UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Computer Science Curriculum

Hans Kristjan Veri

Always Two, There Are: Towards Two-Party SDitH

Master's Thesis (30 ECTS)

Supervisor: Toomas Krips, PhD

Tartu 2025

Always Two, There Are: Towards Two-Party SDitH

Abstract: The rise of quantum computing threatens to break many of the cryptographic systems that secure today's digital world. In response, researchers are developing new tools designed to remain secure in a post-quantum future. Most of the promising candidates for post-quantum digital signatures rely on security assumptions based on lattices or properties of hash functions.

Another promising approach transforms secure multi-party computation protocols into zero-knowledge proofs, which are then turned into digital signatures. This technique, known as multi-party computation in-the-head (MPCitH), offers strong security properties and flexibility for distributed applications.

This thesis investigates whether MPCitH digital signatures can be efficiently adapted for use by two cooperating parties to jointly produce a signature. Here we show how to construct two-party signatures based on syndrome decoding in-the-head (SDitH) signatures. We propose a provably secure scheme that achieves the smallest known communication overhead among two-party MPCitH signatures, while resulting in a signature size approximately double that of a single-prover variant. This result provides a new data point in the design space of multi-party MPCitH signatures and post-quantum digital signatures in general.

Keywords:
Cryptography, MPC-in-the-Head, Zero-Knowledge, Multi-Prover Zero-Knowledge, Signatures

CERCS: P170. Computer science, numerical analysis, systems, control.

Alati kaks : Kahe osapoolega SDitH digiallkirjad

Lühikokkuvõte:

Kvantarvutite areng ähvardab murda kogu avaliku võtme krüptoraafiat, mis kaitseb inimeste privaatsust digitaalses maailmas. Vastuseks sellele töötavad teadlased välja post-kvant avaliku võtme krüptograafiat, mis on turvaline ka kvantrünnakute vastu. Enamik paljulubavaid post-kvant digitaalallkirju põhinevad turvaeeldustel, mis on seotud võredega või räsifunktsioonide omadustega.

Teine paljulubav lähenemine muudab turvalised ühisarvutuse protokollid nullteadmustõestusteks, mis teisendatakse seejärel teisendada digiallkirjadeks. Seda tehnikat tuntakse nimega "peas toimuv turvaline ühisarvutus"(MPCitH).

Käesolev magistritöö uurib, kas MPCitH digitaalallkirju saab tõhusalt kohandada kahe koostööd tegeva osapoole jaoks, et luua ühine allkiri. Näitame, kuidas koostada kahe osapoole allkirju, mis põhinevad kodeerimisteooriast tuntud keerukuseeldustel ning konkreetselt SDitH allkirjadel. Pakume välja tõestatavalt turvalise skeemi, mis saavutab väikseima teadaoleva suhtlusmahu kahe osapoole MPCitH allkirjade seas ning allkirja suurus on umbes kahekordne võrreldes ühe osapoolega variandiga. See tulemus tekitab uue andmepunkti mitme osapoole MPCitH allkirjade ja ka post-kvant digiallkirjade disainiruumis.

Võtmesõnad:
Krüptograafia, peas toimuv turvaline ühisarvutus, nullteadmustõestused, mitme tõestajaga nullteadmustõestused, digiallkirjad.

CERCS: P170. Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine

Contents

1 Introduction

With the looming threat of quantum computers on the horizon, threatening to break the most practical public key cryptography schemes like RSA and elliptic curve based schemes, the American standardization organization NIST announced a call for post-quantum signature schemes in 2017. This attracted a diverse range of proposed schemes from researchers all around the world [Nat17]. While some of the most promising schemes, such as Falcon [FHK+25] and Crystals-Dilithium [DLL+17], are based on cryptographic assumptions related to hard problems on lattices, others were solely based on properties of hash functions [ABB+25]. They boasted the lowest computational costs along with the smallest signatures and public keys, and were promptly advanced to the next rounds of the standardization process.

Another previously less-developed family of signatures greatly leaped forward thanks to this call for proposals - signatures based on secure multi-party computation in-the-head (MPCitH). The signatures in this family are based on diverse assumptions ranging from one-wayness of quadratic multivariate polynomial evaluation [Beu21] to those related to hard problems in coding theory [FJR22]. Despite their different flavors, all MPCitH schemes include transforming a secure multi-party computation protocol into a zero-knowledge protocol and then transforming that into a non-interactive signature scheme via the Fiat-Shamir transformation. While not competitive with the best lattice-based schemes, MPCitH signatures offer a lot in terms of robust security assumptions and concrete efficiency.

In addition to standard digital signatures, many real-world applications require schemes with advanced properties. This thesis is focused on threshold signatures which allow an honest majority of a defined set of parties to sign a message. In particular, we focus on two-party signatures. This setting naturally arises, for example, in the context of an backdoored mobile device which cannot be trusted to store a full cryptographic key.

The fact that the underlying computations are multi-party friendly, begs the question - are MPCitH signature schemes easily convertible to multi-party threshold signatures?

The answer is a resounding "not easily". A recent impossibility result found that MPCitH signatures converted to a threshold variant either grow linearly with the number of signers or make non black-box use of hash functions or pseudo-random generators [DKR24]. The latter introduces a prohibitively large communication overhead.

My Contribution. This thesis designs a concretely effective two-party signature scheme from the hypercube syndrome decoding in-the-head (SDitH) approach introduced by Aguilar-Melchor *et al.* [AMGH+23] and proves its security in the random oracle model. I came up with the algorithms and proved their security properties. More specifically, as expected, I did not find any way to circumvent the aforementioned impossibility result, so the two-party signatures proposed are approximately twice as large as ones created by a single prover, and in signature size, my approach is asymptotically worse than that of Carozza and Couteau [CC24]. However, I was able to design a novel two-party scheme, prove its security, and make optimizations to the MPC protocol execution part of the MPCitH proof threshold execution.

The key merit of this two-party MPCitH signature is the optimized communication overhead - in addition to the signature itself, the signers just need to send a constant number of field elements per execution of the protocol. To my knowledge, this is a novel result.

The thesis is structured into 3 main sections. The first section introduces the mathematical and cryptographic preliminaries necessary for working with syndrome decoding in-the-head signatures. These include, but are not limited to, lemmata from probability

theory, an introduction to secure multi-party computation and defining the hypercube approach for reducing communication costs.

The second section proposes two-prover zero-knowledge proof and explains the design decisions. Furthermore, this section introduces the definitions for the desired properties of zero-knowledge proofs (knowledge soundness, honest-verifier zero-knowledge), which are generalized to the multi-prover setting. Then, simulation-based proofs are given for these properties.

The third section finally transforms the two-prover zero-knowledge proof of Section 2 to a signature scheme and proves its security in the random oracle model.

2 Preliminaries

2.1 Cryptographic Definitions and Lemmas

We will define the notation and parameters that are used throughout this thesis. The notation is similar to the one used by Aguilar-Melchor *et al.* [AMGH+23] but adapted to our two-prover setting. Denote by $[k]$ the set $\{1, \ldots, k\} \subset \mathbb{N}$. Bold lowercase letters denote vectors. Capital letters denote polynomials and matrices. For example, $[\![\boldsymbol{S}]\!]_i^p$ is the additive of a vector of coefficients belonging to the i^{th} in-the-head party, held by the p^{th} prover. A tuple subscript, e.g $[\![\boldsymbol{S}]\!]_{k,i}^p$ indicates the share of a the i^{th} main party in the k^{th} dimension of the hypercube. A tuple superscript indicates that we want to specify the repetition of the protocol. For example, $[\![\boldsymbol{S}]\!]_i^{e,p}$ is the i^{th} leaf party share of $\boldsymbol{S}$ in the e^{th} repetition of the protocol, held by the p^{th} prover. Denote by $\mathrm{wt}(\boldsymbol{x})$ the number of nonzero coordinates of a vector. All relevant notation is summarized in Table 1.

Definition 1 (Negligible Functions)**.** *We say that a function* $\mu\colon \mathbb{N} \to \mathbb{R}$ *is* negligible *if for all real-valued polynomials* $p(X)$*, there exists* $N \in \mathbb{N}$ *such that for all* $n > N$

$$\mu(n) < p(n).$$

Definition 2 (Indistinguishability)**.** *We say that two distributions* X, Y *are* (t, ϵ) - indistinguishable *if for all probabilistic polynomial-time algorithms* $\mathcal{A}$ *running in time* t

$$\Pr[\mathcal{A}(X) = 1] - \Pr[\mathcal{A}(Y) = 1\ |] \leq \epsilon(\lambda).$$

X, Y *are computationally indistinguishable when* $t \in poly(\lambda)$ *and* ϵ *is a negligible function of* λ*.* X, Y *are statistically indistinguishable when and* ϵ *is a negligible function of* λ *for a computationally unbounded* t*.*

Definition 3 (Pseudorandom generation (PRG))**.** *We say that a function* $\mathsf{PRG}\colon \{0,1\}^* \to \{0,1\}^*$ *is a pseudorandom generator (PRG) if*

1. PRG *is expansive, meaning for all* $\lambda \in \mathbb{N}$*, for all* $x \in \{0,1\}^\lambda\colon \mathsf{PRG}(x) \in \{0,1\}^{l(\lambda)}$ *where* $l(\lambda) > \lambda$*.*

2. PRG *is pseudorandom, which means that the distributions* $\{G(x) \mid x \in \{0,1\}^\lambda\}$ *and* $\{y \mid y \xleftarrow{\$} \{0,1\}^{l(\lambda)}\}$ *are* (t, ϵ) *indistinguishable.*

Definition 4 (Binding)**.** *A commitment scheme is said to be perfectly binding if, for any probabilistic polynomial-time (in the security parameter* κ*) algorithm* $\mathcal{A}$*, the probability of finding* C, D, D' *such that* $\mathsf{open}(C, D) = M$*,* $\mathsf{open}(C, D') = M'$*, with* $M \neq M'$*, is zero. It is computationally binding if this probability is negligible as a function of* κ*.*

Definition 5 (Hiding)**.** *A commitment scheme is perfectly, statistically, or computationally (respectively) hiding if, for any two messages* M, M'*, the distributions*

$$\{C : (C, D) \leftarrow \mathsf{com}(M)\}_{\kappa \in \mathbb{N}}$$

and

$$\{C : (C, D) \leftarrow \mathsf{com}(M')\}_{\kappa \in \mathbb{N}}$$

are perfectly, statistically, or computationally indistinguishable.

Table 1. Descriptions of the notation and parameters used in our scheme.

