_{i,j}) = 0$ then
            count=count+1
$b = 1$
if count$\geq \ell$ then $b = 0$
return $b$
```

where the maximum is over $\mathbf{P} = (P_1, ..., P_{n_p})$ such that $P_i \in \mathfrak{P}_i$. In words, this is the maximum probability for any set of polynomials that doesn't fulfil the predicate to pass a test where each testing polynomials T_i is evaluated at a random point using $\mathcal{M}$, and at least ℓ of the results are 0.

In Round 1 of Π, $\mathbf{P}$ is secret shared. The secret shares of $\mathbf{P}$ are generated using a two stage PRG structure, first using $\mathsf{TreePRG}$ to generate seeds $seed_i$ from a single root seed $seed$ followed by PRG to expand the $seed_i$ into secret shares, commitment randomness, and other objects irrelevant here). The evaluation points and masks for the polynomial test are then derived from the individual commitments to all $seed_i$ by hashing all these commitments together. This complicates the analysis because this three-step process is not exactly indistinguishable from a random oracle. We will not need the pseudorandomness properties to give a query bound for our search problems. For this section it is sufficient to define two black box algorithms $\mathcal{S}$, and $\mathcal{R}$ which abstract away the generation of the secret shared values and their recombination as follows:

$$\mathcal{S}(\mathbf{P}; seed) = (state_i, \rho_i)_{i=1}^{n_c} \text{ and } \mathcal{R}(state) = \mathbf{P},$$

where we set $state = (state_i)_{i=1}^{n_c}$. Let $\mathfrak{S}$ be such that $(state_i, \rho_i) \in \mathfrak{S}$ for all i, $\mathbf{T}$ a tuple of testing polynomials for a predicate $\mathcal{P}$, t a non-negative integer and $\mathsf{Com} : \mathfrak{S} \to \mathfrak{C}$ and $G : \mathfrak{C}^{n_c} \to \mathbb{F}_{\text{points}}^{2t \cdot n_t}$ two hash functions modeling the commitment scheme and the use of PRG for computing the challenge points and masks. We define the abstract non-interactive polynomial zero test algorithm $\mathcal{T}$ in Algorithm 4.

We now model the hash functions Com and G as random oracles. The most natural oracle search problem associated with the task of cheating the polynomial test in Algorithm 4 would be to find inputs $\mathbf{P}$ and $seed$ such that $\mathcal{P}(\mathbf{P}) = 1$, i.e., the predicate is not satisfied, yet $\mathbf{P}$ evaluates to zero at the challenge points, i.e. running algorithm $\mathcal{S}$ followed by $\mathcal{T}$ (Algorithm 4) returns 0. Unfortunately, a special soundness extractor for Π cannot solve this problem, as the root seed is never revealed. A search problem that can be solved using a special soundness extractor for the protocol in Algorithms 1 and 2 is to find $(state_i, \rho_i)_{i=1}^{n_c}$ such that for $\mathcal{R}(state) = \mathbf{P}$ we have $\mathcal{P}(\mathbf{P}) = 1$ but executing Algorithm 4 directly results in output $b = 0$. We are now ready to define our search problem.

Definition 5 (Non-interactive polynomial zero test cheating problem). *Let $\mathcal{P}, \mathbf{T}, t$ be as above. An oracle algorithm $\mathsf{A}^{\mathsf{Com},G}$ with access to random oracles* Com *and G as above solves the* *non-interactive polynomial zero test cheating problem* $\mathsf{Cheat}_{\mathcal{P},\mathbf{T},t,\ell}$ *if it outputs* $o = (state_i, \rho_i)_{i=1}^{n_c}$ *such that* $\mathcal{P}(\mathcal{R}(o)) = 1$ *but* $\mathcal{T}_{\mathbf{T},t}(o,\ell) = 0$.

We first give a query bound for the case where A has classical oracle access only. In the following, for $O \in \{\mathsf{Com}, G\}$, let D_O be the list of pairs $(x, O(x))$ for queries x made by A to its oracle O. We overload the list symbols by writing

$$D_O(x) = \begin{cases} y & (x,y) \in D_O \\ \perp & \text{else.} \end{cases}$$

Remark 1. We can now regard Com and G as domain-separated parts of the same random oracle F. We assume that F has a sufficiently large output space $\mathfrak{F}$ and introduce truncation functions trunc_O such that $\mathsf{Com} = \mathrm{trunc}_{\mathsf{Com}} \circ F|_{\mathfrak{S}}$ and $G = \mathrm{trunc}_G \circ F|_{\mathfrak{C}^{n_c}}$. We let D be the query list for F. The lists D_O for $O \in \{\mathsf{Com}, G\}$ are obtained as sublists of D with trunc_O applied to all outputs. In the following, we try to keep the notation lean by omitting the truncation functions.

Following [9] we call a predicate on query lists a *database property*. For database properties P and Q, the classical transition capacity is defined as

$$[P \to Q] = \max_{\substack{L:P(D)\\ s\in\mathfrak{S}\cup\mathfrak{C}^{n_c}}} \Pr_{u\leftarrow\mathfrak{F}}[Q(D \cup (s,u))].$$

Here, $D \cup (s,u)$ denotes the query list D with the pair (s,u) added if D did not contain a pair (s,y) yet. The proof strategy for the following theorems bears some similarity to the proof of Lemma 4.1 in [11].

Theorem 2. *Let* $\mathsf{A}^{\mathsf{Com},G}$ *be an algorithm that makes* q_{Com}*, and* q_G *classical queries to its oracles* Com*, and* G*, respectively and let* $q = q_{\mathsf{Com}} + q_G$*. Then*

$$\Pr_{o\leftarrow\mathsf{A}^{\mathsf{Com},G}}[(\mathcal{P}(\mathcal{R}(o))=1) \wedge (\mathcal{T}(o,\ell)=0)] \le (q+n_c+1)\max\left(p^{\mathrm{fp}}_{\mathbf{T},\ell}, n_c\frac{q_G}{|\mathfrak{C}|}\right).$$

Proof. We denote by $\mathcal{T}_D$ the variant of the zero test $\mathcal{T}$ where for $O \in \{\mathsf{Com}, G\}$, any call to the oracle O is replaced by a call to D_O. If any such call outputs $\perp$, $\mathcal{T}_D$ outputs $\perp$. Let $\mathsf{A}^{\mathsf{Com},G}$ be an algorithm as in the theorem statement. We define ${\mathsf{A}'}^{\mathsf{Com},G}$ as follows. ${\mathsf{A}'}^{\mathsf{Com},G}$ computes $o \leftarrow \mathsf{A}^{\mathsf{Com},G}$, makes queries $\mathsf{com}_i = \mathsf{Com}(o_i)$ and $\mathbf{r} = G(\mathsf{com}_1, \ldots, \mathsf{com}_{n_c})$, and outputs o. Now we have

$$\begin{aligned}
&\Pr_{o\leftarrow\mathsf{A}^{\mathsf{Com},G}}[(\mathcal{P}(\mathcal{R}(o))=1) \wedge (\mathcal{T}(o,\ell)=0)] \\
&\qquad= \Pr_{o\leftarrow{\mathsf{A}'}^{\mathsf{Com},G}}[(\mathcal{P}(\mathcal{R}(o))=1) \wedge (\mathcal{T}_D(o,\ell)=0)] \\
&\qquad\le \Pr_{o\leftarrow{\mathsf{A}'}^{\mathsf{Com},G}}[\exists o' : (\mathcal{P}(\mathcal{R}(o'))=1) \wedge (\mathcal{T}_D(o',\ell)=0)]. \qquad (2)
\end{aligned}$$

On an intuitive level the above inequality reflect the fact that the adversary A' is guaranteed to perform the test $\mathcal{T}$ on its own output, so if the test checks out, a combination of input-output pairs which certifies the existence of a successful output can be found in the query list. The event in the last probability expression defines the database property

$$\mathsf{Found}(D) = (\exists o' : \mathcal{P}(\mathcal{R}(o')) = 1 \wedge \mathcal{T}_D(o') = 0).$$

Now let D_i be the list of queries after A' has been run until its ith query (of any kind). Clearly, $\mathsf{Found}(D_i) \implies \mathsf{Found}(D)$. We thus get

$$\begin{aligned}\Pr[\mathsf{Found}(D)] &= \Pr[\mathsf{Found}(D_{\tilde{q}})] \\ &= \sum_{k=1}^{\tilde{q}} \Pr[\mathsf{Found}(D_k) \wedge \neg\mathsf{Found}(D_{k-1})] \\ &\leq \sum_{k=1}^{\tilde{q}} [\neg\mathsf{Found} \wedge (|D| \leq k-1) \to \mathsf{Found}], \end{aligned} \tag{3}$$

where the right hand side represents the sum of transition probabilities, and

$$\tilde{q} = q + n_c + 1. \tag{4}$$

It remains to bound the transition capacities in the sum. For this we now make a case distinction. Setting $P = \neg\mathsf{Found} \wedge (|D| \leq k-1)$, we have

$$\begin{aligned}[P \to \mathsf{Found}] = &\max_{\substack{D:P(D) \\ s\in\mathfrak{S}\cup\mathfrak{C}^{n_c}}} \Pr_{u\leftarrow\mathfrak{F}}[\mathsf{Found}(D \cup (s,u))] \\ = &\max\Big(\max_{\substack{D:P(D) \\ s\in\mathfrak{S}}} \Pr_{u\leftarrow\mathfrak{F}}[\mathsf{Found}(D \cup (s,u))], \max_{\substack{D:P(D) \\ s\in\mathfrak{C}^{n_c}}} \Pr_{u\leftarrow\mathfrak{F}}[\mathsf{Found}(D \cup (s,u))]\Big)\end{aligned}$$

If $\neg\mathsf{Found}(D)$ and $\mathsf{Found}(D \cup (s,u))$ for some $s \in \mathfrak{S}$, then there exist $i \in n_c$ and $(o_{i'}, \mathsf{com}_{i'}) \in D_{\mathsf{Com}}$ for $i' \neq i$ such that $(\mathsf{com}_1, \ldots, \mathsf{com}_{i-1}, u, \mathsf{com}_{i+1}, \ldots, \mathsf{com}_{n_c}) \in D_G$. Upper-bounding the first event by 1, we obtain the bound

$$\max_{\substack{D:P(D) \\ s\in\mathfrak{S}}} \Pr_{u\leftarrow\mathfrak{F}}[\mathsf{Found}(D \cup (s,u))] \leq n_c \frac{|D_G|}{|\mathfrak{C}|} \leq n_c \frac{q_G}{|\mathfrak{C}|}, \tag{5}$$

as for each entry of D_G there are n_c targets for the output of Com to match. If $\neg\mathsf{Found}(D)$ and $\mathsf{Found}(D \cup (s,u))$ for some $s \in \mathfrak{C}^{n_c}$, then $(o_i, \mathsf{com}_i) \in D_{\mathsf{Com}}$ for $i \in [n_c]$ such that $s = (\mathsf{com}_1, \ldots, \mathsf{com}_{n_c})$, and $\mathbf{T}(\mathcal{R}(o_1, \ldots, o_{n_c}))(G(x)) = 0$. Based on only the last condition, we get

$$\max_{\substack{D:P(D) \\ s\in\mathfrak{C}^{n_c}}} \Pr_{u\leftarrow\mathfrak{F}}[\mathsf{Found}(D \cup (s,u))] \leq p^{\mathrm{fp}}_{\mathbf{T},\ell} \tag{6}$$

Combining the last three equations we get

$$\Pr[\mathsf{Found}(L)] = \tilde{q} \max\left(p^{\mathrm{fp}}_{\mathbf{T},\ell}, n_c \frac{q_G}{|\mathfrak{C}|}\right). \tag{7}$$

Combining Equations (2), (3) and (7) yields the desired bound. □

We now move on to bound the success probability of an algorithm trying to solve $\mathsf{Cheat}_{\mathcal{P},\mathbf{T},t}$ given quantum access to the random oracle(s). This is necessary to later prove the security of our digital signature scheme in the quantum-accessible random oracle model (QROM).

In [9], a generic method for proving such bounds is introduced that essentially generalizes (a very general version of) the technique used in the proof of Theorem 2 to the QROM. Their technique uses Zhandry's compressed oracle method [34], but their results are sufficiently versatile to allow us to prove our desired bound without introducing compressed oracles.