Indices:	
i	Index of a leaf party, in $[N^D]$.
i^*	Index of challenge party, which remains hidden.
$\boldsymbol{i}^* = (i_1, \ldots, i_D)$	Representation of i on dimension D hypercube with side N.
(k, j)	Index of a *main party* in $[D] \times [N]$, where k indexes the hypercube dimension.
$\mathbb{F}_{\text{poly}}$	Field extension of $\mathbb{F}_{\text{SD}}$ from which S, Q, P, F coefficients are drawn.
$\mathbb{F}_{\text{points}}$	Field from which $\alpha, \beta, \mathbf{v}, r, \epsilon$ are drawn.
Multi-Prover ZKP	
p	Index of real prover, $p \in \{1, 2\}$.
$\mathcal{P}_p$	The p^{th} prover.
$\mathcal{V}$	The verifier.
$\tilde{\mathcal{P}} = (\mathcal{P}_1, \mathcal{P}_2)$	Joint algorithm of two maliciously collaborating provers.
$[\![X]\!]^p$	Additive share of value X held by prover p.
Multi-Party Computation:	
Main party	Party using an aggregated share and for which we actually run the MPC protocol.
a, b, c	Elements of the Beaver triplet such that $a \cdot b = c$.
$\alpha, \beta, \mathbf{v}$	Communications output, drawn from $\mathbb{F}_{\text{points}}$.
$[\![X]\!]_i^p$	i^{th} leaf party share of X by real party p.
t	Number of random evaluation points.
p_f	False positive probability.
Syndrome Decoding:	
S, Q, P, F	Polynomials in $\mathbb{F}_{\text{poly}}$ which encode the syndrome decoding proof.
aux	Uncompressed secret shares of leaf party N^D, $[\![S]\!] \Vert [\![Q]\!] \Vert [\![P]\!] \Vert [\![\mathbf{a}]\!] \Vert [\![\mathbf{b}]\!] \Vert [\![c]\!]$.
$(\mathsf{state}_i, \rho_i)$	State and commitment randomness of a leaf party. For $i \neq N^D$, $state_i$ is a pseudorandom seed, and $state_{N^D} = (\text{seed}_{N^D} \Vert \text{aux})$
m	Code length.
k	Vector dimension.
ω	Hamming weight bound on weight of witness vector $\boldsymbol{x}$.
Signature Parameters:	
λ	The security parameter.
ϵ	The soundness parameter.
D	The dimension of the hypercube.
N	The number of secret shares.
τ	The number of repetitions.

It is impossible for a commitment scheme to be both perfectly binding and perfectly hiding at the same time. To see why, suppose first that the scheme is perfectly binding. Then, once a commitment $\mathsf{com}^k(\mathsf{open}, x)$ is published, no other pair (open, x) can produce the same commitment value. However, a computationally unbounded adversary could exhaustively test pairs (open', x') until discovering the unique pair yielding the correct opening. As a result, the scheme cannot simultaneously achieve perfect hiding under these conditions.

Lemma 1 (Splitting Lemma [PS00]). *Let $A \subset X \times Y$, and $\Pr[(x, y) \in A] \geq \kappa$. Then, for*

any $\alpha \in [0,1)$*, let*

$$B = \{(x,y) \in X \times Y \mid \Pr_{y \in Y}[x,y'] \in A \geq (1-\alpha)\cdot\kappa\}.$$

Then $\Pr[(x,y) \in B] \geq \alpha \cdot \kappa$ *and* $\Pr[(x,y) \in B \mid (x,y) \in A] \geq \alpha$*.*

Lemma 2 (Schwartz-Zippel, multi-point variant, [FJR22] Lemma 3)**.** *Let* $R \in \mathbb{F}[X]$ *be of degree* $d > 0$*, for any* $\mathbb{S} \subset \mathbb{F}$ *and any* $t \geq 1$*,*

$$\Pr_{r_1,\ldots,r_t \leftarrow \mathbb{S}}[R(r_1) = \ldots = R(r_t) = 0 \mid \{r_i\} \textit{ are distinct }] \leq \frac{\binom{d}{t}}{\binom{|\mathbb{S}|}{t}}.$$

Proof. There are $\binom{|\mathbb{S}|}{t}$ possible ways to sample t distinct points from $\mathbb{S}$. The studied event occurs when all t distinct points are roots of R. Since R has at most d roots, this event occurs at most $\binom{d}{t}$ times and with probability at most $\frac{\binom{d}{t}}{\binom{|\mathbb{S}|}{t}}$.

□

Lemma 3 (Schwartz-Zippel, multi-point variant 2, [FJR22] Lemma 4)**.** *Let* $R \in \mathbb{F}[X]$ *be of degree* $d > 0$*, for any* $\mathbb{S} \subset \mathbb{F}$ *and any* $t, \ell \geq 1$*,*

$$\Pr_{r_1,\ldots,r_t \leftarrow \mathbb{S}}[|\{i\colon R(r_i) = 0\}| = l \mid \{r_i\} \textit{ are distinct }] \leq \frac{\max\left\{\binom{i}{\ell}\cdot\binom{|\mathbb{S}|-i}{t-\ell}\right\}}{\binom{|\mathbb{S}|}{t}}.$$

If $t \cdot d \leq \ell \cdot (|\mathbb{S}| - 1)$*, we have*

$$\Pr_{r_1,\ldots,r_t \leftarrow \mathbb{S}}[|\{i\colon R(r_i) = 0\}| = l \mid \{r_i\} \textit{ are distinct }] \leq \frac{\binom{d}{\ell}\cdot\binom{|\mathbb{S}|-d}{t-\ell}}{\binom{|\mathbb{S}|}{t}}.$$

2.2 Multi-Party Computation With Additive Shares

A commonly used approach in secure multiparty computation (MPC) protocols is *additive secret sharing*, where a secret value is additively split into multiple random shares distributed among a set of parties. Each individual share reveals no information about the underlying secret, yet the original value can be recovered by summing all the shares.

Formally, let $\mathbb{F}$ be a finite field, and let a secret value $x \in \mathbb{F}$ be shared among n parties. To create an additive sharing of x, we sample uniformly random shares $x_1, \ldots, x_{n-1} \in \mathbb{F}$, and define $x_n = x - \sum_{i=1}^{n-1} x_i$. A key observation is that holding all-but-one of those shares doesn't leak any information to the holder about the secret itself. These shares are commonly denoted as $[\![x]\!]_1 \ldots [\![x]\!]_n$. The secret can be *reconstructed* by adding the shares:

$$x = \sum_{i=1}^{n} [\![x]\!]_i.$$

Operations on Additive Shares

Additive secret sharing is linear, which makes certain operations on shared values straightforward and efficient:

- **Adding a Public Constant:** To add a public constant $c \in \mathbb{F}$ to a secret x, one party (commonly party 1) adds c to their share:

  $$[\![z]\!]_1 = [\![x]\!]_1 + c, \quad \text{and} \quad [\![z]\!]_i = [\![x]\!]_i \text{ for } i \neq 1.$$

  The resulting shares $[\![z]\!]$ sum up to $x + c$.

- **Adding Secret Values:** Given shares of two secrets x and y, namely $[\![x]\!]$ and $[\![y]\!]$, parties can locally compute shares of $z = x + y$ by setting:

$$[\![z]\!] = [\![x]\!] + [\![y]\!].$$

- **Multiplying by a Constant:** To multiply a shared secret x by a public constant $c \in \mathbb{F}$, each party multiplies their share locally:

$$[\![z]\!] = c \cdot [\![x]\!]_i.$$

  The resulting shares $[\![z]\!]$ add up to $c \cdot x$.

- **Multiplying Two Secrets**

  Multiplication of two shared secrets x and y requires communication between the parties. A standard approach uses *Beaver triples* [Bea92]. A Beaver triple is a tuple of shared values $([\![a]\!], [\![b]\!], [\![c]\!])$ where a, b are random and $c = a \cdot b$. Parties in possession of additive shares of a Beaver triple can "sacrifice" it to securely compute the product $[\![z]\!] = [\![xy]\!]$ as follows:

  1. Each party locally computes:

$$[\![d]\!] = [\![x]\!] - [\![a]\!], \quad [\![e]\!] = [\![y]\!] - [\![b]\!].$$

  2. Parties open $d = \sum_i [\![d]\!]_i$ and $e = \sum_i [\![e]\!]_i$ by broadcasting their values.
  3. Each party computes:

$$[\![z]\!] = [\![c]\!] + d \cdot [\![b]\!] + e \cdot [\![a]\!] + d \cdot e.$$

  The resulting shares $[\![z]\!]$ add up to $x \cdot y$.

2.3 Multi-Party Computation in-the-Head

The Multi-Party Computation in-the-Head (MPCitH) paradigm was originally introduced in the works of Ishai, Kushilevitz, and Ostrovsky [IKOS07] and laid the foundation for building zero-knowledge from secure MPC protocols. Assume that we have an MPC protocol in which N parties $\mathcal{P}_1, \ldots, \mathcal{P}_N$ securely and correctly evaluate a function f on a secret input x so that:

- The secret x is additively secret shared between all parties with each party $\mathcal{P}_i$ holding $[\![x]\!]_i$.
- The function f outputs 0 or 1. After the protocol, party $\mathcal{P}_i$ holds $[\![f(x)]\!]_i$.
- Seeing the views of $N - 1$ in-the-head parties does not reveal anything about the secret x.

A protocol with these properties can be transformed into a zero-knowledge proof that a single prover $\mathcal{P}$ knows x such that $f(x) = 1$. To this end, the prover proceeds as follows:

- Distributes additive shares of x to N imaginary parties so that the i^{th} in-the-head party holds $[\![x]\!]_i$.
- Simulates the protocol between the in-the-head parties.

- Sends commitments of each party's view to the verifier. That includes its input share, all sent and received messages, and its secret random tape.
- Send all shares $[\![f(x)]\!]_i$ to the verifier.

The verifier checks the proof as follows:

- Randomly chooses $N-1$ in-the-head parties and requests that the prover sends their views.
- Replay the MPC protocol among the $N-1$ parties and verify that the resulting views and commitments are consistent with those sent by the prover.
- Accept if views and commitments are consistent and shares of $[\![f(x)]\!]$ reconstruct to 1.

This zero-knowledge proof achieves a $\frac{1}{N}$ soundness error. This corresponds to the probability that the prover cheated in executing the protocol in the one in-the-head party which the verifier doesn't ask to reveal.

The protocol for syndrome decoding in-the-head [FJR22] uses an additional round of communication. That is, the verifier provides some challenge randomness to evaluate f after the prover commits to the secret sharing of $[\![x]\!]$, but before the protocol is simulated.

2.4 Coding Theory and the Syndrome Decoding Problem

A linear code C of length n and dimension k is defined as a k-dimensional subspace of $\mathbb{F}_q^n$ [EB21]. This code can be described as either the image of a generator matrix $G \in \mathbb{F}_q^{k\times n}$ or, equivalently, as the kernel of a parity check matrix $H \in \mathbb{F}_q^{(n-k)\times n}$. In this thesis, the parity-check matrix formulation is relevant.

Since any codeword $c \in C$ satisfies the equation $Hc = 0$, decoding a potentially corrupted codeword $z = c + x$, where x represents the error vector, leads to the following relation:

$$Hz = H(c + x) = Hx =: y.$$

The vector y is referred to as the *syndrome* of z. The problem of determining the error vector x given the parity-check matrix H and the syndrome y is known as the *syndrome decoding problem.*

Definition 6 (Syndrome Decoding Problem, [EB21])**.** *Let $H \in \mathbb{F}_q^{(n-k)\times n}$ be the parity-check matrix of a randomly chosen linear code, let $y \in \mathbb{F}_q^{n-k}$ be a syndrome, and let $\omega \in [n]$ denote an integer weight parameter. The* syndrome decoding problem *is to find a vector $x \in \mathbb{F}_q^n$ such that* $\mathrm{wt}(x) = \omega$ *and*

$$He = s.$$

The challenge can therefore be stated as follows: From (H, y), find x. This work uses a relaxed version of the syndrome decoding problem - x has to satisfy $\mathrm{wt}(e) \leq \omega$*. Parameters k, n, ω can be chosen such that the solution x is, with overwhelming probability, unique and requires an exponential number of computations to find.*

The syndrome decoding problem is known to belong to the class of NP-complete problems [BMvT78]. This inspired the McEliece cryptosystem - the first code-based post-quantum public-key encryption scheme [McE78] based on decoding random Goppa codes. It is still considered secure.