In fact, we need to prove a bound in a slightly stronger model. As we will use the technique of [11] to construct a special soundness adversary from an adversary against the signature scheme, that special soundness adversary can only work in a model where the quantum-accessible random oracle is instantiated with the efficient oracle simulation via Zhandry's compressed oracle, and any adversary can proceed by i) making a number of queries to the oracle, ii) obtain the measurement outcome of measuring the internal state of the oracle simulation, and iii) computing the output. The measured internal state of the oracle essentially contains a query transcript of the adversarial algorithm. A classical adversary can just compile such a query transcript themselves, without relying on augmented access to the random oracle. In the quantum setting, the no-cloning principle prevents the adversary from recording a query transcript. This can be an issue if the adversary has a black-box subroutine that makes queries and relies knowledge of these queries. For solving an oracle search problem, however, the additional power of obtaining the measured query transcript does not help. It is important to notice that we use this model as a proof tool and don't have to ascribe it any predictive power for real-world hash functions. We call this model QROM+. In Appendix B, we prove the following lemma, specializing and slightly improving a combination of results from [9].

Lemma 1 (A compressed oracle query bound lemma). *Let $F : \mathcal{X} \to \mathcal{Y}$ be a random oracle and let $\mathcal{P}^F$ be a predicate on some set $\mathcal{Z}$ that can be computed using at most $q_{\mathcal{P}}$ classical queries to F. Let further A^F be a QROM+ algorithm making at most q quantum queries to F and outputing $z \in \mathcal{Z}$. Then*

$$\sqrt{\Pr_{z\leftarrow \mathsf{A}^F}[P(z)]} \leq \sum_{k=1}^{q+q_{\mathcal{P}}} \max_{\substack{x,D:\\ |D|\leq k\\ \neg\mathsf{Found}(D)}} \sqrt{10 \Pr_{u\leftarrow\mathcal{Y}}[\mathsf{Found}_{\mathcal{P}}(D[x\mapsto u])]} \tag{8}$$

where $\mathsf{Found}_{\mathcal{P}}$ *is the database property*

$$\mathsf{Found}_{\mathcal{P}} = (\exists z \in \mathcal{Z} : \mathcal{P}^D(z)) \tag{9}$$

and $\mathcal{P}^D$ is the algorithm that computes $\mathcal{P}$ but makes queries to D instead of F, and if any query returns $\perp$, $\mathcal{P}^D$ ouptuts 'false'.

We use this lemma to prove a quantum query complexity bound for $\mathsf{Cheat}_{\mathcal{P},\mathbf{T},t,\ell}$.

Theorem 3. *Let $\mathsf{A}^{\mathsf{Com},G}$ be a QROM+ algorithm that makes q_{Com} and q_G quantum queries to its oracles Com and G, respectively, and let $\tilde{q} = q_{\mathsf{Com}} + q_G + n_c + 1$. Then*

$$\Pr_{o\leftarrow \mathsf{A}^{\mathsf{Com},G}}[\mathcal{P}(\mathcal{R}(o)) = 1 \wedge \mathcal{T}(o,\ell) = 0] \leq 10 \cdot \begin{cases} \tilde{q}^2 p^{\mathrm{fp}}_{\mathbf{T},\ell} & \text{if } n_c\tilde{q} \leq p^{\mathrm{fp}}_{\mathbf{T},\ell}|\mathfrak{C}| \\ n_c \frac{\tilde{q}^3}{|\mathfrak{C}|} & \text{else.} \end{cases}$$

Proof. We again view the two random oracles as being constructed from a single one using domain separation and truncation, see Remark 1. Using the same reasoning as for Equations (5) and (6), but without taking a maximal size of the sub-database corresponding to G into account, we get for $|D| \leq k$ that

$$\Pr_{u\leftarrow\mathfrak{F}}[\mathsf{Found}_{\mathcal{P}}(D[x\mapsto u])] \leq \max\left(p^{\mathrm{fp}}_{\mathbf{T},\ell}, n_c\frac{k}{|\mathfrak{C}|}\right).$$

Suppose now first that $n_c\tilde{q} \leq p^{\mathrm{fp}}_{\mathbf{T},\ell}|\mathfrak{C}|$. Then we have

$$n_c\frac{k}{|\mathfrak{C}|} \leq n_c\frac{\tilde{q}}{|\mathfrak{C}|} \leq p^{\mathrm{fp}}_{\mathbf{T},\ell}$$

and thus

$$\max\left(p^{\mathrm{fp}}_{\mathbf{T},\ell}, n_c\frac{k}{|\mathfrak{C}|}\right) = p^{\mathrm{fp}}_{\mathbf{T},\ell}.$$

If on the other hand $n_c\tilde{q} > p^{\mathrm{fp}}_{\mathbf{T},\ell}|\mathfrak{C}|$, we have

$$\max\left(p^{\mathrm{fp}}_{\mathbf{T},\ell}, n_c\frac{k}{|\mathfrak{C}|}\right) \le \max\left(p^{\mathrm{fp}}_{\mathbf{T},\ell}, n_c\frac{\tilde{q}}{|\mathfrak{C}|}\right) = n_c\frac{\tilde{q}}{|\mathfrak{C}|}.$$

Note that the predicate checking whether A has solved $\mathsf{Cheat}_{\mathcal{P},\mathbf{T},t}$ makes n_c+1 queries. Setting

$$\eta = \begin{cases} p^{\mathrm{fp}}_{\mathbf{T},\ell} & \text{if } n_c\tilde{q} \le p^{\mathrm{fp}}_{\mathbf{T},\ell}|\mathfrak{C}| \\ n_c\frac{\tilde{q}}{|\mathfrak{C}|} & \text{else,} \end{cases}$$

we apply Lemma 1 to obtain

$$\sqrt{\Pr_{o\leftarrow\mathsf{A}^{\mathsf{Com},G}}[\mathcal{P}(\mathcal{R}(o)) = 1 \wedge \mathcal{T}(o,\ell) = 0]} \le \sum_{k=1}^{\tilde{q}} \max_{\substack{x,D:\\ |D|\le k\\ \neg\mathsf{Found}(D)}} \sqrt{10\Pr_{u\leftarrow\mathcal{Y}}[\mathsf{Found}_{\mathcal{P}}(D[x\mapsto u])]}$$

$$\le \sum_{k=1}^{\tilde{q}}\sqrt{10\eta} = \tilde{q}\sqrt{10\eta}.$$

Squaring both sides of the inequality yields the desired bound. □

We proceed to apply the above theorems to the particular polynomial zero test that appears in Hypercube-SDitH. In this test, there are τ parallel repetitions of the atomic test described in Section 2.3, and the evaluation points for all of them are generated by hashing all commitments together. Each test involves 3 polynomials in addition to the public polynomial F, , i.e. we have $n_p = 3\tau + 1$. Denoting the polynomials involved in the ith test by $S^{(i)}, P^{(i)}, Q^{(i)}$ and F, we set $P_i = S^{(i)}$, $P_{\tau+i} = P^{(i)}$ and $P_{2\tau+i} = Q^{(i)}$ for $i = 1,\dots,\tau$, and $P_{3\tau+1} = F$. We define the corresponding domains. For $i = 1,\dots,\tau$ we set

$$\begin{aligned}\mathfrak{P}_i &= \{S \in \mathbb{F}_{\mathrm{poly}}[X] | \deg(S) \le m\},\\ \mathfrak{P}_{\tau+i} &= \mathbb{F}_{\mathrm{poly}}[X]\\ \mathfrak{P}_{2\tau+i} &= \left\{Q \in \mathbb{F}_{\mathrm{poly}}[X] | Q(x) = x^\omega + Q'(x) \text{ with } \deg(Q') \le \omega - 1\right\}, \text{ and}\\ \mathfrak{P}_{3\tau+1} &= \{F\}.\end{aligned}$$

The predicate $\mathcal{P}$ is defined by

$$\mathcal{P}(\mathbf{P}) = \begin{cases} 0 & \text{if } \exists i \in [\tau] : P_iP_{2\tau+i} = P_{\tau+i}P_{3\tau+1} \\ 1 & \text{else} \end{cases} \tag{10}$$

and $T_i = P_iP_{2\tau+i} - P_{\tau+i}P_{3\tau+1}$. The intuition behind this predicate is, that any one out of the τ sets of four polynomials can be used to extract the secret key if it fulfils the polynomial identity. Define

$$p = \max_{\mathbf{P}:T_i(\mathbf{P})\ne 0}\ \Pr_{(r.\epsilon)\leftarrow\mathbb{F}^2_{\mathrm{points}}}[\mathcal{M}(T_i, \mathbf{P}, r; \epsilon) = 0]. \tag{11}$$

A bound for this probability can be obtained as follows. The polynomial $SQ - FP$ is non-zero and has degree at most $m + w - 1$. Setting $|\mathbb{F}_{\mathrm{points}}| = \Delta$, we get

$$\Pr_{r\leftarrow\mathbb{F}_{\mathrm{points}}}[(SQ - FP)(r) = 0] \le \frac{m + \omega - 1}{\Delta}.$$

In case $(SQ - FP)(r) \neq 0$, the probabilistic product verification fails with probability $\frac{1}{\Delta}$. We thus get

$$p \leq \frac{m+\omega-1}{\Delta} + \left(1 - \frac{m+\omega-1}{\Delta}\right)\frac{1}{\Delta} = \frac{m+\omega}{\Delta} - \frac{m+\omega-1}{\Delta^2}. \tag{12}$$

The t evaluation points and t masks are sampled independently, so the false positive probability for a single test with t points is just p^t. The probability that the t tests with random masks and evaluation points all fail for T_i, for all $i \in J \subset [\tau]$ with $|J| = \ell$ is just $p^{t\ell}$. Via a union bound, we obtain

$$p^{\mathrm{fp}}_{\mathbf{T},\ell} \leq \binom{\tau}{\ell} p^{t\ell}.$$

Combining the discussion above, we get the following

Corollary 1. *Let $\mathsf{A}^{\mathsf{Com},G}$ be an adversary that makes q_{Com}, and q_G queries to its oracles Com, and G, respectively, and let $\tilde{q} = q_{\mathsf{Com}} + q_G + n_c + 1$, where $n_c = \tau \cdot N^D$ is the number of commitments. The probability that its output wins $\mathsf{Cheat}_{\mathcal{P},\mathbf{T},t,\ell}$ in this case is bounded by*

$$\Pr_{o \leftarrow \mathsf{A}^{\mathsf{Com},G}}[(\mathcal{P}(\mathcal{R}(o)) = 1) \wedge (\mathcal{T}(o,\ell) = 0)] \leq \tilde{q} \max\left(\binom{\tau}{\ell} p^{t\ell}, \tau \cdot N^D \frac{q_G}{2^c}\right).$$

in the ROM. In the QROM+, the bound

$$\Pr_{o \leftarrow \mathsf{A}^{\mathsf{Com},G}}[\mathcal{P}(\mathcal{R}(o)) = 1 \wedge \mathcal{T}(o,\ell) = 0] \leq 10 \cdot \begin{cases} \tilde{q}^2 \binom{\tau}{\ell} p^{t\ell} & \text{if } \tilde{q} \leq \binom{\tau}{\ell} p^{t\ell} 2^c \\ \tau \cdot N^D \frac{\tilde{q}^3}{2^c} & \text{else} \end{cases}$$

holds.

Distance-d Special Soundness in the QROM+.

We can now use Corollary 1 to prove that the identification scheme given in Algorithms 1 and 2 has query-bounded distance-d special soundness. The special soundness extractor $\mathsf{Ext}_d^{\mathsf{Com},G}$ is defined in a straight-forward way: Given two valid transcripts with the same w and challenges of distance d, for all repetitions i where the challenges differ do the following: If the openings are not consistent, abort. Here consistency means that all openings of the same commitments agree. Otherwise, reconstruct x from the secret shares and check if $Hx = y$ and $wt(x) \leq \omega$. If not, move on to the next i, if yes, output x.