The pioneering work of Stern [Ste94] introduced the first zero-knowledge proof of knowledge of a solution to a syndrome decoding instance, which reported a soundness error of $\frac{2}{3}$. By repeating the protocol τ times, the protocol can be made to have any desired soundness error $\left(\frac{2}{3}\right)^{\tau}$. Subsequent works, such as the papers which inspired this thesis [FJR22], [AMGH+23], greatly improved the soundness error to $\frac{1}{N}$ for arbitrary N. They proposed zero-knowledge proofs of a solution to a syndrome decoding instance, which are based in the MPC-in-the-head paradigm.

2.5 A Zero-Knowledge Protocol for Syndrome Decoding

In the following, we will detail the main ideas behing the syndrome decoding in-the-head zero-knowledge proof, which was first introduced by Feneuil *et al.* [FJR22] and later improved by Aguilar-Melchor *et al.* [AMGH+23].

Consider an instance (H, y) of the syndrome decoding problem, where x denotes a valid solution to this instance. Let $\mathbb{F}_{SD}$ be the finite field over which the problem is defined. Without loss of generality, we assume the parity-check matrix H is in standard form, i.e., $H = (H_0 \mid I_{m-k})$ for some matrix $H_0 \in \mathbb{F}_{SD}^{(m-k)\times k}$. Every full-rank H can be transformed into this form by applying Gaussian elimination. It follows that, the solution vector x can be decomposed as $(x_A \mid x_B)$ such that the linear relation holds:

$$y = H_0 x_A + x_B. \tag{1}$$

In the following, we first construct a multi-party computation (MPC) protocol that operates on a shared value of x_A, reconstructs x using the relation above, and verifies that x has Hamming weight at most w.

Let $\mathbb{F}_{poly}$ be a field extension of $\mathbb{F}_{SD}$ such that $|\mathbb{F}_{poly}| \geq m$, where m is the length of the secret vector $x \in \mathbb{F}_{SD}^m$. Let $\varphi : \mathbb{F}_{SD} \to \mathbb{F}_{poly}$ denote the canonical embedding of $\mathbb{F}_{SD}$ into $\mathbb{F}_{poly}$, and let γ be a bijection between $\{1, \ldots, |\mathbb{F}_{poly}|\}$ and $\mathbb{F}_{poly}$. For notational convenience, denote $\gamma(i)$ as γ_i.

The protocol must verify both that $y = Hx$ and that the Hamming weight of x satisfies $\mathrm{wt}(x)\omega$. As introduced earlier, the MPC protocol receives shares of x_A and reconstructs shares of x using equation (1). The equation $y = Hx$ is automatically checked by construction, and it remains to verify the weight constraint.

To establish that $\mathrm{wt}(x) \leq \omega$, the prover constructs the following polynomials:

- An interpolating polynomial $S \in \mathbb{F}_{poly}[X]$ satisfying

  $$\forall i \in [m],\ S(\gamma_i) = \varphi(x_i)$$

  with $\deg S \leq m-1$. This polynomial is unique and can be computed via interpolation.

- A monic polynomial $Q \in \mathbb{F}_{poly}[X]$ defined by

  $$Q(X) = \prod_{i \in E} (X - \gamma_i)$$

  for some subset $E \subset [m]$ of size w that contains the indices of all nonzero elements of x, e.g $\{i \in [m] \mid x_i \neq 0\} \subset E$. This ensures $\deg Q = w$.

- A polynomial $P \in \mathbb{F}_{poly}[X]$ defined by

  $$P = \frac{S \cdot Q}{F}$$

where $F(X) = \prod_{i=1}^{m}(X - \gamma_i)$. To see why F divides $S \cdot Q$, consider a root γ_k of F. Then, either $x_k = 0$ and $S(\gamma_k) = 0$ or $x_k \neq 0$ and $Q(\gamma_k) = 0$. In both cases, $S \cdot Q(\gamma_k) = 0$. Therefore, P is well-defined and has degree at most $w - 1$.

Therefore, to convince a verifier of knowledge of syndrome decoding solution x, it suffices to prove that there exist P and Q satisfying $\deg P \leq \omega - 1$ and $\deg Q = \omega$ such that

$$S \cdot Q = P \cdot F,$$

where S is constructed by interpolation as described above. Since F has roots at each γ_i, the verifier can infer from this relation that $S(\gamma_i) = 0$ unless $Q(\gamma_i) = 0$. As Q has at most ω roots, it follows that $\mathrm{wt}(x) \leq \omega$.

We now describe how to use this for a proof in the MPC-in-the-Head framework. First, consider the MPC protocol, which, given inputs x, P, and Q, will output `Accept` if the equality above holds and `Reject` otherwise, with a negligible false positive probability. The shares of the polynomials are distributed as additive shares of their coefficient vectors, with the exception of Q's leading coefficient, which is public and equal to 1 to enforce its degree.

The protocol proceeds as follows:

1. Each party $\mathcal{P}_i$ holds the shares $[\![x_A]\!]_i, [\![Q]\!]_i, [\![P]\!]_i$.
2. The parties sample t random evaluation points $r_1, \ldots, r_t$ in a larger field $\mathbb{F}_{\text{points}} \supset \mathbb{F}_{poly}$.
3. They locally compute $[\![x]\!]$ from $[\![x_A]\!]$ using equation (1).
4. The parties compute $[\![S(r_j)]\!]$, $[\![Q(r_j)]\!]$, and $[\![(F \cdot P)(r_j)]\!]$ for each $j \in [t]$. The computation of $[\![S(r_j)]\!]$ uses the linearity of Lagrange interpolation:

$$[\![S(r_j)]\!] = \sum_{i=1}^{m} [\![x_i]\!] \prod_{\ell \neq i} \frac{r_j - \gamma_\ell}{\gamma_i - \gamma_\ell}$$

   $[\![(F \cdot P)(r_j)]\!]$ can be directy computed as $F(r_j)[\![(P)(r_j)]\!]$ as F is a public polynomial.
5. For each $j \in [t]$, the parties verify the product relation $Q(r_j) \cdot S(r_j) = (F \cdot P)(r_j)$ using a protocol proposed by Lindell and Nof [LN17]. They will sacrifice a an additively shared multiplication triple (a_j, b_j, c_j) such that $a_j \cdot b_j = c_j$:
   (a) Sample a random $\varepsilon_j \in \mathbb{F}_{\text{points}}$.
   (b) Compute
   $$[\![\alpha_j]\!] = \varepsilon_j [\![Q(r_j)]\!] + [\![a_j]\!], \quad [\![\beta_j]\!] = [\![S(r_j)]\!] + [\![b_j]\!]$$
   (c) Open $[\![\alpha_j]\!], [\![\beta_j]\!]$ to get α_j, β_j.
   (d) Compute
   $$[\![v_j]\!] = \varepsilon_j [\![(F \cdot P)(r_j)]\!] - [\![c]\!] + \alpha_j [\![b_j]\!] + \beta_j [\![a_j]\!] - \alpha_j \beta_j$$
   (e) Open $[\![v_j]\!]$ to obtain v_j.
6. The protocol outputs `Accept` if all $v_j = 0$, otherwise `Reject`.

Notice that the product check protocol introduces an additional false positive probability. Given P, S, Q such that $P \cdot F \neq S \cdot Q$, consider a point r_j such that $S(r_j) \cdot Q(r_j) \neq (F \cdot P)(r_j)$. Then, the computations result in

$$\begin{aligned} v &= \varepsilon_j (F \cdot P)(r_j) - c_j + \alpha_j b_j + \beta_j a_j - \alpha_j \beta_j \\ &= \varepsilon_j (F \cdot P)(r_j) - c_j + (\varepsilon_j Q(r_j) + a_j) b_j + (S(r_j) + b_j) a_j - (\varepsilon_j Q(r_j) + a_j)(S(r_j) + b_j) \\ &= \varepsilon_j (F \cdot P)(r_j) - c_j + (S(r_j) + b_j) a_j - (\varepsilon_j Q(r_j) + a_j)(S(r_j)) \\ &= \varepsilon_j (F \cdot P)(r_j) - c_j - \varepsilon_j Q(r_j) S(r_j) + a_j b_j \\ &= \varepsilon_j ((F \cdot P)(r_j) - Q(r_j) S(r_j)) - c_j + a_j b_j. \end{aligned}$$

By setting $v = 0$ and assuming $c_j \neq a_j b_j$, then solving for ϵ_j we see that there is a $\frac{1}{|\mathbb{F}_{\text{points}}|}$ false positive probability if $S(r_j) \cdot Q(r_j) \neq (F \cdot P)(r_j)$. When transforming to the zero-knowledge setting, the random challenges (r_j, ε_j) will be provided by the verifier.

The above MPC protocol can be summarized as computing a non-deterministic function f taking x, Q, and P (plus t multiplication triples) as input and outputs `Accept` or `Reject`. Randomness comes from the random evaluations points $r_1, \ldots, r_t$ and the product checking protocol's challenges $\varepsilon_1, \ldots, \varepsilon_t$. Whenever x satisfies $\text{wt}(x) \leq \omega$ and polynomials P and Q are genuinely computed, the protocol outputs `Accept` with probability one. Otherwise, it outputs `Reject` except with a small false positive probability p.

Let's make explicit the false positive probability p. Set $\Delta := |\mathbb{F}_{\text{points}}|$. If the input is not valid, i.e. $\text{wt}(x) > \omega$ or P, Q are not well-formed, we have $S \cdot Q \neq F \cdot P$. Evaluating both sides at t random points and applying Lemma 3, the probability to get equality at i out of t points is at most

$$\frac{\max_{\ell \leq m+\omega-1} \left\{ \binom{\ell}{i} \binom{\Delta-\ell}{t-i} \right\}}{\binom{\Delta}{t}},$$

since $S \cdot Q - F \cdot P$ is a polynomial of degree at most $m + \omega - 1$. When this occurs, the probability to get `Accept` is:

$$\left(\frac{1}{\Delta}\right)^{t-i}$$

which corresponds to $t - i$ false positives for the remaining evaluation points r_j where $Q(r_j) \cdot S(r_j) \neq F(r_j) \cdot P(r_j)$. Thus, the global false positive probability satisfies:

$$p_f \leq \sum_{i=0}^{t} \frac{\max_{\ell \leq m+\omega-1} \left\{ \binom{\ell}{i} \binom{\Delta-\ell}{t-i} \right\}}{\binom{\Delta}{t}} \left(\frac{1}{\Delta}\right)^{t-i} \tag{2}$$

2.6 Hypercube MPCitH

An important optimization in the SDitH protocol is the use of a hypercube structure to arrange the parties, introduced by Aguilar-Melchor *et al.* [AMGH+23]. While the two-party protocol proposed in this thesis does not rely on hypercube geometry, this technique is incorporated for its performance benefits. This section summarizes the main ideas behind hypercube MPCitH, following the same approach as Aguilar-Melchor *et al.* [AMGH+23].

The key idea is to *recycle generated random values* for multiple independent executions of the protocol. This allows us to reduce the computational cost of MPC-in-the-Head (MPCitH) protocols while preserving soundness. To achieve this, we arrange additive shares of in-the-head parties in a hypercube and combine them for multiple, smaller MPC

executions. A general parameter n is introduced for the number of parties: in SDitH, $n = N$, while in the hypercube-based variant, $n = N^D$.

In a standard MPCitH protocol, the prover generates n additive shares of a witness, simulates the MPC protocol for all n parties, and commits to the internal states. Later, $n - 1$ of these are opened for verification. In the hypercube approach, the $n = 2^D$ shares are arranged in a D-dimensional hypercube, and D MPCitH executions are performed, each involving N parties. This reduces the number of parties we have to simulate, while maintaining the same soundness error.

For illustration, consider a 4-party protocol where shares $s_1, s_2, s_3, \text{aux}$ sum to the witness. Arranging four in-the-head shares into a two-dimensional hypercube of side 2 like in Table 2 allows running the underlying MPC on rows and columns. Instead of running

s_1	s_3
s_2	aux

Table 2. Hypercube Example

one 4-party protocol, we run two 2-party protocols on the following splits:

$$m_1 = s_1 + s_2, \quad m_2 = s_3 + \text{aux},$$
$$n_1 = s_1 + s_3, \quad n_2 = s_2 + \text{aux}.$$

Notice that the two splits are independent - knowing all-but-one of the row shares does not reveal anything about the column shares.

Let us introduce the following notation: $[\![s]\!]_{k,i}$ denotes the i^{th} *main party share* in the k^{th} dimension. So $[\![s]\!]_{1,1} = n_1, [\![s]\!]_{1,2} = n_2$. We can consider each leaf party index as a vector of indices - leaf party $\boldsymbol{i}$ can be represented as its hypercube indices $(i_1, i_2, \ldots, i_N)$. In general main party shares are aggregated as follows

$$[\![s]\!]_{k,j} = \sum_{i_k = j} [\![s]\!]_{\boldsymbol{i}}.$$

While this offers no efficiency gain for $D = 2$, increasing D quickly improves performance. For example, a 256-party protocol typically requires 256 MPC simulations. Using an 8-dimensional hypercube of side 2 reduces this to $2 \times 8 = 16$ MPC runs, with no increase in proof size.

This method generalizes to any N^D-party protocol by arranging leaf parties in a D-dimensional hypercube and performing one MPC execution per axis, each aggregating shares along that dimension.

2.7 GGM Trees

MPC in-the-head schemes make extensive use of tree PRG (TreePRG) to effectively generate and communicate randomness, initially proposed by Goldreich, Goldwasser, and Micali [GGM86]. The main idea is to apply a length-doubling pseudo-random generator on a root seed in a binary tree structure.

To generate 2^n random seeds, we can recursively apply the length-doubling PRG to a root seed, which, after n layers of application, has generated 2^n seeds. Those seeds can be extended to 2^n random values. Using the TreePRG structure, we can effectively open $2^n - 1$ random values. To open all-but-one random seeds, we need to send the unopened seeds' *sibling path*. That sibling path consists of the siblings of the $n - 1$ nodes from the root of the tree to the unopened leaf.

This allows us to generate N witness shares efficiently and open $N-1$ of them by sending only $\lceil \log_2 N \rceil$ values.

Despite their efficiency in the single-prover setting, GGM trees are one of the main barriers in constructing threshold signatures from MPCitH. They make extensive use of PRG functions, which are often implemented by hash functions. Applying a straightforward approach to a distributed witness scenario, multiple users would have to generate an additively shared GGM tree. As PRG functions are not homomorphic, every execution would require the MPC evaluation of a large number of PRG. A recent preprint estimated that the communication for realistic parameters would be almost a terabyte [CC24].

3 Two-Party SDitH Protocol

In this section, we propose a two-prover zero-knowledge protocol in which the provers $\mathcal{P}_1, \mathcal{P}_2$, under public parameters $H \in \mathbb{F}^{r \times n}, y \in \mathbb{F}^r$ aim to convince a verifier $\mathcal{V}$ that they know low weight vectors x^1, x^2 such that $(x^1 + x^2)H^T = y$.

3.1 Towards Multi-Prover MPCitH

A recent impossibility result [DKR24] showed that MPC-in-the-Head signatures either

- make non-black-box use of hash functions/PRGs, or
- grow linearly in size with the number of provers.

Because the cost of running a real MPC protocol for hash functions/PRGs is extremely high, we will focus on the second option. Multiple general transformations have been proposed to create multi-prover zero-knowledge proofs from general zero-knowledge proofs. They can be rendered non-interactive using the Fiar-Shamir transformation to create threshold signatures.

In particular, Cui *et al.* [CZC+21] proposed a construction specifically to transform MPCitH proofs into multi-prover zero-knowledge proofs. If the underlying protocol splits a witness w into n in-the-head parties, the authors propose to give each of the m real parties responsibility of n/m in-the-head shares and then, propagate all communication between the in-the-head parties. For example, if in the single prover proof, the in-the-Head party i sends a message to party j, then after applying the MPC-in-the-Multi-Head transformation, the holder of party i would propagate the message to the holder of party j.

This approach results in a lot of communication, even in the SDitH setting where the underlying protocol is simple with each party opening two values $[\![\alpha]\!], [\![\beta]\!]$. The multiple provers in this thesis will aggregate their shares' communications before transmission, reducing the communication to a constant number of field elements.

The most relevant work to this thesis is that of Carozza and Couteau [CC24], which offers concrete optimizations in light of the impossibility result stated above. Before detailing an optimized approach, it provides two trivial approaches to thresholdize MPCitH signatures:

- The first approach is for all provers to sign the message m and then concatenate the signatures. Concretely, given n secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_n$ and a message m, the provers compute $\sigma_i = \mathcal{S}(\mathsf{sk}_i, m)$ for $i = 1, \ldots, n$ and obtain a combined signature like $\sigma = \sigma_1 || \ldots || \sigma_n$. This approach has an important semantic difference from a threshold signature - it conveys the information that the n parties have signed a message. A valid collaborative signature, such as the one proposed in this thesis,

means that n parties actively collaborated at some point in time to sign a message, and they are aware and agree to the other provers' collaboration.

- The second approach is to use MPC for simulating the entire protocol, including MPC executions of the PRG.

As mentioned above, the second approach results in a prohibitive communication cost. To the knowledge of the author of this thesis, this is the only paper which specifically works towards threshold signatures from MPCitH. A crucial similarity to this thesis is that the authors also propose to additively share all the N^D witness shares among the real parties. Additionally, the two works use the same procedure for creating a sharing of the witness, where multiple provers generate a GGM tree locally to commit to witness shares. Another common optimization is that the provers always prefer to share hash digests of their concatenated commitments to reduce communication. Where the two approaches diverge is the simulation of the MPC protocol, this thesis proposes a concrete optimization to reduce the communication between the provers.

The protocol proposed in this thesis follows the hypercube SDitH protocol by Aguilar-Melchor *et al.* [AMGH+23] and is specifically designed to minimize communication between signers.

Trusted Setup: For simplicity, this work assumes that a witness is distributed through a maliciously secure trusted setup. $\mathcal{P}_p$ knows polynomials x_A^p, S^p, Q^p, P^p such that $(S^1 + S^2)(Q^1 + Q^2) = (P^1 + P^2)F$. Denote those additive shares by $[\![\boldsymbol{S}]\!]^p, [\![\boldsymbol{Q}]\!]^p, [\![\boldsymbol{P}]\!]^p, [\![x_A]\!]^p$ respectively. Furthermore, the two parties hold a multiplication triple $[\![a]\!], [\![b]\!], [\![c]\!] \coloneqq [\![ab]\!]$.

The proposed two-party SDitH protocol proceeds in five rounds, with the following structure. Here, we will ignore the hypercube structure for simplicity.

Round 1 (Commitment Generation): Each prover samples a root seed and derives a N^D leaf seeds via a tree-based pseudorandom generator (TreePRG). These seeds are extended to additive shares of the witness, with both real parties setting the last in-the-head share so that all local shares add up to that provers' witness. Both parties broadcast the hash of all commitments to their in-the-head parties.

Round 2 (First Challenge): The verifier samples and broadcasts the first challenge, consisting of a random point r and a value ε.

Round 3 (MPC Protocol): Each prover computes and broadcasts aggregated communications $[\![\alpha]\!]^p$ and $[\![\beta]\!]^p$ using the verifier's challenge. Aggregating $[\![\alpha]\!]^p, [\![\beta]\!]^p$ at this point is a key optimization of this thesis, greatly reducing the total communication cost. Using the globally opened values α and β, the provers proceed to compute the values $[\![v]\!]$ as described in the previous section. Both parties computed and broadcast a hash digest H^p of parties' communications.

Round 4 (Second Challenge): The verifier samples and broadcasts the second challenge - an index i^* of a leaf party.

Round 5 (Opening): Each prover reveals the requested parties' sibling paths commitments, along with the commitments and communications of the unopened party i^*

Verification: The verifier expands the sibling paths for both provers into all-but-one states of both provers, and checks the consistency of first round commitments. Because all-but-one states can be reconstructed from the sibling path of the hidden state, we use sibling paths and the states themselves interchangeably. Then, the verifier simulates the communications of the MPC protocol for both provers, checks that they are consistent with the hash digest H^p, and finally checks that the values $[\![v]\!]$ globally reconstruct to 0.

Algorithm 1. Two-party SDitH

Prover $\mathcal{P}_p$ knows $[\![\boldsymbol{x}_A]\!]^p, [\![\boldsymbol{S}]\!]^p, [\![\boldsymbol{Q}]\!]^p, [\![\boldsymbol{P}]\!]^p, [\![a]\!]^p, [\![a]\!]^p, [\![b]\!]^p, [\![c]\!]^p$

Round 1:

for party $p \in \{1,2\}$ **do**

Sample a root seed: $\mathsf{seed}^p \xleftarrow{\$} \{0,1\}^\lambda$

Derive leaf seeds $\mathsf{seed}^p_{i'}$ using TreePRG.

for each leaf $i' = 1$ to N^D **do**

if $i' \neq N^D$ **then**

$\mathsf{state}_{i'} = \{[\![a]\!]^p_{i'}, [\![b]\!]^p_{i'}, [\![c]\!]^p_{i'}, [\![\boldsymbol{x}_A]\!]^p_{i'}, [\![\boldsymbol{Q}]\!]^p_{i'}, [\![\boldsymbol{P}]\!]^p_{i'}\} \leftarrow \mathsf{PRG}(\mathsf{seed}_{i'})$

else

$[\![\boldsymbol{x}_A]\!]^p_{N^D} = [\![\boldsymbol{x}_A]\!]^p - \sum_{i=1}^{N^D-1} [\![\boldsymbol{x}_A]\!]^p_i, \quad [\![a]\!]^p_{N^D} = a^p - \sum_{i=1}^{N^D-1} [\![a]\!]^p_i$

$[\![\boldsymbol{Q}]\!]^p_{N^D} = [\![\boldsymbol{Q}]\!]^p - \sum_{i=1}^{N^D-1} [\![\boldsymbol{Q}]\!]^p_i, \qquad [\![b]\!]^p_{N^D} = b^p - \sum_{i=1}^{N^D-1} [\![b]\!]^p_i$