Theorem 4. *Our identification scheme Π has query-bounded distance-d special soundness. More precisely, let $\mathsf{A}^{\mathsf{Com},G}$ be a distance-d special soundness adversary making at most q_{Com} and q_G queries to its oracles Com and G, respectively, and set $q = q_{\mathsf{Com}} + q_G$ and $\tilde{q} = q + \tau \cdot N^D + 1$. Then the bounds*

$$\mathrm{Adv}^{d-\mathsf{spS}}_{\mathsf{IDS},\mathsf{Ext}}(\mathsf{A}) \leq \begin{cases} (\tau N^D + 1)\frac{\tilde{q}^2}{2^c} + \tilde{q}\binom{\tau}{d} p^{t \cdot d} & \text{in the ROM} \\ (10\tau N^D + 47)\frac{\tilde{q}^3}{2^c} + 10\tilde{q}^2\binom{\tau}{d} p^{t \cdot d} & \text{in the QROM+} \end{cases}$$

hold, where c is the output length of Com.

Proof. Given adversary A and extractor Ext_d, we construct adversaries B against the binding property of the commitment scheme and C against $\mathsf{Cheat}_{\mathcal{P},\mathbf{T},t,d}$ as follows. Let E be the event that the side-conditions for spS are fulfilled,

$$E = (\mathsf{Vrf}(\mathsf{pk}, \mathsf{w}_i, \mathsf{c}_i, \mathsf{z}_i) = 1, i \in \{1,2\} \wedge (\mathsf{w}_1 = \mathsf{w}_2) \wedge d = \mathsf{Dist}(\mathsf{c}_1, \mathsf{c}_2)).$$

The adversary B runs $((\mathsf{w}_1,\mathsf{c}_1,\mathsf{z}_1),(\mathsf{w}_2,\mathsf{c}_2,\mathsf{z}_2)) \leftarrow \mathsf{A}^{\mathsf{Com},G}(\mathsf{pk})$. If z_1 and z_2 are consistent, B outputs $\perp$. Otherwise, B uses the inconsistency to break the binding property: Let com_i and $(state_i,\rho_i) \neq (\widetilde{state}_i,\tilde{\rho}_i)$ be a commitment and two distinct openings for it that are present in $((\mathsf{w}_1,\mathsf{c}_1,\mathsf{z}_1),(\mathsf{w}_2,\mathsf{c}_2,\mathsf{z}_2))$ causing the inconsistency of z_1 and z_2. The two transcripts are valid, so the openings must be as well. B outputs $(state_i,\rho_i),(\widetilde{state}_i,\tilde{\rho}_i)$.

The adversary C runs $((\mathsf{w}_1,\mathsf{c}_1,\mathsf{z}_1),(\mathsf{w}_2,\mathsf{c}_2,\mathsf{z}_2)) \leftarrow \mathsf{A}^{\mathsf{Com},G}(\mathsf{pk})$. If A aborts, C aborts. Otherwise, C outputs the set $(state_i,\rho_i)_{i=1}^{\tau\cdot N^D} = \mathsf{z}_1 \cup \mathsf{z}_2$.

Moreover, we observe that Ext_d successfully extracts a matching secret key for pk whenever A outputs transcripts such that E holds, B fails (i.e., z_1 and z_2 are consistent), and C fails (implying that the polynomial test was cheated for at most $d-1$ challenges). The reason is that if B fails, we know that E will be able to extract $\mathbf{x}$ such that $\mathbf{Hx}=\mathbf{y}$ and the result of the polynomial zero test is 0, according to the correctness of the MPC protocol. If C fails, we additionally have that the polynomial test cannot have been cheated for all d challenges and therefore we can extract at least one $\mathbf{x}$ such that $\mathbf{Hx}=\mathbf{y}$ *and* $wt(\mathbf{x}) \leq w$.

Putting things together, we now bound the success probability of A. Consider the experiment where Ext_d, B and C use the same runs of A. The probabilities are taken over $(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{Keygen}(); ((\mathsf{w}_1,\mathsf{c}_1,\mathsf{z}_1),(\mathsf{w}_2,\mathsf{c}_2,\mathsf{z}_2)) \leftarrow \mathsf{A}^{\mathsf{Com},G}(\mathsf{pk}); \mathsf{sk}' \leftarrow \mathsf{Ext}^{\mathsf{Com},G}((\mathsf{w}_1,\mathsf{c}_1,\mathsf{z}_1),(\mathsf{w}_2,\mathsf{c}_2,\mathsf{z}_2))$ and abusing notation we define the event that B or C succeed by B and C, respectively. We bound

$$\begin{aligned}\mathrm{Adv}^{d-\mathsf{spS}}_{\mathsf{IDS},\mathsf{Ext}}(\mathsf{A}) &= \Pr[E \wedge (\mathsf{sk}',\mathsf{pk}) \notin \mathsf{Keygen}()]\\ &\leq \Pr[E \wedge (A \vee B)]\\ &\leq \Pr[A] + \Pr[B],\end{aligned}$$

where the inequality results from dropping the condition E and a union bound. Considering that we implement Com using a random oracle and applying a standard bound for collision finding in the ROM, we obtain

$$\Pr[B] \leq \frac{q^2}{2^c} \quad \text{, in the ROM.}$$

The bound in Theorem 5.29 in [9], for $k=1$, which generalizes to the QROM+ by Lemma 2 in Appendix B, yields

$$\Pr[B] \leq 47\frac{(q+1)^3}{2^c} \quad \text{, in the QROM+}$$

after simplifying the constants. The adversary C plays the $\mathsf{Cheat}_{\mathcal{P},\mathbf{T},t,d}$ game. According to Corollary 1, we thus have

$$\Pr[C] \leq \begin{cases} \tilde{q}\left(\binom{\tau}{d}p^{t\cdot d} + \tau N^D \frac{q_G}{2^c}\right) & \text{in the ROM}\\ 10\tilde{q}^2\binom{\tau}{d}p^{t\cdot d} + 10\tau N^D \frac{\tilde{q}^3}{2^c} & \text{in the QROM+}\end{cases}$$

Combining the inequalities with $q_G \leq q+1 \leq \tilde{q}$ yields the desired bound. □

4 The Signature Scheme

The main target of this paper is not the security of Π but that of the resulting signature scheme that we obtain by applying the Fiat-Shamir transform to it. This is what we focus on now. We start with necessary definitions and previous results, then we present our results for the security of the signature scheme.

```
Game UF-NMA
  $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Keygen}()$
  $(m^*, \sigma^*) \leftarrow \mathsf{A}(\mathsf{pk})$
  return $\mathsf{Vrfy}(\mathsf{pk}, m^*, \sigma^*)$

Game UF-CMA
  $\mathcal{L} \leftarrow \{\}$
  $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Keygen}()$
  $(m^*, \sigma^*) \leftarrow \mathsf{A}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(\mathsf{pk})$
  if $m^* \in \mathcal{L}$ return $0$
  return $\mathsf{Vrfy}(\mathsf{pk}, m^*, \sigma^*)$

Sign(sk, m)
  $\mathcal{L} := \mathcal{L} \cup \{m\}$
  $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
  return $\sigma$
```

Fig. 1: Games UF-CMA and UF-NMA.

```
Sign(sk, m)
  $(\mathsf{w}, \mathsf{st}) \leftarrow \mathsf{Commit}(\mathsf{sk})$
  $\mathsf{c} := \mathsf{RO}(\mathsf{w}, m)$
  $\mathsf{z} \leftarrow \mathsf{Resp}(\mathsf{sk}, \mathsf{w}, \mathsf{c}, \mathsf{st})$
  return $\sigma := (\mathsf{w}, \mathsf{z})$

Vrfy(pk, m, σ = (w, z))
  $\mathsf{c} = \mathsf{RO}(\mathsf{w}, m)$
  return $\mathsf{Vrf}(\mathsf{pk}, \mathsf{w}, \mathsf{c}, \mathsf{z})$
```

Fig. 2: Signing and verification algorithms of $\mathsf{DSS} = \mathsf{FS}[\mathsf{IDS}, \mathsf{RO}]$.

Definitions. Below we define syntax and security of digital signature schemes as well as the well known Fiat-Shamir transform. We again closely follow [17] for our definitions.

Definition 6 (Signature scheme). *A digital signature scheme* DSS *is defined as a triple of algorithms* $\mathsf{DSS} = (\mathsf{Keygen}, \mathsf{Sign}, \mathsf{Vrfy})$.

- *The probabilistic key generation algorithm* $\mathsf{Keygen}()$ *returns a key pair* $(\mathsf{pk}, \mathsf{sk})$*. We assume that* pk *defines the message space* $\mathcal{M}$.
- *The possibly probabilistic signing algorithm* $\mathsf{Sign}(\mathsf{sk}, m)$ *returns a signature* σ.
- *The deterministic verification algorithm* $\mathsf{Vrfy}(\mathsf{pk}, m, \sigma)$ *returns 1 (accept) or 0 (reject).*

UF-CMA, and UF-NMA security. We define unforgeability under chosen message attacks (UF-CMA), and unforgeability under no message attacks, i.e., with no access to a signing oracle (also known as UF-KOA, or $\mathsf{UF\text{-}CMA}_0$) success functions of a possibly quantum adversary A against DSS as

$$\mathrm{Succ}^{\mathsf{UF\text{-}X}}_{\mathsf{DSS}}(\mathsf{A}) := \Pr[1 \leftarrow \mathsf{UF\text{-}X}^{\mathsf{A}}_{\mathsf{DSS}}] \ ,$$

where the games for $X \in \{\mathsf{CMA}, \mathsf{NMA}\}$ are given in Fig. 1.

The Fiat-Shamir transform. Here we describe the standard Fiat-Shamir transform. To an identification scheme $\mathsf{IDS} = (\mathsf{Keygen}, \mathsf{Commit}, \mathsf{Resp}, \mathsf{Vrf})$ with commitment space $\mathcal{COM}$, and random oracle $\mathsf{RO} : \mathcal{COM} \times \mathcal{M} \to \mathcal{C}$ for some message space $\mathcal{M}$, we associate

$$\mathsf{FS}[\mathsf{IDS}, \mathsf{RO}] := \mathsf{DSS} := (\mathsf{Keygen}, \mathsf{Sign}, \mathsf{Vrfy}) \ ,$$

where algorithms Sign and Vrfy of DSS are defined in Fig. 2.

In [17] the following result was stated that relates the UF-NMA and UF-CMA security of a Fiat-Shamir transformed IDS in the QROM, and the HVZK property of the IDS. The bound makes use of what they call commitment entropy:

$$\gamma_{\mathsf{w}} := \mathbb{E} \max_{\mathsf{w}} \Pr[\mathsf{w}] \ ,$$

where the expectation is taken over $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Keygen}$, and the probability is taken over $(\mathsf{w}, \mathsf{st}) \leftarrow \mathsf{Commit}(\mathsf{sk})$.

Theorem 5. *[17, Theorem 3] For any (quantum)* UF-CMA *adversary* A *issuing at most* q_{S} *(classical) queries to the signing oracle* sign *and at most* q_{H} *quantum queries to* RO*, there exists a* UF-NMA *adversary* B *and a* q_{S}-HVZK *adversary* C *such that*

$$\begin{aligned}\mathrm{Succ}^{\mathsf{UF\text{-}CMA}}_{\mathsf{FS[IDS,RO]}}(\mathsf{A}) \leq \mathrm{Succ}^{\mathsf{UF\text{-}NMA}}_{\mathsf{FS[IDS,RO]}}(\mathsf{B}) + \mathrm{Adv}^{q_{\mathsf{S}}-\mathsf{HVZK}}_{\mathsf{IDS}}(\mathsf{C}) \\ + \frac{3q_{\mathsf{S}}}{2}\sqrt{(q_{\mathsf{H}} + q_{\mathsf{S}} + 1)\cdot \gamma_{\mathsf{w}}}\ ,\end{aligned} \tag{13}$$

and the running time of B *and* C *is about that of* A*, where* γ_{w} *is the maximum over the probability that* w *takes any given value. The bound given in Eq.* (13) *also holds for the modified Fiat-Shamir transform that defines challenges by letting* $\mathsf{c} := \mathsf{RO}(\mathsf{w}, m, \mathsf{pk})$ *instead of letting* $\mathsf{c} := \mathsf{RO}(\mathsf{w}, m)$.