$[\![\boldsymbol{P}]\!]^p_{N^D} = [\![\boldsymbol{P}]\!]^p - \sum_{i=1}^{N^D-1} [\![\boldsymbol{P}]\!]^p_i, \qquad [\![c]\!]^p_{N^D} = c^p - \sum_{i=1}^{N^D-1} [\![c]\!]^p_i$

$\mathsf{state}_{i'} = \{[\![a]\!]^p_{N^D}, [\![b]\!]^p_{N^D}, [\![c]\!]^p_{N^D}, [\![\boldsymbol{x}_A]\!]^p_{N^D}, [\![\boldsymbol{Q}]\!]^p_{N^D}, [\![\boldsymbol{P}]\!]^p_{N^D}\}$

end if

for each main party index (k, i_k) in $\{(1,i_k),(2,i_k),\ldots,(D,i_k)\}$ **do**

$[\![\boldsymbol{x}_A]\!]^p_{(k,i_k)} += [\![\boldsymbol{x}_A]\!]^p_{i'}, [\![\boldsymbol{Q}]\!]^p_{(k,i_k)} += [\![\boldsymbol{Q}]\!]^p_{i'}, [\![\boldsymbol{P}]\!]^p_{(k,i_k)} += [\![\boldsymbol{P}]\!]^p_{i'}$

$[\![a]\!]^p_{(k,i_k)} += [\![a]\!]^p_{i'}, [\![b]\!]^p_{(k,i_k)} += [\![b]\!]^p_{i'}, [\![c]\!]^p_{(k,i_k)} += [\![c]\!]^p_{i'}$

end for

$\mathsf{com}^p_{i'} = \mathsf{Hash}_0(i', \mathsf{state}_{i'})$.

end for

$\mathsf{COM}^p_1 = \mathsf{Hash}(\mathsf{com}^p_1, \ldots, \mathsf{com}^p_{N^D})$

Broadcast COM^p_1.

end for

Round 2: $\mathcal{V}$ broadcasts the first challenge (r, ε).

Round 3:

for party $p \in \{1,2\}$ **do**

for dimension $k \in [D]$ **do**

for main party index $i \in [N]$ **do**

Set $[\![\alpha]\!]^p_{k,i} = \epsilon[\![\boldsymbol{Q}(r)]\!]^p_{k,i} + [\![a]\!]^p_{k,i}$.

Set $[\![\beta]\!]^p_{k,i} = [\![\boldsymbol{S}(r)]\!]^p_{k,i} + [\![b_l]\!]^p_{k,i}$.

end for

Once: broadcast $[\![\alpha]\!]^p = \sum_{i=1}^N [\![\alpha]\!]_{1,i}, \quad [\![\beta]\!]^p = \sum_{i=1}^N [\![\beta]\!]_{1,i}$.

Set $\alpha = [\![\alpha]\!]^1 + [\![\alpha]\!]^2$.

for main party index $i \in [N]$ **do**

$$[\![v]\!]^p_{k,i} = -[\![c]\!]^p_{k,i} + \langle \epsilon, \mathbf{F}(r) \cdot [\![\boldsymbol{P}(r)]\!]^p_{k,i} \rangle + \langle \alpha, [\![b]\!]^p_{k,i} \rangle + \langle \beta, [\![a]\!]^p_{k,i} \rangle - \langle \alpha, \beta \rangle.$$

end for

end for

Set $H^p_k = \mathsf{Hash}([\![\boldsymbol{\alpha}]\!]^p_k, [\![\boldsymbol{\beta}]\!]^p_k, [\![\boldsymbol{v}]\!]^p_k)$, where $[\![\boldsymbol{\alpha}]\!]^p_k$ is the the vector $([\![\alpha]\!]^p_{k,1}, \ldots [\![\alpha]\!]^p_{k,N})$.

Broadcast $h^p = \mathsf{Hash}(H^p_1, \ldots H^p_D)$.

end for

Round 4: $\mathcal{V}$ broadcasts the second challenge (i^*).

Round 5:

for party $p \in \{1,2\}$ **do**

Broadcast the sibling path for $\mathsf{state}^p_{i_1}, \ldots, \mathsf{state}^p_{i_D} \forall (i_1, \ldots, i_D) \neq (i^*_1, \ldots, i^*_D)$ to $\mathcal{V}$.

Broadcast $\mathsf{com}^p_{(i^*_1,\ldots,i^*_D)}$, $[\![\alpha]\!]^p_{(i^*_1,\ldots,i^*_D)}$, $[\![\beta]\!]^p_{(i^*_1,\ldots,i^*_D)}$ and $[\![v]\!]^p_{(i^*_1,\ldots,i^*_D)}$.

end for

Verification: Verifier accepts iff Algorithm 2 accepts.

Algorithm 2. Verification

Input: $\{\mathtt{COM}^p, h^p, \{\mathtt{state}_i^p\}_{i\neq \mathbf{i}^*}, \mathtt{com}_{\mathbf{i}^*}^p, [\![\boldsymbol{\alpha}]\!]_{\mathbf{i}^*}^p, [\![\boldsymbol{\beta}]\!]_{\mathbf{i}^*}^p, [\![\boldsymbol{v}]\!]_{\mathbf{i}^*}^p\}_{p\in\{1,2\}}$, (r,ϵ), $\mathbf{i}^*$.
Output: Accept or Reject.

for $i \neq \mathbf{i}^*$ **do**
Calculate $\mathtt{com}_i^1 = \mathtt{Hash}(i, \mathtt{state}_i^1)$, $\mathtt{com}_i^2 = \mathtt{Hash}(i, \mathtt{state}_i^2)$.
end for
for party $p \in \{1,2\}$ **do**
Calculate $h_1^p = \mathtt{Hash}(\mathtt{com}_1^p, \ldots, \mathtt{com}_{\mathbf{i}^*}^p, \ldots, \mathtt{com}_{N^D}^p)$
end for
if $h_1^1 \neq \mathtt{COM}^1$ **or** $h_1^2 \neq \mathtt{COM}^2$ **then**
Reject
end if
Obtain main party states by aggregating leaf parties.

for party $p \in \{1,2\}$ **do**
for dimension $k \in [D]$ **do**
for main party index $i \in [N]$ **do**
Calculate main party shares $[\![\alpha]\!]_{k,i}^p, [\![\beta]\!]_{k,i}^p$, using $[\![\alpha]\!]_{i^*}^p, [\![\beta]\!]_{i^*}^p$ when $i = i_k^*$. Otherwise:
Set $[\![\alpha]\!]_{k,i}^p = \epsilon[\![\boldsymbol{Q}(r)]\!]_{k,i}^p + [\![a]\!]_{k,i}^p$.
Set $[\![\beta]\!]_{k,i}^p = [\![\boldsymbol{S}(r)]\!]_{k,i}^p + [\![b_l]\!]_{k,i}^p$.
end for
Open $[\![\alpha]\!]_k^p, [\![\beta]\!]_k^p$.
Set $[\![\alpha]\!]_k = [\![\alpha]\!]_k^1 + [\![\alpha]\!]_k^2$, $[\![\beta]\!]_k = [\![\beta]\!]_k^1 + [\![\beta]\!]_k^2$.
for main party index $i \in [N]$ **do**
if $i_k^* = i$ **then** Calculate $[\![v]\!]_{k,i}^p$ by adding $[\![v]\!]_{i^*}^p$ to the partial aggregation of $[\![v]\!]_{k,i}^p$.
else

$$[\![v]\!]_{k,i}^p = -[\![c]\!]_{k,i}^p + \langle \epsilon, \mathbf{F}(r) \cdot [\![\boldsymbol{P}(r)]\!]_{k,i}^p \rangle + \langle [\![\alpha]\!]_k, [\![b]\!]_{k,i}^p \rangle + \langle \beta_k, [\![a]\!]_{k,i}^p \rangle - \langle [\![\alpha]\!]_k, \beta_k \rangle.$$

end if
end for
Set $[\![v]\!]_k^p = \sum_{i=1}^N [\![v]\!]_{k,i}$
end for
Set $\tilde{H}_k^p = \mathtt{Hash}([\![\boldsymbol{\alpha}]\!]_k^p, [\![\boldsymbol{\beta}]\!]_k^p, [\![\boldsymbol{v}]\!]_k^p)$
Accept if for $p \in \{1,2\}$: $\tilde{h}^p = \mathtt{Hash}(\tilde{H}_1^p, \ldots \tilde{H}_D^p) = h^p$ and for all dimensions $k \in [D]$: $[\![\alpha]\!]_k^p, [\![\beta]\!]_k^p, [\![v]\!]_k^p$ are equal and $[\![v]\!]_k^1 + [\![v]\!]_k^2 = 0$.
end for

4 Proof of Security

In this section, we will prove all the desired security properties of Algorithm 1, while carefully exploring security in the multi-prover setting.

4.1 Correctness

To show that Algorithm 1 is perfectly correct, let us examine how the shares of $[\![v]\!]$ globally reconstruct to 0. To this end, let us forget about the hypercube approach and consider the unstructured n in-the-Head parties shared between real parties. Denote $P = [\![P]\!]^1 + [\![P]\!]^2, S = [\![S]\!]^1 + [\![S]\!]^2, Q = [\![Q]\!]^1 + [\![Q]\!]^2$. Assume a good witness satisfying

$S \cdot Q = P \cdot F$. The opened communications α, β are by both provers as

$$\begin{aligned}
\alpha &= [\![\alpha]\!]^1 + [\![\alpha]\!]^2 = \varepsilon \cdot \sum_i \left([\![Q(r)]\!]_i^1 + [\![a]\!]^1\right) + \varepsilon \cdot \sum_i \left([\![Q(r)]\!]_i^2 + [\![a]\!]^2\right) \\
&= \varepsilon \cdot ([\![Q]\!]^1 + [\![Q]\!]^2)(r) + a \\
&= \varepsilon \cdot Q(r) + a, \\
\beta &= [\![\beta]\!]^1 + [\![\beta]\!]^2 \\
&= ([\![S]\!]^1 + [\![S]\!]^2)(r) + b \\
&= S(r) + b.
\end{aligned}$$

Consider the sum of all additive shares of $[\![v]\!]$ for a the p^{th} prover:

$$[\![v]\!]^p = \varepsilon F(r) \cdot [\![P(r)]\!]^p - [\![c]\!]^p + \alpha \cdot [\![b]\!]^p + \beta \cdot [\![a]\!]^p - \alpha\beta.$$

This means that $[\![v]\!]$ globally reconstructs to

$$\begin{aligned}
v &= [\![v]\!]^1 + [\![v]\!]^2 \\
&= \varepsilon F(r) \cdot ([\![P(r)]\!]^1 + [\![P(r)]\!]^2) - c + \alpha \cdot ([\![b]\!]^1 + [\![b]\!]^2) + \beta \cdot ([\![a]\!]^1 + [\![a]\!]^2) - \alpha\beta \\
&= \varepsilon \cdot F(r) \cdot P(r) - c + (\epsilon Q(r) + a) \cdot b + (S(r) + b) \cdot a - (\epsilon Q(r) + a)(S(r) + b) \\
&= 0.
\end{aligned}$$

4.2 Defining Security

In this section, we will follow the simulation-based security definitions of Lindell and Nof [Lin17], particularly the notion of a m-party honest verifier zero-knowledge proof introduced by Cui *et al.* [CZC+21].

In the simulation paradigm, the security of a protocol is proven by constructing a simulator that can emulate the honest execution of a protocol so that it is indistinguishable from an ideal functionality, which is secure by definition. The presence of a malicious adversary $\mathcal{A}$, who can actively control any strict subset of parties, can be accounted for by allowing $\mathcal{A}$ to abort without sending its output to any of the other parties.

Definition 7 (Secure Computation)**.** *An n-party protocol defined by n interactive Turing machines $M_1, ..., M_n$ is said to securely compute an n-party functionality $\mathcal{F}$ if there exists a PPT simulator $\mathcal{S}$, such that for all PPT adversaries $\mathcal{A}$ that may corrupt a subset $\mathcal{I}$ of parties, the joint outputs of the adversary and honest parties in the real world are indistinguishable from that of the simulated outputs of $\mathcal{A}$ and honest parties in the ideal world.*

It is implicit in this definition that an adversary $\mathcal{A}$ may additionally output any polynomial-time computable function, which importantly includes its view. To take this into account, the ideal-world simulator must be able to generate views which are indistinguishable from the views of all parties controlled by $\mathcal{A}$ in the real world.

Definition 8 (Extended Relation R^m)**.** *For an integer $m \in \mathbb{N}$, an* extended relation R^m *defined by a polynomial-size uniform circuit family $\{C_n\}_{n\in\mathbb{N}}$ such that each circuit C_n takes $m + 1$ inputs and*

$$(x, y_1, \ldots, y_m) \in R^m \text{ if and only if } C(x, y_1, \ldots, y_m) = 1.$$

In the two-party distributed syndrome decoding context the desired relation R^2 is implemented by the circuit C, which, for public $(H, \boldsymbol{y}, \omega)$ and private inputs $\boldsymbol{x}_1, \boldsymbol{x}_2$, checks if $(\boldsymbol{x}_1 + \boldsymbol{x}_2)H = \boldsymbol{y}$ and that $\mathrm{wt}(\boldsymbol{x}_1 + \boldsymbol{x}_2) \leq \omega$.

Definition 9 (Language Induced by R^m)**.** *Given an extended relation R^m, the corresponding* language $L(R)$ *is defined as*

$$L(R) = \{x \mid \exists y_1, \ldots, y_m \text{ such that } (x, y_1, \ldots, y_m) \in R^m\}.$$

Equivalently, $x \in L(R)$ if and only if there exist witnesses $y_1, \ldots, y_m$ such that $C(x, y_1, \ldots, y_m) = 1$.

For the relation R^2, $L(R)$ is precisely the set of syndromes $\boldsymbol{y}$ for which there exists a preimage $\boldsymbol{x}_1 + \boldsymbol{x}_2$ such that $(\boldsymbol{x}_1 + \boldsymbol{x}_2)H = y$.

Definition 10 (Proof System Syntax)**.** *A multi-prover zero-knowledge proof system Π with m provers for an extended $\mathcal{NP}$ relation R^m is defined by $m+1$ interactive Turing machines: m provers $\mathcal{P}_1, \ldots, \mathcal{P}_m$ and a verifier $\mathcal{V}$. Each prover $\mathcal{P}_i$ holds a public input x and a private witness w_i while verifier $\mathcal{V}$ only holds the public input. The provers try to convince the verifier that $x \in L(R)$.*

Functionality $\mathcal{F}_{R^m}^{\mathsf{zk}}$

On input (sid, x, w_i) from $P_1, \ldots, P_m$ and (sid, x) from V, the functionality checks whether all parties have sent consistent x values.

- If the check fails, it sends a message $(\mathsf{sid}, \perp)$ indicating failure to all parties.
- Otherwise, it sets $y = C(x, w_1, \ldots, w_m)$ and then sends (sid, y) to $\mathcal{A}$
  - If $\mathcal{A}$ sends back $(\mathsf{sid}, \mathsf{continue})$, then it forwards (sid, y) to honest parties;
  - Otherwise, it sends $(\mathsf{sid}, \perp)$ instead.

Figure 1. The m-prover zero-knowledge functionality $\mathcal{F}_{R^m}^{\mathsf{zk}}$ for extended $\mathcal{NP}$ relation R^m.

Definition 11 (m-prover Zero-Knowledge Proof System)**.** *With regard to an extended $\mathcal{NP}$ relation R^m, an $(m+1)$-party protocol Π following the syntax of 4.2 is an m-*prover zero-knowledge proof system *if it* securely computes *the ideal zero-knowledge functionality $\mathcal{F}_{R^m}^{\mathsf{zk}}$ defined in Fig. 1 against a PPT adversary $\mathcal{A}$ that may corrupt a subset $\mathcal{I}$ of parties with different controlling capabilities.*

Let $S_P = \{P_1, \ldots, P_m\}$ denote the set of all provers. The corrupted parties $\mathcal{I}$ can be the following cases:

- ***Soundness.*** *$\{P_1, \ldots, P_m\} \subseteq \mathcal{I}$ and all $\{P_1, \ldots, P_m\}$ are actively* controlled*;*
- ***Standard ZK.*** *$V \in \mathcal{I}$ and the V is actively* controlled*.*
- ***Honest-Verifier ZK.*** *$V \in \mathcal{I}$ and V is passively* controlled*.*

Additionally, this proof system has passive/active $\mathcal{S}$***-partial witness privacy*** *(*$\mathcal{S}$ *contains subsets of* S_P*) if the adversary has the power to passively/actively control a subset* $Q \in \mathcal{S}$ *of provers with the possible additional ability to passively control the verifier (i.e.,* $V \in \mathcal{I} \wedge \mathcal{I} \setminus \{V\} \in \mathcal{S}$*). If the definition of* $\mathcal{S}$ *is based on a threshold* $t \in \mathbb{N}$ *(namely,* $\mathcal{S} = \{S \subseteq S_P : |S| \leq t\}$*), then we call it* t-witness privacy.

The simulators in this thesis are slightly informal, they never explicitly interact with ideal functionalities. Rather, the ideal functionality $\mathcal{F}^{\mathsf{zk}}_{R^m}$ is the formalization of the following informal properties:

1. Corrupt parties should not learn anything about the other provers' witnesses, other than whether all committed witnesses reconstruct to a valid solution to the syndrome decoding problem.

2. An adversary should be able to simulate the messages of the honest parties without access to their witness shares.

To formalize the idea of witness extractability in the multi-prover setting, we look to the definitition of multi-prover knowledge soundness as defined by Ozdemir and Boneh [OB22] and Liu *et al.* [LZW+24].

Definition 12 (Knowledge Soundness)**.** *Denote by* π^* *the transcript of a multi-prover zero-knowledge proof. For all public inputs* x *and for all sets of probabilistic polynomial-time (PPT) adversarial algorithms* $\vec{\mathcal{A}} = \{\mathcal{A}_1, \ldots, \mathcal{A}_N\}$ *controlling the provers, there exists a PPT extractor* $\mathcal{E}$ *such that*

$$\Pr\left[\mathsf{Verify}(x, \pi^*) = 1 : \begin{array}{l} w^* \leftarrow \mathcal{E}^{\vec{\mathcal{A}}}(x), \\ \pi^* \leftarrow \vec{\mathcal{A}}(x), \\ (x, w^*) \notin R^m \end{array}\right] \leq \mathsf{negl}(\lambda)$$

In words: if a set of malicious provers interacting via algorithms $\vec{\mathcal{A}}$ *can convince the verifier to accept a proof for* $x \in L(R)$*, then an extractor* $\mathcal{E}$ *exists which can efficiently recover a witness tuple* $w^* = (y_1, \ldots, y_m)$ *such that* $(x, w^*) \in R^m$*, except with negligible probability in the security parameter* λ*.*

4.3 Honest-Verifier Zero Knowledge

Theorem 1 (Honest-Verifier Zero Knowledge (HVZK))**.** *If the outputs of PRG and commitment Com are indistinguishable from the uniform random distribution, then Algorithm 1 is Honest-Verifier Zero Knowledge as defined above.*

Proof. To prove the HVZK property, we will construct a simulator $\mathcal{S}$, which will output transcripts which are indistinguishable from the views of the verifier in the real protocol. For this, assume that the PRG of Algorithm 1 is $(t, \epsilon_{\mathsf{PRG}})$-secure and the commitment scheme com is $(t, \epsilon_{\mathsf{com}})$-hiding. The leaf party indices $(i_{k_1}, \ldots, i_{k_D})$ will be denoted by i' and the challenge party index $(i^*_1, \ldots, i^*_D)$ as i^*. To argue that the transcripts detailed in algorithm 1 are indistinguishable from transcripts of the real protocol, we will consider 'intermediate simulators', which are not true simulators, but just a thought exercise to show proximity of $\mathcal{S}$ and Algorithm 1. After every change, we argue that changes to the transcript do not provide a non-negligable advantage to a distinguisher.

Simulator v0 This corresponds to the real execution of the protocol. It receives valid witnesses from the trusted setup and executes Algorithm 1 correctly. The view of the verifier is correct by definition.

Simulator v1 Replace the values contained in $\mathtt{state}^p_{i^*}$ with true randomness, unless $i^* = (N, \dots N)$, in which case $[\![\boldsymbol{x}_A]\!]^p_{N^D}, [\![\boldsymbol{Q}]\!]^p_{N^D}$ and $[\![\boldsymbol{P}]\!]^p_{N^D}$ are generated in the usual way and $[\![a]\!]^p_{N^D}, [\![b]\!]^p_{N^D}$ are truly random. This ensures that all states still add up to the correct witness. The values $[\![\alpha]\!]^p_{i^*_k}, [\![\beta]\!]^p_{i^*_k}, [\![\alpha]\!]^p, [\![\beta]\!]^p$ will be truly randomly distributed, which means that distinguishing **v0** from **v1** is as hard as distinguishing PRG from true randomness. Therefore, a t-time adversary has an advantage of at most ϵ_{PRG}.

Simulator v2 Replace $[\![\boldsymbol{x}_A]\!]^p_{N^D}, [\![\boldsymbol{Q}]\!]^p_{N^D}$, $[\![\boldsymbol{P}]\!]^p_{N^D}$ and $[\![c]\!]^p_{N^D}$ with true randomness. After this alteration, the shares no longer sum up to the global witness.

If $i^* = N^D$, the distribution is identical to **v1**, because changes to unopened state with index N^D only affect the transcript in the values $[\![\alpha]\!]^p_{i^*_k}, [\![\beta]\!]^p_{i^*_k}$, which were already truly randomly distributed in **v1**.

If $i^* \neq N^D$, the distribution also remains identical to **v1**, because all values in $\mathtt{state}^p_{N^D}$ included a truly random summand from party i^*. For example, $[\![\boldsymbol{x}_A]\!]^p_{N^D} = [\![\boldsymbol{x}_A]\!]^p - \sum_{i=1}^{N^D} [\![x]\!]^p_i$ includes the truly random summand $[\![\boldsymbol{x}]\!]_{i^*}$

Simulator v3 Replace $[\![\alpha]\!]^p_{i^*}, [\![\beta]\!]^p_{i^*}$ by true randomness. The distribution remains unchanged because those values already appear truly random in **v2** and **v1**.