In our actual construction, for efficiency reasons, we use a variant called Fiat-Shamir for *commitment-recoverable* IDS (see e.g., [23]), where the challenge c is sent instead of the first message w (sometimes referred to as the *commitment*). As defined in Sec. 2.1.1 a commitment-recoverable scheme like Π provides a function Rcvr that allows to recover the first message from the other two $\mathsf{Rcvr}(\mathsf{c}, \mathsf{z}) = \mathsf{w}$. In Fiat-Shamir for commitment-recoverable IDS the verifier first *recovers* w using Rcvr and then checks that indeed $\mathsf{c} = \mathsf{RO}(\mathsf{w}, m)$.

From a security perspective the two are equivalent as Rcvr allows to compute the values of a standard Fiat-Shamir signature from one resulting from a commitment recoverable scheme. The other direction, i.e., get c from w– RSP is not needed – is as simple as $\mathsf{c} = \mathsf{RO}(\mathsf{w}, m)$.

In our implementation, we use a nonce per signature, which we call *salt*. The nonce is included as a prefix in calls to all commitments, PRG operations, and Hash functions, in order to domain separate between distinct signature queries. This allows to minimize the impact of multi-target attacks. For the sake of readability, we do not consider the nonce in our formal security arguments (and therefore gain a loss in tightness) but we discuss the impact on practical security when selecting parameters.

4.1 Signature Scheme Security

The security of the signature scheme FS[Π, RO] obtained by applying the Fiat-Shamir transform to our three round IDS, can be argued in two steps as is commonly done. First, we show that we can turn any UF-NMA adversary against the scheme into an adversary against the special soundness of Π. This step follows the recipe of [11]. Afterwards, we apply Theorem 5 to argue full UF-CMA security. In [11], a tight online-extractability result is proven for the Fiat-Shamir transform of sigma-protocols with commit-and-open structure, both for simple random-oracle-based commitments and for tree commitments. The following is a specialized variant of the tree commitment variant of the result for query-bounded distance-d special soundness. We give a description of how the proof of Theorem 5.2 in [11] implies the below variant in Supp.Mat. B.

Theorem 6 (Variant of Theorem 5.2 from [11]). *Let* $\Pi^{\mathsf{Com},G}$ *be a distance-d special-sound commit-and-open identification scheme with* ϕ*-ary tree commitment with* n_c *leaves using a random oracle* Com *with output length c, splittable challenge, challenge space* $\mathcal{C}^\tau$ *and an additional random oracle G. Let further* A *be a* UF-NMA*-adversary against* FS[Π, RO] *making* q_{RO}, q_{Com} *and* q_G *queries to* RO, Com *and G respectively. Then there exists a* (q_{Com}, q_G)*-query QROM+ adversary* B *against the query-bounded distance-d special soundness of* $\Pi^{\mathsf{Com},G}$ *with* *respect to the special soundness extractor* Ext_d *of* Π *such that*

$$\begin{aligned}\mathrm{Adv}^{\mathsf{UF\text{-}NMA}}_{\mathsf{FS[IDS,RO]}}(\mathsf{A}) \leq &\Pr[\mathsf{sk}' \leftarrow \mathsf{Ext}_d \circ \mathsf{B} : (\mathsf{sk}', \mathsf{pk}) \in \mathsf{Keygen}()] \\ &+ \mathrm{Adv}^{d\mathsf{-spS}}_{\mathsf{IDS,Ext}}(\mathsf{B}) + (22 n_c \log_\phi n_c + 60) q^3 2^{-c} + 20 q^2 \frac{1}{|\mathcal{C}|^{\tau - d}},\end{aligned}$$

where $q = q_{\mathsf{Com}} + q_{\mathsf{RO}}$*. The runtime of* B *is bounded as* $\mathsf{TIME}(\mathsf{B}) \leq \mathsf{TIME}(\mathsf{A}) + \gamma(q + q_G)^2))$*, where* γ *is polynomial in the input and output lengths of the random oracles.*

As a corollary, we get a UF-NMA-security result for our signature scheme in the QROM. We note that this corollary does not need to refer to the QROM+ anymore, as it combines a reduction to an adversary agains the query-bounded distance-d special soundness in the QROM+ with an explicit bound on the success probability of such an adversary.

Corollary 2. *Let* A *be a* $\mathsf{UF\text{-}NMA}$*-adversary against* $\mathsf{FS}[\Pi, \mathsf{RO}]$ *that makes* $q_{\mathsf{RO}} \geq \tau \cdot N^D + 1$, q_{Com} *and* q_G *quantum queries to* RO, Com *and* G *respectively. Then for all* $d = 0, 1, \dots, \tau$ *we get*

$$\mathrm{Adv}^{\mathsf{UF\text{-}NMA}}_{\mathsf{FS[IDS,RO]}}(\mathsf{A}) \leq \epsilon_{\mathrm{SD}} + (32 \tau N^D + 107) \frac{q^3}{2^c} + 10 \cdot q^2 \binom{\tau}{d} p^{t \cdot d} + 20 q^2 \frac{1}{N^{D \cdot (\tau - d)}}.$$

Here, ϵ_{SD} *is the maximal success probability that an adversary with runtime* $\mathsf{TIME}(\mathsf{A}) + \mathsf{TIME}(\mathrm{CompOr}(q)) + \mathsf{TIME}(\mathsf{Ext}_d)$*, where* $\mathsf{TIME}(\mathrm{CompOr}(q))$ *is the runtime of a compressed oracle simulation for* q *queries, can solve syndrome decoding. Also* $q = q_{\mathsf{Com}} + q_{\mathsf{RO}} + q_G$ *is the total number of random oracle queries of* A*,* c *is the output length of* Com*, and the atomic polynomial zero test false-positive probability* p *is defined and bounded in Equation* (11) *and Equation* (12)*.*

Note that the restriction on q_{RO} is almost without loss of generality ($\tau \cdot N^D + 1$, q_{Com} queries to RO can be made in time similar to, e.g., the signing time) and is only needed to allow for a less cluttered bound expression.

Proof. Π uses a commitment that works by hashing each state with some randomness, and then hashing all theses hashes together to produce a single collective commitment. This is a $\tau \cdot N^D$-ary tree commitment with $\tau \cdot N^D$ leaves, so we can apply Theorem 6 for these parameters. Plugging in the number of possible split-challenges N^D, we get

$$\begin{aligned}\mathrm{Adv}^{\mathsf{UF\text{-}NMA}}_{\mathsf{FS[IDS,RO]}}(\mathsf{A}) \leq &\Pr[\mathsf{sk}' \leftarrow \mathsf{Ext}_d \circ \mathsf{B} : (\mathsf{sk}', \mathsf{pk}) \in \mathsf{Keygen}()] \\ &+ \mathrm{Adv}^{d\mathsf{-spS}}_{\mathsf{IDS,Ext}}(\mathsf{B}) + (22 n_c + 60) q^3 2^{-c} + 20 q^2 N^{-D \cdot (\tau - d)},\end{aligned}$$

where c is the length of the commitments. $\Pr[\mathsf{sk}' \leftarrow \mathsf{Ext}_d \circ \mathsf{B} : (\mathsf{sk}', \mathsf{pk}) \in \mathsf{Keygen}()]$ is the success probability of $\mathsf{Ext}_d \circ \mathsf{B}$ as a syndrome decoding algorithm and thus

$$\Pr[\mathsf{sk}' \leftarrow \mathsf{Ext}_d \circ \mathsf{B} : (\mathsf{sk}', \mathsf{pk}) \in \mathsf{Keygen}()] \leq \epsilon_{\mathrm{SD}}.$$

Setting $\tilde{q} = q_{\mathsf{Com}} + q_G + N^D + 1$, by Theorem 4, we have the bound

$$\begin{aligned}\mathrm{Adv}^{d\mathsf{-spS}}_{\mathsf{IDS,Ext}}(\mathsf{B}) \leq &(10 \tau N^D + 47) \frac{\tilde{q}^3}{|\mathfrak{C}|} + 10 \tilde{q}^2 \binom{\tau}{d} p^{t \cdot d} \\ \leq &(10 \tau N^D + 47) \frac{q^3}{|\mathfrak{C}|} + 10 q^2 \binom{\tau}{d} p^{t \cdot d}\end{aligned}$$

where the second inequality holds because $\tilde{q} \leq q$ by assumption on q_{RO}. Combining the inequalities yields the desired bound. □

Finally we obtain a bound for the $\mathsf{UF\text{-}CMA}$ security as follows:

Corollary 3. *Let* A *be a* $\mathsf{UF\text{-}CMA}$*-adversary against* $\mathsf{FS}[\Pi, \mathsf{RO}]$ *that makes* $q_{\mathsf{RO}} \geq \tau \cdot N^D + 1$, q_{PRG}, q_{Com} *and* q_G *quantum queries to* RO, PRG, Com *and* G *respectively, and* q_{S} *(classical) signing queries. Then for all* $d = 0, 1, \ldots, \tau$,

$$\mathrm{Adv}^{\mathsf{UF\text{-}CMA}}_{\mathsf{FS[IDS,RO]}}(\mathsf{A}) \leq \epsilon_{\mathrm{SD}} + (32\tau N^D + 107)q^3 2^{-c} + 10 \cdot q^2 \binom{\tau}{d} p^{t \cdot d} + 20q^2 \frac{1}{N^{D\cdot(\tau-d)}}$$
$$+ q_{\mathsf{S}}\tau \left(16 q_{\mathsf{Com}} 2^{-r/2} + \log(N^D - 1)\frac{(q_{\mathsf{PRG}} + q_{\mathsf{S}}\tau)^2}{2^n}\right) + \frac{3q_{\mathsf{S}}}{2}\sqrt{\frac{q_{\mathsf{RO}} + q_{\mathsf{S}} + 1}{2^n}}, \tag{14}$$

Here ϵ_{SD} *is the maximal success probability that an adversary that runs in time* $\mathsf{TIME}(\mathsf{A}) + \mathsf{TIME}(\mathrm{CompOr}(q)) + \mathsf{TIME}(\mathsf{Ext}_d)$*, where* $\mathsf{TIME}(\mathrm{CompOr}(q))$ *is the runtime of a compressed oracle simulation for* q *queries, can solve syndrome decoding. Moreover,* $q = q_{\mathsf{Com}} + q_{\mathsf{RO}} + q_G$ *is the total number of random oracle queries of* A*,* c *is the output length of* Com*, and the atomic polynomial zero test false-positive probability* p *is defined in Equation* (11) *and bounded in Equation* (12)*,* n *is the seed length of* $\mathsf{TreePRG}$*,* r *is the length of commitment randomness.*

Proof. This follows by applying Theorem 5 to Corollary 2. Moreover, we plug in the HVZK bound from Theorem 1 and observe that γ_{W}, the entropy of the commitment messages, in Π is n bits. Further we note that the reduction in the HVZK proof makes up to τq_{S} additional calls to $\mathsf{TreePRG}$, and use the bound for the security of $\mathsf{TreePRG}$ given in Sec. 2.1. We finally apply the QROM bound for hiding of Com from [25] and note that for the security of PRG when modeled as QRO, a standard search bound applies. The reason is that without seeing an input that maps to a challenge, the adversary can do no better than guessing. □

Discussion. Corollary 3 provides a tight bound for UF-CMA security in terms of the hardness of syndrome decoding. The additive terms are all benign. The first additive term is matched by a collision finding attack on the hash function used for commitment [8], up to the constant preceding $q^3 2^{-c}$. The second and third additive terms are similar to the ones appearing in the bounds in [14, 2], and are matched by a "divide-and-conquer" attack: An adversary can first search for polynomials allowing them to cheat d out of τ polynomial zero tests, and then search for a message to be signed that allows cheating the MPCitH proof of the remaining $\tau - d$ repetitions. When $p \simeq (1/N^D)$ the divide and conquer attack is most powerful, with $d = \frac{\tau}{2}$. But when $p \ll (1/N^D)$ the attack complexity tends towards $(1/N^D)^\tau$, and parameters in section 5 are selected accordingly. The $16q_{\mathsf{Com}}2^{r/2}$ term in the second line stems from the computational hiding property of the commitments and is matched by a Grover search for the used commitment randomness. The term in the last line is negligible compared to the last term in the second line, which is matched by a Grover search attack on PRG.