Simulator v4 (HVZK Simulator $\mathcal{S}$) Replace $\mathtt{com}_{i^*}$ with an uniformly random value from the domain of the commitment scheme. In summary, the simulator $\mathcal{S}$ behaves as follows:

1. Samples random challenges $\mathtt{CH}_1, \mathtt{CH}_2$.
2. Run Simulator **v3** to get both responses from provers.
3. Calculate all to-be-opened leaf party commitments $i \neq i^*$ as $\mathtt{com}^p_{i'} = \mathtt{com}(\mathtt{state}^p_{i'}, \rho^p_{i'})$.
4. Uniformly randomly sample $\mathtt{com}^p_{i*}$.
5. Set initial commitments to $\mathtt{COM}^p = \mathtt{Hash}(\mathtt{com}^p_1, \dots, \mathtt{com}^p_{i^*}, \dots, \mathtt{com}^p_{N^D})$
6. Execute the protocol faithfully, with the exception the calculation of $[\![v]\!]^2_{i^*}$, which is set so that $[\![v]\!]$ reconstructs to 0.

Clearly, the difficulty in distinguishing **v3** from **v4** is the same as distinguishing $\mathtt{com}_{i^*}$ from a random commitment. We conclude that the transcripts output by the simulator $\mathcal{S}$ are $(t, \epsilon_{\mathsf{PRG}} + \epsilon_{\mathsf{com}})$ indistinguishable from the transcripts of the true protocol.

Algorithm 3. **HVZK Simulator**

Step 0 (Sample seed):
Sample $\mathtt{seed}^1, \mathtt{seed}^2 \xleftarrow{\$} \{0,1\}^\lambda$
Generate $(\mathtt{seed}^p_{i'}, \rho^p_{i'})$ for all leaf parties via TreePRG(seed)

Step 1 (Sample Challenges):
$\mathtt{CH1} = (r, \epsilon) \leftarrow \mathbb{F}_{\mathtt{points}}, \quad \mathtt{CH2} = i^* \leftarrow [1, \dots, N^D]$

Step 2 (Generate Leaf Party States):

Expand root seeds recursively via TreePRG to get N^D leaf states for both parties.

Step 3 (Generate Leaf Party Commitments and Witness Shares):
for each party $p \in \{1, 2\}$ **do**
  for each $i' \neq i^*$ **do**
    Compute $\mathsf{com}^p_{i'} \leftarrow \mathsf{Hash}(\mathsf{state}^p_{i'}, \rho^p_{i'})$
    if $i' \neq N^D$ **then**
      Expand the leaf party seeds into witness share $\mathsf{state}^p_{i'}$.
    else
      Randomly draw:
        $[\![\mathbf{x}_A]\!]^p_{N^D}$, $[\![\mathbf{Q}]\!]^p_{N^D}$, $[\![\mathbf{P}]\!]^p_{N^D}$, $[\![\mathbf{a}]\!]^p_{N^D}$, $[\![\mathbf{b}]\!]^p_{N^D}$, $[\![\mathbf{c}]\!]^p_{N^D}$
    end if
  end for
  for $i' = i^*$ **do**
    Draw $\mathsf{com}^p_{i^*}$ at random
  end for
  Compute initial commitment:
    $\mathsf{COM}^p = \mathsf{Hash}(\mathsf{com}^p_1, \ldots, \mathsf{com}^p_{i^*}, \ldots, \mathsf{com}^p_{N^D})$
  Accumulate main party states honestly.
end for

Step 4 (Generate In-the-Head Party Communications):
for each party $p \in \{1, 2\}$ **do**
  Draw $[\![\alpha]\!]^p_{\mathbf{i}^*}$, $[\![\beta]\!]^p_{\mathbf{i}^*}$ uniformly at random
  for $k \in \{1, \ldots, D\}$ **do**
    Compute $[\![\alpha]\!]_{k,i^*_k}$, $[\![\beta]\!]_{k,i^*_k}$ using $[\![\alpha]\!]^p_{\mathbf{i}^*}$, $[\![\beta]\!]^p_{\mathbf{i}^*}$.
    if $p = 2$ **then**
      Set $[\![v]\!]_{i^*_k} = -\sum_{j\in\{1,2\}} \sum_{i'_k \neq i^*_k} [\![v]\!]^j_{i'_k} - [\![v]\!]^1_{i^*_k}$
    else
      Draw $[\![v]\!]^1_{i^*_k}$ uniformly randomly.
      for $i_k \neq i^*_k$ **do**
        Calculate and accumulate communication shares $[\![\alpha]\!]^p_{k,i}$, $[\![\beta]\!]^p_{k,i}$, $[\![v]\!]^p_{k,i}$
      end for
    end if
  end for
end for

Step 5 (Output Transcript):
for each party $p \in \{1, 2\}$ **do**
  Set $[\![\alpha]\!]^p_k = \sum_{i=1}^n [\![\alpha]\!]_{k,i}$ and $[\![\beta]\!]^p = \sum_{i=1}^n [\![\beta]\!]_{k,i}$
  $h^p \leftarrow \mathsf{Hash}(H^p_1, \ldots, H^p_D)$, where
    H^p_k is honestly computed from $([\![\mathbf{x}_A]\!]^p_k, [\![\mathbf{Q}]\!]^p_k, [\![\mathbf{P}]\!]^p_k, [\![\mathbf{a}]\!]^p_k, [\![\mathbf{b}]\!]^p_k, [\![\mathbf{c}]\!]^p_k, r, \epsilon)$
  $\mathsf{RSP2}^p = \mathsf{com}^p_{i^*}, [\![\alpha]\!]^p_{i^*_k}, [\![\beta]\!]^p_{i^*_k}$,
    and $\{(\mathsf{state}^p_{i'}, \ldots, \rho^p_{i'}) \mid (i_1, \ldots, i_D) \neq (i^*_1, \ldots, i^*_D)\}$
end for
Output$\{\mathsf{COM}^p, h^p, \{\mathsf{state}^p_i\}_{i\neq \mathbf{i}^*}, \mathsf{com}^p_{\mathbf{i}^*}, [\![\alpha]\!]^p_{\mathbf{i}^*}, [\![\beta]\!]^p_{\mathbf{i}^*}, [\![v]\!]^p_{\mathbf{i}^*}\}_{p\in\{1,2\}}$, (r, ϵ), $\mathbf{i}^*$.

□

4.4 Knowledge Soundness

In this section, we will provide a proof for the knowledge soundness of Algorithm 1. The two-prover protocol has no soundness loss compared to its single-prover counterpart. Although the soundness proof closely follows the proof by Aguilar-Melchor *et al.* [AMGH+23], we will include it for completeness.

Lemma 4. *Let $\tilde{\mathcal{P}} = (\mathcal{P}_1, \mathcal{P}_2)$ be provers who jointly commit to a bad witness such that $(\llbracket S \rrbracket^1 + \llbracket S \rrbracket^2) \cdot (\llbracket Q \rrbracket^1 + \llbracket Q \rrbracket^2) \neq (\llbracket P \rrbracket^1 + \llbracket P \rrbracket^2)F$, and assume they are unable to find a commitment or hash collision. Then, the probability that they are accepted by an honest verifier $\mathcal{V}$ is at most $\varepsilon = p_f + (1 - p_f)/N^D$.*

Proof. Assume that an honest verifier accepts, even though the two provers committed to a bad witness as defined above. Let $(r, \epsilon), i^*$ be the challenges of the verifier. In this case, one of two things must happen:

1. The random values $\llbracket v \rrbracket$ were correctly computed in all in-the-Head states, and $v = 0$.
2. A faithful calculation leads to $v \neq 0$, so $\tilde{\mathcal{P}}$ cheat in the communications they send —at least one of $\llbracket \alpha \rrbracket$, $\llbracket \beta \rrbracket$, $\llbracket v \rrbracket$ is incorrectly computed in an in-the-Head party.

In the first case, which occurs with probability p_f, the provers commit polynomials such that $(\llbracket S \rrbracket^1 + \llbracket S \rrbracket^2) \cdot (\llbracket Q \rrbracket^1 + \llbracket Q \rrbracket^2) \neq (\llbracket P \rrbracket^1 + \llbracket P \rrbracket^2)F$, but with

$$\delta = \left(\left(\llbracket S \rrbracket^1 + \llbracket S \rrbracket^2 \right) \cdot \left(\llbracket Q \rrbracket^1 + \llbracket Q \rrbracket^2 \right) - \left(\llbracket P \rrbracket^1 + \llbracket P \rrbracket^2 \right) F \right) (r)$$

equal to zero, or the Beaver triplet committed in round 1 must satisfy

$$(\llbracket c \rrbracket^1 + \llbracket c \rrbracket^2) - (\llbracket a \rrbracket^1 + \llbracket a \rrbracket^2)(\llbracket b \rrbracket^1 + \llbracket b \rrbracket^2) = \epsilon \cdot \delta.$$

In the second case, which occurs with probability $1 - p$, in the point r

$$(S^1 + S^2) \cdot (Q^1 + Q^2)(r_i) \neq (P^1 + P^2)F(r),$$

and

$$v = c - a \cdot b - \epsilon\delta$$

is nonzero. Correctly computed communications $\llbracket \alpha \rrbracket, \llbracket \beta \rrbracket, \llbracket v \rrbracket$ would fail the verifier's checks.

However, in each dimension $k \in [D]$, the opened communications $\llbracket \boldsymbol{\alpha} \rrbracket_k, \llbracket \boldsymbol{\beta} \rrbracket_k, \llbracket \boldsymbol{v} \rrbracket_k$ are verified in all-but-one of the N states. If either prover cheats in two main party communications in any dimension, the verifier immediately rejects with probability 1. Therefore, for a prover—having committed to a bad witness at some point r_i—to have any chance of passing the verifier's checks, they must cheat in exactly one main party per dimension.

This means the cheating prover must "guess" the correct main party index to cheat in, across D independent runs.

Since each main party state is the sum of N^{D-1} leaf party shares, and all but one of those shares will be opened, the only viable cheating strategy is to add $\delta \neq 0$ to one of the leaf party communications (either $\llbracket \alpha \rrbracket, \llbracket \beta \rrbracket$, or $\llbracket v \rrbracket$), and then aggregate honestly at the main party level. To cheat across D main parties with indices $i_1, \ldots, i_D$, the prover adds δ to the leaf party at index $\boldsymbol{i}^* = (i_1, \ldots, i_D)$.

This strategy only succeeds if the verifier's second challenge picks exactly this leaf party index $\boldsymbol{i}^*$, which occurs with probability $\frac{1}{N^D}$. □

Theorem 2 (Knowledge Soundness)**.** *If efficient provers $\mathcal{P}_1, \mathcal{P}_2$ can convince the verifier V with probability*

$$\tilde{\varepsilon} = \Pr[\langle \mathcal{P}_1, \mathcal{P}_2, V \rangle = 1] > \varepsilon = p_f + (1 - p_f)\frac{1}{N^D},$$

where p_f is bounded in Equation 2, then there exists a polynomial time simulator $\mathcal{S}$ that produces a commitment collision or extracts a good witness $\mathbf{x}$ such that $\mathbf{Hx} = y$ and $wt(x) < \omega$.

Proof. First, we will argue why two accepting transcripts with the same initial commitment but a different second challenge will, with high probability, allow for the extraction of a good witness. Recall that a single transcript contains

$$\{\mathtt{COM}^p, h^p, \{\mathtt{state}_i^p\}_{i \neq \mathbf{i}^*}, \mathtt{com}_{\mathbf{i}^*}^p, [\![\boldsymbol{\alpha}]\!]_{\mathbf{i}^*}^p, [\![\boldsymbol{\beta}]\!]_{\mathbf{i}^*}^p, [\![\boldsymbol{v}]\!]_{\mathbf{i}^*}^p\}_{p \in \{1,2\}}, (r, \epsilon), \mathbf{i}^*$$

Denote the two transcripts by T_1, T_2.