Comparing the bound in Corollary 3 to the ROM bound proven in [2], we observe that each term in Corollary 3 either has been neglected in [2] (e.g. the term corresponding to the hiding security of RO-based committments), or leaves at most the possibility for a quadratic speed-up due to Grover search, up to small mutiplicative constants (e.g. the terms characterizing the security of the polynomial identity test and MPCitH proof).

5 Performance

The tweaks introduced to the original Hypercube-SDitH scheme not only make possible a security proof in the QROM setting, they also have a positive impact on the performance of the scheme. They allow to slightly reduce signature size and to significantly reduce the online signing time.

Algorithm 5 PoW - Proof-of-Work for challenge derivation

```
Input: Commitment-message hash $h_{\mathsf{w}} \leftarrow H(\mathsf{w}, m)$, number of iterations $2^{k_{iter}}$.
1: $dgst \leftarrow h_{\mathsf{w}}$
2: for $ctr \in \{0, \ldots, 2^{k_{iter}} - 1\}$ do
3:     $dgst \leftarrow H(dgst \| h_{\mathsf{w}})$
4: return $dgst$
```

Signing with one hash. The original Hypercube-SDitH signature is based on a five round identification scheme with two verifier challenges, the first challenge being between the evaluation points. When applying the Fiat-Shamir transform, the message is required to compute the first challenge, so the online phase of the original Hypercube-SDitH signature scheme (the part that requires presence of the message) includes both the MPC computation and the MPC party opening.

In the three round version, the online phase of the signature corresponds just to one random oracle call and the MPC party opening. In practice this can be as fast as one hash call, plus arithmetic to compute $[\![\alpha]\!]_{\mathsf{c}}, [\![\beta]\!]_{\mathsf{c}}$, plus building sibling paths required for the openings.

Reducing the online cost of a signature to just one hash is amazing. However, some applications may prefer smaller signatures at the cost of a slightly slower online phase. For this we introduce an online-time - signature-size trade-off which *stretches* the challenge generation time in order to reduce the signature size.

Proof-of-Work. Our trade-off is inspired by the Proof-of-Work (PoW) technique used in SPHINCS+C [24]. In our scheme, we exchange the counter based PoW, with $2^{k_{iter}}$ times iterative hashing. To generate the challenge c, we first generate the hash $h_{\mathsf{w}} \leftarrow H(\mathsf{w}, m)$. We then apply the PoW routine (Algorithm 5) to increase the cost of the hash computation, such that the final challenge is $\mathsf{c} \leftarrow PoW(h_{\mathsf{w}})$.

The proof-of-work technique (PoW) does not change the applicability of the security proof as it only replaces one hash function by a more costly one. However, the choice of parameters according to the PoW cannot be supported by our security proof. Because of this, we only introduce the proof-of-work trick here as an optimization. To be covered by the security proof, we would need our bound to distinguish between queries made to the different functions that are modeled as random oracles. This is not possible with our current proof as we are using the technique from [11] in a black box manner where possible and [11] does not distinguish between queries to different functions.

We note, that unlike the PoW counter-based solution of [24], there is no variability in the runtime of our PoW algorithm. The downside, is that unlike the counter-based solution, our iterative solution adds the same running time to the verifier. However, at the same time it reduces the running time as we are able to reduce the number of parallel repetitions τ of Π (c.f.,Table 1).

For concrete parameters, we increase the cost of the message-hash query by $2^{k_{iter}}$, but can in turn reduce the requirement on D and τ to $\approx (1/N^D)^\tau \leq 2^{-\lambda} \cdot 2^{k_{iter}} = 2^{-\lambda + k_{iter}}$. As discussed below, we use $N = 2$. Choosing $k_{iter} = D$ increases the attack complexity by a factor 2^D, and each additional parallel repetition increases the attack complexity by a factor $\approx N^D = 2^D$. Thus selecting $k_{iter} = D$ allows us to use $\tau' = \tau - 1$ at the same security level. This means that by using the PoW, we need one less repetition of the protocol and can thereby reduce the overall size of the signature. However, this is clearly not the only possible choice for k_{iter}. Reasonable values would be any multiple of D, as $k_{iter} = kD$ means that we can run the protocol with $\tau' = \tau - k$ parallel repetitions. Moreover, as D and τ are integers, we might find cases where $(1/N^D)^\tau$ is slightly larger than $2^{-\lambda}$, forcing us to increase the parameters and signature size, also this could be compensated for using the PoW. Thereby, we can increase our degree of freedom in choosing parameters, possibly resulting in better-optimized variants.

Parameters. For our implementation we stick with the parameters from [14] also used in [2]. Our security bound is (except for some small constants) the same as in [2] up to the generic Grover search and quantum collision finding bounds. These were already (heuristically) considered in the parameter selection by the previous works. For security we target NIST security level I which refers to 128bit security against conventional attacks and 64 bit security against quantum attacks. We use the Variant 3 parameters of the original SDitH scheme (also used in the Hypercube scheme). These parameters use the syndrome decoding problem in $\mathbb{F}_{SD} = \mathbb{F}_{2^8}$ with $m = 256, k = 128$, and $w = 80$. When looking at the original Hypercube proposal, they fix $N = 2$ and define further parameters applying different trade-offs between signature-size and speed (chosen to match the equally named parameter sets proposed in [14]). We focus on the "Short" ($D = 8, \tau = 17$) and "Shorter" ($D = 12, \tau = 12$) configurations from Hypercube-SDitH since they offer the most interesting trade-offs in our opinion. We use these parameters as baseline to demonstrate the impact of our results.

It remains to fix the values for the seed length n, the commitment randomness length r, and the commitment length c. For these values we use $n = r = 128$ bit and $c = 256$ bit. Taking a close look at the terms of the sum on the RHS of Equation (14) shows that our choice for the values of n and r are ignoring the $q_{\mathsf{S}}\tau\log(N^D - 1)$ and $q_{\mathsf{S}}\tau$ factors respectively. Examining the proof shows that these factors are caused by the hybrid arguments which reflect multi-target attacks. When modeling the PRG and the commitment as random oracles, these attacks can be mitigated using domain separation (as for example demonstrated in [21]). Hence, as mentioned previously, our implementation makes use of an additional random 128 bit nonce, called *salt*, which is freshly chosen for each signature. This nonce is used as a prefix to the inputs to the PRG, the commitment, and the hash function. Thereby, it domain-separates these calls over different signature calls, effectively removing the need of the factor q_{S} in the bound. This leaves as worst case the $\tau\log(N^D - 1)$ factor to be considered for the seed length. For our parameters, this accounts to a less then 8 bit loss in security. Given that we count hash function calls as a single operation while this takes more than 256 bit operations, we consider this compensated for. We note that we did not consider this domain separation in our proof as it would significantly hurt readability of the arguments at rather limited novelty given that this kind of solution was discussed already in previous works.

Implementation Results. We base our implementation of the tweaked scheme on top of the previous Hypercube-SDitH implementation from [2], using the XKCP library (SHAKE) for all symmetric primitives (hashes, commitments, and PRGs). Before making modifications, we thoroughly examined the Hypercube-SDitH implementation regarding constant execution time and identified and hardened several key routines that rely on signer-private information and could leak information if done naively.

Next, we benchmarked the original 5-round Hypercube-SDitH scheme with the updated implementation to obtain reference values. Then, we benchmarked the 3-round version of this scheme (Ours - Vanilla), and the same scheme but applying the PoW algorithm above (Ours - PoW) with parameter $k_{iter} = D$.

For the benchmarks we used an optimized implementation that leverages AVX2 instructions to parallelize SHAKE and SHA3 calls. The experiments ran on an Intel Xeon E-2378 with frequency fixed at 2.6 GHz and Turbo Boost disabled. We prepared a test routine that runs keygen, sign, and verify on a fixed text input. Finally, we run the test routine for 100 times sequentially on a single CPU core and average the timing measurement results.

Table 1: Implementation benchmarks of Hypercube-SDitH vs our tweaked scheme for NIST security level I. For the PoW, the parameter $k_{iter} = D$ is used.

Scheme	**Aim**	**Signature Size (bytes)**	**Parameters**				**Sign Time (in ms)**			**Verify Time**
			$\lvert\mathbb{F}_{\text{points}}\rvert$	t	D	τ	Offline	Online	Total	(in ms) Total
Hypercube-SDitH [2]	Short	8464	2^{24}	5	8	17	3.83	0.68	4.51	4.16
	Shorter	6760	2^{24}	5	12	12	44.44	0.60	45.04	42.02
Ours Vanilla	Short	8464	2^{24}	5	8	17	4.45	0.049	4.50	4.17
	Shorter	6760	2^{24}	5	12	12	44.98	0.080	45.06	42.02
Ours PoW	Short	7968	2^{24}	5	8	16	4.20	0.14	4.34	4.00
	Shorter	6204	2^{24}	5	12	11	41.06	1.49	42.55	39.75

The implementation is available at https://github.com/sandbox-quantum/sdith-impl-release.

References

[1] C. Aguilar Melchor, N. Aragon, S. Bettaieb, L. Bidoux, O. Blazy, J.-C. Deneuville, P. Gaborit, E. Persichetti, G. Zémor, J. Bos, A. Dion, J. Lacan, J.-M. Robert, and P. Veron. *HQC*. Tech. rep. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions. National Institute of Standards and Technology, 2022.

[2] C. Aguilar Melchor, N. Gama, J. Howe, A. Hülsing, D. Joseph, and D. Yue. "The Return of the SDitH". In: *EUROCRYPT 2023, Part V*. Ed. by C. Hazay and M. Stam. Vol. 14008. LNCS. Springer, Heidelberg, Apr. 2023, pp. 564–596. DOI: 10.1007/978-3-031-30589-4_20.

[3] M. R. Albrecht, D. J. Bernstein, T. Chou, C. Cid, J. Gilcher, T. Lange, V. Maram, I. von Maurich, R. Misoczki, R. Niederhagen, K. G. Paterson, E. Persichetti, C. Peters, P. Schwabe, N. Sendrier, J. Szefer, C. J. Tjhai, M. Tomlinson, and W. Wang. *Classic McEliece*. Tech. rep. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-4-submissions. National Institute of Standards and Technology, 2022.

[4] N. Aragon, P. Barreto, S. Bettaieb, L. Bidoux, O. Blazy, J.-C. Deneuville, P. Gaborit, S. Gueron, T. Guneysu, C. Aguilar Melchor, R. Misoczki, E. Persichetti, N. Sendrier, J.-P. Tillich, G. Zémor, V. Vasseur, S. Ghosh, and J. Richter-Brokmann. *BIKE*. Tech. rep. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions. National Institute of Standards and Technology, 2022.

[5] E. Berlekamp, R. McEliece, and H. Van Tilborg. "On the inherent intractability of certain coding problems (corresp.)" In: *IEEE Transactions on Information Theory* 24.3 (1978), pp. 384–386.

[6] W. Beullens. "Breaking Rainbow Takes a Weekend on a Laptop". In: *CRYPTO 2022, Part II*. Ed. by Y. Dodis and T. Shrimpton. Vol. 13508. LNCS. Springer, Heidelberg, Aug. 2022, pp. 464–479. DOI: 10.1007/978-3-031-15979-4_16.

[7] D. Boneh, Ö. Dagdelen, M. Fischlin, A. Lehmann, C. Schaffner, and M. Zhandry. "Random Oracles in a Quantum World". In: *ASIACRYPT 2011*. Ed. by D. H. Lee and X. Wang. Vol. 7073. LNCS. Springer, Heidelberg, Dec. 2011, pp. 41–69. DOI: 10.1007/978-3-642-25385-0_3.

[8] G. Brassard, P. Høyer, and A. Tapp. “Quantum Cryptanalysis of Hash and Claw-Free Functions”. In: *LATIN ’98*. Ed. by C. L. Lucchesi and A. V. Moura. Vol. 1380. Lecture Notes in Computer Science. Springer, 1998, pp. 163–169. DOI: 10.1007/BFb0054319.

[9] K.-M. Chung, S. Fehr, Y.-H. Huang, and T.-N. Liao. “On the Compressed-Oracle Technique, and Post-Quantum Security of Proofs of Sequential Work”. In: *EUROCRYPT 2021, Part II*. Ed. by A. Canteaut and F.-X. Standaert. Vol. 12697. LNCS. Springer, Heidelberg, Oct. 2021, pp. 598–629. DOI: 10.1007/978-3-030-77886-6_21.

[10] T. Debris-Alazard, N. Sendrier, and J.-P. Tillich. “Wave: A New Family of Trapdoor One-Way Preimage Sampleable Functions Based on Codes”. In: *ASIACRYPT 2019, Part I*. Ed. by S. D. Galbraith and S. Moriai. Vol. 11921. LNCS. Springer, Heidelberg, Dec. 2019, pp. 21–51. DOI: 10.1007/978-3-030-34578-5_2.

[11] J. Don, S. Fehr, C. Majenz, and C. Schaffner. “Efficient NIZKs and Signatures from Commit-and-Open Protocols in the QROM”. In: *CRYPTO 2022, Part II*. Ed. by Y. Dodis and T. Shrimpton. Vol. 13508. LNCS. Springer, Heidelberg, Aug. 2022, pp. 729–757. DOI: 10.1007/978-3-031-15979-4_25.

[12] J. Don, S. Fehr, C. Majenz, and C. Schaffner. “Online-Extractability in the Quantum Random-Oracle Model”. In: *EUROCRYPT 2022, Part III*. Ed. by O. Dunkelman and S. Dziembowski. Vol. 13277. LNCS. Springer, Heidelberg, 2022, pp. 677–706. DOI: 10.1007/978-3-031-07082-2_24.

[13] S. Even, O. Goldreich, and S. Micali. “On-Line/Off-Line Digital Schemes”. In: *CRYPTO’89*. Ed. by G. Brassard. Vol. 435. LNCS. Springer, Heidelberg, Aug. 1990, pp. 263–275. DOI: 10.1007/0-387-34805-0_24.

[14] T. Feneuil, A. Joux, and M. Rivain. “Syndrome Decoding in the Head: Shorter Signatures from Zero-Knowledge Proofs”. In: *CRYPTO 2022, Part II*. Ed. by Y. Dodis and T. Shrimpton. Vol. 13508. LNCS. Springer, Heidelberg, Aug. 2022, pp. 541–572. DOI: 10.1007/978-3-031-15979-4_19.

[15] A. Fiat and A. Shamir. “How to Prove Yourself: Practical Solutions to Identification and Signature Problems”. In: *CRYPTO’86*. Ed. by A. M. Odlyzko. Vol. 263. LNCS. Springer, Heidelberg, Aug. 1987, pp. 186–194. DOI: 10.1007/3-540-47721-7_12.

[16] O. Goldreich, S. Goldwasser, and S. Micali. “How to Construct Random Functions (Extended Abstract)”. In: *25th FOCS*. IEEE Computer Society Press, Oct. 1984, pp. 464–479. DOI: 10.1109/SFCS.1984.715949.

[17] A. B. Grilo, K. Hövelmanns, A. Hülsing, and C. Majenz. “Tight Adaptive Reprogramming in the QROM”. In: *ASIACRYPT 2021, Part I*. Ed. by M. Tibouchi and H. Wang. Vol. 13090. LNCS. Springer, Heidelberg, Dec. 2021, pp. 637–667. DOI: 10.1007/978-3-030-92062-3_22.

[18] E. Grumbling and M. Horowitz. *Quantum Computing: Progress and Prospects*. 1st. National Academies of Sciences, Engineering, and Medicine. The National Academies Press, Apr. 2019. ISBN: 9780309479691. DOI: 10.17226/25196.

[19] K. Hövelmanns, A. Hülsing, and C. Majenz. “Failing Gracefully: Decryption Failures and the Fujisaki-Okamoto Transform”. In: *ASIACRYPT 2022, Part IV*. LNCS. Springer, Heidelberg, Dec. 2022, pp. 414–443. DOI: 10.1007/978-3-031-22972-5_15.

[20] A. Hulsing, D. J. Bernstein, C. Dobraunig, M. Eichlseder, S. Fluhrer, S.-L. Gazdag, P. Kampanakis, S. Kolbl, T. Lange, M. M. Lauridsen, F. Mendel, R. Niederhagen, C. Rechberger, J. Rijneveld, P. Schwabe, J.-P. Aumasson, B. Westerbaan, and W. Beullens. *SPHINCS+*. Tech. rep. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022. National Institute of Standards and Technology, 2022.

[21] A. Hülsing, J. Rijneveld, and F. Song. "Mitigating Multi-target Attacks in Hash-Based Signatures". In: *PKC 2016, Part I*. Ed. by C.-M. Cheng, K.-M. Chung, G. Persiano, and B.-Y. Yang. Vol. 9614. LNCS. Springer, Heidelberg, Mar. 2016, pp. 387–416. DOI: 10.1007/978-3-662-49384-7_15.

[22] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. "Zero-knowledge from secure multiparty computation". In: *39th ACM STOC*. Ed. by D. S. Johnson and U. Feige. ACM Press, June 2007, pp. 21–30. DOI: 10.1145/1250790.1250794.

[23] E. Kiltz, V. Lyubashevsky, and C. Schaffner. "A Concrete Treatment of Fiat-Shamir Signatures in the Quantum Random-Oracle Model". In: *EUROCRYPT 2018, Part III*. Ed. by J. B. Nielsen and V. Rijmen. Vol. 10822. LNCS. Springer, Heidelberg, 2018, pp. 552–586. DOI: 10.1007/978-3-319-78372-7_18.

[24] M. A. Kudinov, A. Hülsing, E. Ronen, and E. Yogev. "SPHINCS+C: Compressing SPHINCS+ With (Almost) No Cost". In: *IACR Cryptol. ePrint Arch.* (2022), p. 778. URL: https://eprint.iacr.org/2022/778.

[25] D. Leichtle. *Post-quantum signatures from identification schemes*. Master's thesis, Technische Universiteit Eindhoven. https://pure.tue.nl/ws/portalfiles/portal/125545339/Dominik_Leichtle_thesis_final_IAM_307.pdf. 2018.

[26] V. Lyubashevsky, L. Ducas, E. Kiltz, T. Lepoint, P. Schwabe, G. Seiler, D. Stehlé, and S. Bai. *CRYSTALS-DILITHIUM*. Tech. rep. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022. National Institute of Standards and Technology, 2022.

[27] R. J. McEliece. *A public-key cryptosystem based on algebraic coding theory*. The Deep Space Network Progress Report 42-44. https://ipnpr.jpl.nasa.gov/progress_report2/42-44/44N.PDF. Jet Propulsion Laboratory, California Institute of Technology, 1978, pp. 114–116.

[28] M. Mosca. "Cybersecurity in an Era with Quantum Computers: Will We Be Ready?" In: *IEEE Security & Privacy* 16 (Sept. 2018), pp. 38–41. DOI: 10.1109/MSP.2018.3761723.

[29] NIST. *National Institute for Standards and Technology. PQC Standardization Process: Announcing Four Candidates to be Standardized, Plus Fourth Round Candidates*. https://csrc.nist.gov/News/2022/pqc-candidates-to-be-standardized-and-round-4. Mar. 2022.

[30] NIST. *Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process*. https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf. 2016.

[31] T. Prest, P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang. *FALCON*. Tech. rep. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022. National Institute of Standards and Technology, 2022.

[32] J. Stern. "Designing Identification Schemes with Keys of Short Size". In: *CRYPTO'94*. Ed. by Y. Desmedt. Vol. 839. LNCS. Springer, Heidelberg, Aug. 1994, pp. 164–173. DOI: 10.1007/3-540-48658-5_18.

[33] G. Zaverucha, M. Chase, D. Derler, S. Goldfeder, C. Orlandi, S. Ramacher, C. Rechberger, D. Slamanig, J. Katz, X. Wang, V. Kolesnikov, and D. Kales. *Picnic*. Tech. rep. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions. National Institute of Standards and Technology, 2020.

[34] M. Zhandry. "How to Record Quantum Queries, and Applications to Quantum Indifferentiability". In: *CRYPTO 2019, Part II*. Ed. by A. Boldyreva and D. Micciancio. Vol. 11693.

LNCS. Springer, Heidelberg, Aug. 2019, pp. 239–268. DOI: 10.1007/978-3-030-26951-7_9.

A Notation summary

Table 2: Notation

Indices:	
i	index of a leaf party, in $\{0,\dots,N^D-1\}$
i^*	index of challenge party, which remains hidden
$(i_1,\dots,i_D)$	representation of i on dimension D hypercube with side N
(k,j)	index of a main party in $\{0,\dots,D-1\}\times\{0,\dots,N-1\}$, where k indexes the hypercube dimension
$\mathbb{F}_{\text{poly}}$	field of coefficients from which S,Q,P,F are drawn
$\mathbb{F}_{\text{points}}$	field from which $\alpha,\beta,v,r,\epsilon$ are drawn
MPC:	
Π	the MPC computation, described in Alg 6
a,b,c	elements of the Beaver triplet such that $a\cdot b=c$
α,β,v	Communications output, drawn from $\mathbb{F}_{\text{points}}$
(r,ε)	(evaluation point, masking point)
$[\![X]\!]_i$	i^{th} leaf party secret share of X
$[X]_{kj}$	main party (k,j)'s secret share of X
$\{[X]_{kj}\}_{k,j=0,\dots,N}$	a full main party sharing, such that all shares add up to give X
Parameters:	
λ	security parameter
ϵ	soundness
Syndrome Decoding proof:	
S,Q,P,F	polynomials in $\mathbb{F}_{\text{poly}}$ which encode the syndrome decoding proof
aux	uncompressed secret shares of leaf party $i=N^D-1$, $[\![S]\!]\|[\![Q]\!]\|[\![P]\!]\|[\![a]\!]\|[\![b]\!]\|[\![c]\!]$
$(state_i,\rho_i)$	state and commitment randomness of a leaf party. For $i\neq N^D-1$, $state_i$ is a pseudo-random seed, and $state_{N^D-1}=(seed_{N^D-1}\|\|aux)$
Polynomial test:	
$\mathfrak{P}$	polynomial domain (abstract notation)
$\mathcal{P}(\mathbf{P})$	predicate on polynomials $\mathbf{P}$
$\mathfrak{S},(\mathfrak{C})$	state (commitment) space

B Additional QROM definitions and proofs

In the following, we write $D[x\mapsto y]$ for the database where the pair (x,y) replaces any pre-existing pair (x,y') and is added otherwise. We write $D[x\mapsto\bot]$ for the database D with any pair (x,y') removed. More generally, we freely use notation from [9].

To formalize the QROM security reduction for our signature scheme in a modular way, we need to prove certain intermediate results for algorithms in a slightly strengthened version of the QROM which we call QROM+. In this model, we instantiate the quantum-accessible random oracle with a compressed-oracle-based simulation. After making all its queries to the oracle, the adversary can also request a computational basis measurement of the entire oracle database. We begin by formalizing this model, which is somewhat similar in spirit to the extractable QRO model defined in [12] and further studied in [19].

Definition 7 (QROM+). *A QROM+ algorithm A^F trying to fulfil a predicate P^F is a pair of algorithms (A_0^F,A_1) which are run in the following experiment.*

1. *The first stage of* A *gets access to a compressed oracle simulation of* F *with database* D *which we denote by* F_D *and outputs a quantum state* st, $st \leftarrow \mathsf{A}_0^{F_D}(inp)$, *where* inp *is the input* A *expects.*
2. *The database* D *is measured in the computational basis to obtain outcome* $\hat{D}$, *keeping the post-measurement state on* D.
3. *The second stage of* A *is run on inputs* st *and* $\hat{D}$, $out \leftarrow \mathsf{A}_1(st, \hat{D})$ *where* out *is* A*s output.*
4. *The predicate* P *is evaluated, making oracle queries to* F_D *(and ignoring the fact that* D *has been measured),* $b \leftarrow P^{F_D}(out)$.

A *is successful if* $b = 0$.

We remark that this definition can be extended to more complex settings, but the above definition suffices for our purposes.

To simplify the formalism, we introduce the purified computational basis measurement operator

$$M = \mathsf{CNOT}^{\otimes \mathcal{N}},$$

acting on the database register D and a fresh auxiliary register D' of the same size. Here, $\mathcal{N}$ is the number of qubits in D. For database properties P and Q, denote by $\hat{P}$ and $\hat{Q}$ the corresponding projectors acting on a compressed oracle database. Suppose that $PQ = 0$. Then it also holds that

$$\hat{P}_D M_{DD'} \hat{Q}_D = 0. \tag{15}$$

We recall the definition of the quantum transition capacity from [9] (we only need the non-parallel case).

Definition 8 (Quantum transition capacity). *Let* P, P' *be two database properties. Then, the* quantum transition capacity *is defined as*

$$[\![P \xrightarrow{q} P']\!] := \sup_{U_1,\ldots,U_{q-1}} \|P' \mathsf{cO}\, U_{q-1}\, \mathsf{cO} \cdots \mathsf{cO}\, U_1\, \mathsf{cO}\, P\| \,.$$

where the supremum is over all adversary register sizes and all unitaries $U_1, \ldots, U_{q-1}$ *acting on the adversary's registers. We write*

$$[\![P \to P']\!] := [\![P \xrightarrow{1} P']\!] = \|P' \mathsf{cO} P\|$$

To analyze the query complexity of search tasks in the QROM+, we define the transition capacity with a final measurement,

$$[\![P \xrightarrow{q} P']\!]_M := \sup_{U_1,\ldots,U_{q-1},U_q} \|P' M U_q \mathsf{cO}\, U_{q-1}\, \mathsf{cO} \cdots \mathsf{cO}\, U_1\, \mathsf{cO}\, P\| \,.$$

Here it is understood that M is the only operator that acts on D'.

Any bounds on the success probability for oracle search tasks derived using the framework of [9] hold in the QROM+ as well. This is a direct consequence of the following

Lemma 2.

$$[\![P \xrightarrow{q} P']\!]_M = [\![P \xrightarrow{q} P']\!].$$

Proof. The operators M and P' are both diagonal in the computational basis on D (for M, this holds in the sense that $M_{DD'} = \sum_{\hat{D}} |\hat{D}\rangle\langle\hat{D}|_D \otimes M_{D'}^{(\hat{D})}$ for unitaries $M^{(\hat{D})}$). The operators therefore commute, $P'_D M_{DD'} = M_{DD'} P'_D$. Also P' acts on D only, so it commutes with any unnitary acting on the adversary registers only. We thus have

$$\begin{aligned}
\llbracket P \xrightarrow{q} P' \rrbracket_M &= \sup_{U_1,\ldots,U_{q-1},U_q} \| P' M U_q \mathsf{cO}\, U_{q-1}\, \mathsf{cO} \cdots \mathsf{cO}\, U_1\, \mathsf{cO}\, P \| \\
&= \sup_{U_1,\ldots,U_{q-1},U_q} \| M U_q P' \mathsf{cO}\, U_{q-1}\, \mathsf{cO} \cdots \mathsf{cO}\, U_1\, \mathsf{cO}\, P \| \\
&= \sup_{U_1,\ldots,U_{q-1},U_q} \| P' \mathsf{cO}\, U_{q-1}\, \mathsf{cO} \cdots \mathsf{cO}\, U_1\, \mathsf{cO}\, P \| \\
&= \llbracket P \xrightarrow{q} P' \rrbracket .
\end{aligned}$$

□

The following Lemma combines several results from [9] to avoid the additive error term that exists in, e.g., Theorem 5.7 in [9], and generalizes the technique to the QROM+.

Lemma 1 (A compressed oracle query bound lemma). *Let $F : \mathcal{X} \to \mathcal{Y}$ be a random oracle and let $\mathcal{P}^F$ be a predicate on some set $\mathcal{Z}$ that can be computed using at most $q_{\mathcal{P}}$ classical queries to F. Let further A^F be a QROM+ algorithm making at most q quantum queries to F and outputing $z \in \mathcal{Z}$. Then*

$$\sqrt{\Pr_{z \leftarrow \mathsf{A}^F}[P(z)]} \leq \sum_{k=1}^{q+q_{\mathcal{P}}} \max_{\substack{x,D:\\ |D|\leq k\\ \neg\mathsf{Found}(D)}} \sqrt{10 \Pr_{u \leftarrow \mathcal{Y}}[\mathsf{Found}_{\mathcal{P}}(D[x \mapsto u])]} \tag{8}$$

where $\mathsf{Found}_{\mathcal{P}}$ *is the database property*

$$\mathsf{Found}_{\mathcal{P}} = (\exists z \in \mathcal{Z} : \mathcal{P}^D(z)) \tag{9}$$

and $\mathcal{P}^D$ is the algorithm that computes $\mathcal{P}$ but makes queries to D instead of F, and if any query returns $\perp$, $\mathcal{P}^D$ ouptuts 'false'.

Proof. Without loss of generality, we instantiate F via a random oracle $\hat{F} : \mathcal{X} \to \mathcal{Y} \times \mathcal{Y}'$ by setting $F(x) = (\hat{F}(x))_1$. Theorem 5.7 in [9] generalizes to the QROM+ in a straightforward manner, with an unchanged bound, by Lemma 2. Applying that generalization to the algorithm A' that runs $z \leftarrow \mathsf{A}^{\hat{F}}$ and then computes $\mathcal{P}^{\hat{F}}(z)$, we get

$$\sqrt{\Pr_{z \leftarrow \mathsf{A}^{\hat{F}}}[P(z)]} \leq \sum_{k=1}^{q+q_{\mathcal{P}}} \llbracket \neg\mathsf{Found} \wedge (|D| \leq k-1) \to \mathsf{Found} \rrbracket + \frac{q_{\mathcal{P}}}{|\mathcal{Y}||\mathcal{Y}'|}, \tag{16}$$

Where the quantum transition capacities are defined for a compressed oracle modeling $\hat{F}$, and we slightly abuse notation by writing Found for the database property on $D_{\hat{F}}$. Using a straightforward generalization of Theorem 5.17 in [9] , we get

$$\llbracket \neg\mathsf{Found} \wedge (|D| \leq k-1) \to \mathsf{Found} \rrbracket \leq \max_{\substack{x,D:\\ |D|\leq k\\ \neg\mathsf{Found}(D)}} \sqrt{10 \Pr_{u \leftarrow \mathcal{Y}}[\mathsf{Found}(D[x \mapsto u])]} \tag{17}$$

via uniform strong recognizability and the local property

$$L^{x,D} = \begin{cases} \{y \in \mathcal{Y} : \mathsf{Found}(D[x \mapsto y])\} & \text{if } |D| \leq k \\ \emptyset & \text{else.} \end{cases}$$

$$\sqrt{\Pr_{z \leftarrow \mathsf{A}^{\hat{F}}}[P(z)]} \leq \sum_{k=1}^{q+q_{\mathcal{P}}} \max_{\substack{x,D:\\ |D| \leq k \\ \neg \mathsf{Found}(D)}} \sqrt{10 \Pr_{u \leftarrow \mathcal{Y}}[\mathsf{Found}(D[x \mapsto u])]} + \frac{q_{\mathcal{P}}}{|\mathcal{Y}||\mathcal{Y}'|}$$

Taking the limit $|\mathcal{Y}'| \to \infty$ finishes the proof. □

We describe here how the proof of Theorem 4.2 in [11] implies Theorem 6, restated here fore convenience:

Theorem 6 (Variant of Theorem 5.2 from [11]). *Let $\Pi^{\mathsf{Com},G}$ be a distance-d special-sound commit-and-open identification scheme with ϕ-ary tree commitment with n_c leaves using a random oracle* Com *with output length c, splittable challenge, challenge space $\mathcal{C}^\tau$ and an additional random oracle G. Let further* A *be a* $\mathsf{UF\text{-}NMA}$*-adversary against* $\mathsf{FS}[\Pi, \mathsf{RO}]$ *making* q_{RO}*,* q_{Com} *and q_G queries to* RO*,* Com *and G respectively. Then there exists a (q_{Com}, q_G)-query QROM+ adversary* B *against the query-bounded distance-d special soundness of $\Pi^{\mathsf{Com},G}$ with respect to the special soundness extractor* Ext_d *of Π such that*

$$\begin{aligned}\mathrm{Adv}^{\mathsf{UF\text{-}NMA}}_{\mathsf{FS[IDS,RO]}}(\mathsf{A}) \leq & \Pr[\mathsf{sk}' \leftarrow \mathsf{Ext}_d \circ \mathsf{B} : (\mathsf{sk}', \mathsf{pk}) \in \mathsf{Keygen}()] \\ & + \mathrm{Adv}^{d\text{-}\mathsf{spS}}_{\mathsf{IDS,Ext}}(\mathsf{B}) + (22 n_c \log_\phi n_c + 60) q^3 2^{-c} + 20 q^2 \frac{1}{|\mathcal{C}|^{\tau-d}},\end{aligned}$$

where $q = q_{\mathsf{Com}} + q_{\mathsf{RO}}$. The runtime of B *is bounded as* $\mathsf{TIME}(\mathsf{B}) \leq \mathsf{TIME}(\mathsf{A}) + \gamma(q + q_G)^2))$*, where γ is polynomial in the input and output lengths of the random oracles.*

Proof. (Differences to the proof of [11, Theorem 5.2]) Theorem 5.2 in [11] is formulated and proven for binary tree commitments. It is easy to check that the proof also works for ϕ-ary tree commitments for any ϕ, the only change being that the logarithm in the bound needs to be taken with basis ϕ instead.

The structure of the argument in [11] is as follows. First, it is shown that except with a small probability that depends on the number of oracle queries, after an adversary A that produces a valid forgery has interacted with a compressed oracle simulation of Com and RO, using the databases to invert the tree commitment algorithm on the commitments output by A reveals an extractable set of transcripts. In the setting of statistical distance-d special soundness, a set of transcripts S being extractable is equivalent to S containing two valid transcripts with the same w and challenges of distance at least d. In summary, it is shown in [11] that except with small probability, running A with a compressed oracle simulation of Com and RO allows producing two valid transcripts with the same w and challenges of distance at least d.

We can use this intermediate result of [11] to build the (q_{Com}, q_G)-query QROM+ adversary B against the query-bounded distance-d special soundness of $\Pi^{\mathsf{Com},G}$ with respect to the special soundness extractor Ext_d. B runs A, simulating RO using a compressed oracle. It then measures the database of Com (allowed in the QROM+) to invert Com on the commitments output by A. It then constructs two valid transcripts and outputs them. Noting that the online extraction error $\varepsilon_{\mathrm{ex}}$ defined in [11] fulfils

$$\varepsilon_{\mathrm{ex}} \geq \mathrm{Adv}^{\mathsf{UF\text{-}NMA}}_{\mathsf{FS[IDS,RO]}}(\mathsf{A}) - \Pr[\mathsf{sk}' \leftarrow \mathsf{Ext}_d \circ \mathsf{B} : (\mathsf{sk}', \mathsf{pk}) \in \mathsf{Keygen}()] - \mathrm{Adv}^{d\text{-}\mathsf{spS}}_{\mathsf{IDS,Ext}}(\mathsf{B})$$

via a union bound in our setting, we get the desired bound. □

C Additional subroutines and algorithms

In Figure 3 we describe the protocol to verify via MPC evaluation points of polynomials.

```
Verify Beaver multiplication
Verify the triple $[\![s]\!], [\![q]\!], [\![pf]\!]$ with sacrificed triple $[\![a]\!], [\![b]\!], [\![c]\!]$and mask point $\varepsilon$
Set $[\![\alpha]\!] = \epsilon \cdot [\![q]\!] + [\![a]\!]$ and set $[\![\beta]\!] = [\![s]\!] + [\![b]\!]$
Parties open $[\![\alpha]\!]$ and $[\![\beta]\!]$ on bulletin board and sum to construct $\alpha$ and $\beta$
Parties set $[\![v]\!] = \epsilon \cdot [\![pf]\!] - [\![c]\!] + \alpha \cdot [\![b]\!] + \beta \cdot [\![a]\!] - \alpha \cdot \beta$
Parties open $[\![v]\!]$ to obtain $v$
**return** $(v = 0)$
```

Fig. 3: Verify Beaver multiplication on evaluation points.

Algorithm 6 Execute Π on a full set of parties

```
**Input:** $\left\{[\mathbf{x}_A]_i, [Q]_i, [P]_i, [a]_i, [b]_i, [c]_i\right\}_{i=0}^{N-1}, \{r_l, \varepsilon_l\}_{l=0}^{t-1}$.
**Output:** $\{[\alpha], [\beta], [v]\}_{i=0}^{N-1}$
  Parties locally set $[\mathbf{x}_B]_i = \mathbf{y} - \mathbf{H}'[\mathbf{x}_A]_i$.
  Parties locally compute $[S]_i$ via interpolation of $[\mathbf{x}]_i = ([\mathbf{x}_A]_i \mid [\mathbf{x}_B])_i$.

  // Compute $[\alpha], [\beta], [v]$ coordinate-wise:
  **for** $l \in \{0, \dots, t-1\}$ **do**
      Parties locally compute $[S(r_l)]_i, [Q(r_l)]_i, [P(r_l)]_i$.
      Parties locally set $[\alpha_l]_i = \varepsilon_l [Q(r_l)]_i + [a_l]_i$.
      Parties locally set $[\beta_l]_i = [S(r_l)]_i + [b_l]_i$.
      Parties open $[\alpha_l]_i$ and $[\beta_l]_i$ and recombine to get $\alpha_l, \beta_l$.
      Parties locally set
          $$[v_l]_i = -[c_l]_i + \langle \varepsilon_l F(r_l) \cdot [P(r_l)]_i \rangle + \langle \alpha_l, [b_l]_i \rangle + \langle \beta_l, [a_l]_i \rangle - \langle \alpha_l, \beta_l \rangle.$$
  **return** $\{[\alpha], [\beta], [v]\}_{i=0}^{N-1}$.
```

Algorithm 7 Verify a partition of parties

```
**Input:** Index $\mathsf{c}$ and communication $\boldsymbol{\alpha}_{\mathsf{c}}, \boldsymbol{\beta}_{\mathsf{c}}$ of the hidden leaf party. Secret-shares
  $\left\{[\mathbf{x}_A]_i, [Q]_i, [P]_i, [a]_i, [b]_i, [c]_i\right\}_{i \neq \mathsf{c}; i=0}^{N-1}$, and challenge points $\{r_l, \varepsilon_l\}_{l=0}^{t-1}$. The Party $i'$ that contains the
  hidden leaf party $\mathsf{c}$ (hereafter partially-disclosed Party) uses as shares the partial aggregation from its disclosed
  leaf parties.

**Output:** $[\alpha], [\beta], [v]$
  Parties locally set $[\mathbf{x}_B] = \mathbf{y} - \mathbf{H}'[\mathbf{x}_A]$.
  Parties locally compute $[S]$ via interpolation of $[\mathbf{x}] = ([\mathbf{x}_A] \mid [\mathbf{x}_B])$.
  **for** $l \in [t]$ **do**                                   ▷ Compute $[\alpha], [\beta], [v]$ coordinate-wise.
      Parties locally evaluate $[S(r_l)], [Q(r_l)], [P(r_l)]$.
      Parties set $[\alpha_l] = \varepsilon_l [Q(r_l)] + [a_l]$. and $[\beta_l] = [S(r_l)] + [b_l]$.
      The Partially-disclosed Party adds $\mathsf{c}$ communications to $[\![\alpha_l]\!]$ and $[\![\beta_l]\!]$.
      Parties open $[\alpha_l]$ and $[\beta_l]$ to get $\alpha_l, \beta_l$.
      All Parties but the partially-disclosed one locally set
          $$[v_l] = -[c_l] + \langle \varepsilon_l F(r_l) \cdot [P(r_l)] \rangle + \langle \alpha_l, [b_l] \rangle + \langle \beta_l, [a_l] \rangle - \langle \alpha_l, \beta_l \rangle.$$
      The local share $[v_l]$ of the partially-disclosed Party is set so that $v_l = 0$
```

D Proof of Theorem 1

Here we give the detailed proof for Theorem 1:

Theorem 1 (Honest-Verifier Zero Knowledge (HVZK)). *The algorithm* Sim_Π *shown in Algorithm 3 is an HVZK simulator for* Π *such that for any quantum algorithm* A *in distinguishing* Trans_Π *from* Sim_Π *making at most* q_{zk} *queries to its oracle there exist algorithms* B*– distinguishing the outputs of* $\mathsf{TreePRG}$ *from random – and* C*– breaking the hiding property of* Com*– which fulfill*

$$\begin{aligned}\mathrm{Adv}^{\mathsf{hvzk}}_{\Pi,\mathsf{Sim}}(\mathsf{A}) &:= \left|\Pr[1 \leftarrow \mathsf{A}^{\mathsf{Sim}_\Pi}] - \Pr[1 \leftarrow \mathsf{A}^{\mathsf{Trans}_\Pi}]\right| \\ &\leq q_{\mathsf{zk}}\tau(\mathrm{Adv}^{\mathsf{hide}}_{\mathsf{Com}}(\mathsf{C}) + \mathrm{Adv}^{\mathsf{ror}}_{\mathsf{TreePRG}}(\mathsf{B})),\end{aligned}$$

where B *and* C *run in time* $\mathsf{TIME}(\mathsf{B}) = \mathsf{TIME}(\mathsf{C}) = \mathsf{TIME}(\mathsf{A}) + \mathsf{TIME}(\mathsf{Trans})$ *respectively.*

Proof. Consider the following sequence of games as given in the proof sketch. We start with Trans, i.e., the honest execution of the protocol, in GAME_0. In the first hop, we switch the order of operations and sample the challenges first. This defines GAME_1. In GAME_2, we replace the seed seed_{i^*} and the commitment pseudorandomness ρ_{i^*} for the commitments that remain unopened by truly random bits. To be consistent with $\mathsf{TreePRG}$, we also sample a random sibling path path which we use to derive the values for the opened commitments. Next, we replace the state of the unopened parties by truly random bits in GAME_3. In GAME_4 we sample that state uniformly at random. To preserve consistency of the communications, we compute the communications of all opened parties using the original algorithm. Then we compute the communication of the unopened parties to agree with these.

To prove the claimed bound, we bound the distinguishing advantage of A

$$\delta_i := |\Pr[1 \leftarrow \mathsf{GAME_i}(\mathsf{A})] - \Pr[1 \leftarrow \mathsf{GAME_{i-1}}(\mathsf{A})]|$$

between any two consecutive games. First, note that $\delta_1 = 0$ as the change from GAME_0 to GAME_1 does not change the output distribution of the oracle at all.

To bound δ_2, we have to apply a two-stage hybrid argument, first over the oracle calls, then over the parallel executions in one call. The outer hybrid games $\mathsf{Hyb_k}$ sample the commitment randomness and seed for the unopened commitments in the first k oracle calls at random – as well as the path – and use pseudorandom bits for the remaining calls. Consequently, we have that $\mathsf{Hyb}_0 = \mathsf{GAME}_1$ and $\mathsf{Hyb}_{\mathsf{q_{zk}}} = \mathsf{GAME}_2$. It also follows that there has to exist at least one k such that $\overline{\delta}_k := |\Pr[1 \leftarrow \mathsf{Hyb_k}(\mathsf{A})] - \Pr[1 \leftarrow \mathsf{Hyb_{k-1}}(\mathsf{A})]| \geq q_{\mathsf{zk}}^{-1}\delta_2$.

To further bound $\overline{\delta}_k$ we define inner hybrids $\overline{\mathsf{Hyb}}_{\mathsf{k},\ell}$ which act like $\mathsf{Hyb_k}$ except when answering the $k+1$-th query. For that query, the commitment randomness and seed for the unopened commitment and the path in the first ℓ repetitions of the basic IDS are sampled uniformly at random while the remaining repetitions make use of the pseudorandom values. Again, $\overline{\mathsf{Hyb}}_{\mathsf{k},0} = \mathsf{Hyb_k}$, and $\overline{\mathsf{Hyb}}_{\mathsf{k},\tau} = \mathsf{Hyb_{k+1}}$. Given that there are τ repetitions, it follows that for any k, there has to be at least one ℓ such that $\left|\Pr[1 \leftarrow \overline{\mathsf{Hyb}}_{\mathsf{k},\ell}(\mathsf{A})] - \Pr[1 \leftarrow \overline{\mathsf{Hyb}}_{\mathsf{k},\ell-1}(\mathsf{A})]\right| \geq \tau^{-1}\overline{\delta}_k$.

Now note that the difference of any two consecutive such hybrids is upper bounded by the distinguishing advantage against the used $\mathsf{TreePRG}$. This can be shown by a reduction B that given a bit string r^* and a sibling path path^* which are either a leaf and its sibling path as produced by $\mathsf{TreePRG}$ (for an unknown fresh random seed) or random bit strings. B then parses $r^* = (\mathsf{seed}_{i^*_\ell}, \rho_{i^*_\ell})$ and uses $\rho_{i^*_\ell}$ as commitment randomness for the ℓ-th unopened commitment in $\overline{\mathsf{Hyb}}_{\mathsf{k},\ell}$ and the sibling path as the opening information in z. When A makes a decision, B simply forwards that decision. If the strings are random this perfectly simulates

$\overline{\mathsf{Hyb}}_{\mathsf{k},\ell}$, if the values were computed using $\mathsf{TreePRG}$, this perfectly simulates $\overline{\mathsf{Hyb}}_{\mathsf{k},\ell-1}$. Hence, $\mathrm{Adv}^{\mathsf{ror}}_{\mathsf{TreePRG}}(\mathsf{B}) = \left|\Pr[1 \leftarrow \overline{\mathsf{Hyb}}_{\mathsf{k},\ell}(\mathsf{A})] - \Pr[1 \leftarrow \overline{\mathsf{Hyb}}_{\mathsf{k},\ell-1}(\mathsf{A})]\right| \geq \tau^{-1}\overline{\delta}_k \geq \tau^{-1}q_{\mathsf{zk}}^{-1}\delta_2$ and we get

$$\delta_2 \leq \tau q_{\mathsf{zk}} \cdot \mathrm{Adv}^{\mathsf{ror}}_{\mathsf{TreePRG}}(\mathsf{B})\,.$$

Upon inspection B runs A and answers q_{zk} transcript queries, so $\mathsf{TIME}(\mathsf{B}) = \mathsf{TIME}(\mathsf{A}) + q_{\mathsf{zk}} \cdot \mathsf{TIME}(\mathsf{Trans})$. Especially, the number of queries that B makes to $\mathsf{TreePRG}$ in addition to those made by A are bounded by τq_{zk}.

To analyze δ_3, we follow exactly the same hybrid structure. Only this time, we replace the value committed to in the unopened commitments by a random bit string. This means, the atomic code snipped that distinguishes two consecutive inner hybrids is distinguishing which message gets committed to, the real one or a freshly sampled random one. Hence, the difference between these two games is upper bounded by the hiding property of the commitment. More precisely, following the approach above, we can define an algorithm C that follows $\overline{\mathsf{Hyb}'}_{\mathsf{k},\ell}$ (which always uses random values as commitment randomness and commits to a random message for the first ℓ repetitions in the $k+1$-th query and to the actual message otherwise) but for the ℓ-th repetition of the basic IDS, in the $k+1$-th query, C samples a random bit string and sends it together with the message it would commit to in an honest execution to its oracle from the hiding game. It will receive back a commitment that commits to one of the two. In the end, C outputs whatever A outputs. When the commitment contained the real message, this perfectly simulates $\overline{\mathsf{Hyb}}_{\mathsf{k},\ell-1}$, else $\overline{\mathsf{Hyb}}_{\mathsf{k},\ell}$. Consequently, we obtain

$$\delta_3 \leq \tau q_{\mathsf{zk}} \cdot \mathrm{Adv}^{\mathsf{hide}}_{\mathsf{Com}}(\mathsf{C})\,.$$

Moreover, C does essentially the same as B, so $\mathsf{TIME}(\mathsf{C}) = \mathsf{TIME}(\mathsf{B})$.

Finally, the transition from GAME_3 to GAME_4 does not change the output distribution of the games. Hence, $\delta_4 = 0$. The final bound is then obtained applying the triangle inequality. □