Assume that the commitment scheme is perfectly binding. For two sets of transcripts with the same initial commitments, but for two different second challenges $i^* \neq j^*$. There are two distinct cases to consider:

- At least one of the provers committed to a different witness $[\![\mathbf{x}]\!]^p$, $[\![\mathbf{S}]\!]^p$, $[\![\mathbf{Q}]\!]^p$, $[\![\mathbf{P}]\!]^p$ and we have found a collision in the commitment scheme.
- Both provers committed to the same witnesses, e.g $[\![\mathbf{x}]\!]^p$, $[\![\mathbf{S}]\!]^p$, $[\![\mathbf{Q}]\!]^p$, $[\![\mathbf{P}]\!]^p$ (and their particular splits) are equal for $p \in \{1, 2\}$ in both transcripts.

In the second case, and when the second challenge differs, a witness can be extracted. To show this, consider unopened leaf party indices $i^* \neq j^*$. In the first transcript, the verifier learns all $N^D - 1$ leaf party shares which are not i^*, and in the second transcript all leaf party shares which are not i^*. Therefore, with both transcripts, the verifier knows all the witness shares and can reconstruct the full witness.

Extracting the Shared Witness. Let us make this witness reconstruction explicit. Let two accepting transcripts T_1, T_2 have a different second challenge. Consider the hypercube indices $\boldsymbol{i}^* \neq \boldsymbol{j}^*$ meaning $(i_1^*, \ldots, i_D^*) \neq (j_1^*, \ldots, j_D^*)$. There exists at least one dimension in which the coordinates are not equal. Let the first index where they differ be k such that $i_k^* \neq j_k^*$. Then in the k-th dimension, the extractor can sum up all the N main party shares to obtain $[\![\boldsymbol{x}]\!]^1, [\![\boldsymbol{x}]\!]^2$. That is, the extractor can, for all $i \neq i^*$ calculate main party shares $[\![\boldsymbol{x}]\!]_{k,i}^p$ by simple aggregation, then calculate $[\![\boldsymbol{x}]\!]_{k,i_k^*}^p$

$$[\![\boldsymbol{x}]\!]_{k,i_k^*}^p = \sum_{i \neq i^*} [\![\boldsymbol{x}]\!]_i^p + [\![\boldsymbol{x}]\!]_{i^*}^p,$$

where $[\![\boldsymbol{x}]\!]_{i^*}$ is obtained from the second transcript. Notice that all witness shares can be extracted. In the following we will show that if the prover algorithms generate two accepting transcripts with the same initial commitment but different challenges, the witness is good with high probability. Call $[\![\boldsymbol{x}]\!]^1, [\![\boldsymbol{x}]\!]^2, [\![\boldsymbol{Q}]\!]^1, [\![\boldsymbol{Q}]\!]^2, [\![\boldsymbol{P}]\!]^1, [\![\boldsymbol{P}]\!]^2$ a good witness if

$$([\![S]\!]^1 + [\![S]\!]^2) \cdot ([\![Q]\!]^1 + [\![Q]\!]^2) = ([\![P]\!]^1 + [\![P]\!]^2)F.$$

Denote by R_h^p the random variable that party p uses to generate the initial commitment with r_h^p being its value. The extractor works by running $\tilde{\mathcal{P}} \coloneqq (P_1, P_2)$ with an honest $\mathcal{V}$

until a successful transcript T_1 is found. Then $\tilde{\mathcal{P}}$ is rewound using the same randomness r_h^1, r_h^2 until a second accepting transcript T_2 with a different second challenge is found. Extract the witness from the two accepting transcripts. If the witness is bad, start over.

Now, we will argue why the extractor makes a polynomial number of queries to $\tilde{\mathcal{P}}$. Recall that the provers $\tilde{\mathcal{P}}$ can make the verifier accept with probability $\tilde{\varepsilon} > \varepsilon$ where $\varepsilon = p_f + (1 + p_f)\frac{1}{N^D}$. Then there exists $\alpha \in (0, 1)$ such that $(1 - \alpha) \cdot \tilde{\varepsilon} > \varepsilon$. We say (r_h^1, r_h^2) is *good* if $\Pr[\tilde{\mathcal{P}}$ are successful $\mid (r_h^1, r_h^2)] \geq (1 - \alpha) \cdot \tilde{\varepsilon}$.

Now consider the splitting lemma with X being the set of all possible values for the pairs r_h^1, r_h^2 and Y being the set of all other randomness used for the protocol. Denote that joint randomness by t_h. Define $A \subset X \times Y$ by

$$A = \{(r_h^1, r_h^2), t_h \mid \text{Using randomness } (r_h^1, r_h^2), t_h \text{ results in an accepting transcript}\}.$$

Then, by our assumption that $\tilde{\mathcal{P}}$ can produce accepting transcripts with probability $\tilde{\varepsilon}$, we know that $\Pr[(r_h^1, r_h^2), t_h \in A] \geq \tilde{\varepsilon}$. Following the notations of Lemma 1, let $B \subset A$ be the set defined by of good randomness. By the splitting lemma: $\Pr[(x, y) \in B \mid (x, y) \in A] \geq \alpha$, which is to say $\Pr[(r_h^1, r_h^2)$ is good $\mid$ $\tilde{\mathcal{P}}$ are successful$] \geq \alpha$. This means that a good randomness can be found by collecting approximately $\frac{1}{\alpha}$ accepting transcripts.

Now apply Lemma 4 - when (r_h^1, r_h^2) is good, the provers are accepted with a high probability $(1 - \alpha) \cdot \tilde{\varepsilon} > \varepsilon$, the initial commitment in the the transcript encode a good witness, which can be extracted. Having obtained a good transcript T_1, we will now provide a lower bound on the number of rewinds in the inner loop to find another good transcript T_2 with the same randomness (r_h^1, r_h^2) such that $i^* \neq j^*$

$$\begin{aligned}
\Pr[\text{succ}_{\tilde{\mathcal{P}}} \cap i^* \neq j^* \mid (r_h^1, r_h^2) \text{ good}] &= \Pr[\text{succ}_{\tilde{\mathcal{P}}} \mid (r_h^1, r_h^2) \text{ good}] \\
&\quad - \Pr[\text{succ}_{\tilde{\mathcal{P}}} \cap i^* = j^* \mid (r_h^1, r_h^2) \text{ good}] \\
&\geq \Pr[\text{succ}_{\tilde{\mathcal{P}}} \mid (r_h^1, r_h^2) \text{ good}] - \frac{1}{N^D} \\
&\geq (1 - \alpha)\tilde{\epsilon} - \frac{1}{N^D} \\
&\geq (1 - \alpha)\tilde{\epsilon} - \epsilon.
\end{aligned}$$

Recall the approximation $\ln(1 - x) \overset{x \to 0}{\sim} -x$. To estimate L, the number of repetitions necessary for a $\frac{1}{2}$ probability of finding T_2, consider the following inequality:

$$\begin{aligned}
(1 - (1 - \alpha)\tilde{\epsilon} - \epsilon)^L \leq \frac{1}{2} &\Leftrightarrow L \leq \frac{\ln(\frac{1}{2})}{\ln(1 - (1 - \alpha)\tilde{\epsilon} - \epsilon)} \\
&\Leftrightarrow L \leq \frac{-\ln 2}{-((1 - \alpha)\tilde{\epsilon} - \epsilon)} \\
&\Leftrightarrow L \geq \frac{\ln 2}{(1 - \alpha)\tilde{\epsilon} - \epsilon}
\end{aligned}$$

Denote the expected number of calls to $\tilde{\mathcal{P}}$ by $\mathbb{E}(\tilde{\mathcal{P}})$. Then $\mathbb{E}(\tilde{\mathcal{P}})$ can be expressed as a recursive formula. First, it is a function of the probability of finding the initial accepting transcript T_1. After that is found, we have to estimate take into account the probability of obtaining T_2 with L calls to $\tilde{\mathcal{P}}$. That is dependent on whether the randomness from the first transcript is good. Step-by-step, the extractor works as follows:

1. Make a call to $\tilde{\mathcal{P}}$ to interact with an honest verifier.

2. If accepting transcript is not found, with probability $(1 - \Pr[\text{succ}_{\tilde{\mathcal{P}}}] = 1 - \tilde{\varepsilon})$, go to Step 1.

3. If T_1 is accepting, then r_h is good with probability α by the splitting lemma. Then, make L calls to $\tilde{\mathcal{P}}$, after which there is at least $\frac{1}{2}$ probability of success. If another accepting transcript is found, terminate. Else, return to Step 1.

4. If r_h is bad, which we don't yet know and cannot directly check, still make L calls to $\tilde{\mathcal{P}}$. In this case, there aren't any guarantees on the probability of finding T_2. If another accepting transcript is found, terminate. Else, return to Step 1.

To summarize - we repeat Step 1 until a accepting transcript T_1 is found. Then, we make another L calls to $\tilde{\mathcal{P}}$ with the randomness r_h^1, r_h^2. If r_h^1, r_h^2 is good, with probability α, we will find T_2 with probability $\frac{1}{2}$. If r_h^1, r_h^2 is bad, with probability $1 - \alpha$, there is no guarantee about finding T_2. We will assume that T_2 is never found when the randomness is bad. Therefore

$$\begin{aligned}\Pr[T_2 \text{ not found} \mid \text{succ}_{\tilde{\mathcal{P}}}] =& \Pr[T_2 \text{ not found} \mid r_h^1, r_h^2 \text{ is good} \wedge \text{succ}_{\tilde{\mathcal{P}}}] + \\ & \Pr[T_2 \text{ not found} \mid r_h^1, r_h^2 \text{ is bad} \wedge \text{succ}_{\tilde{\mathcal{P}}}] \\ =& \alpha/2 + (1 - \alpha) \\ =& 1 - \alpha/2.\end{aligned}$$

Continuing the analysis, we now derive an upper bound on the expected number of iterations, denoted by $\mathbb{E}(\tilde{\mathcal{P}})$. This expectation is easily expressed by considering whether a good transcript is found in the first phase.

$$\mathbb{E}(\tilde{\mathcal{P}}) \leq 1 + \underbrace{(1 - \Pr[\text{succ}_{\tilde{\mathcal{P}}}])\mathbb{E}(\tilde{\mathcal{P}})}_{\text{Do not find } T_1} + \underbrace{\Pr[\text{succ}_{\tilde{\mathcal{P}}}] \left(L + \left(1 - \frac{\alpha}{2}\right) \mathbb{E}(\tilde{\mathcal{P}})\right)}_{\text{Find } T_1},$$

which simplifies to:

$$\mathbb{E}(\tilde{\mathcal{P}}) \leq \frac{2}{\alpha\tilde{\epsilon}} (1 + \tilde{\epsilon}L) = \frac{2}{\alpha\tilde{\epsilon}} \left(1 + \frac{\tilde{\epsilon} \ln 2}{(1 - \alpha)\tilde{\epsilon} - \epsilon}\right).$$

To express the bound in terms of ϵ and $\tilde{\epsilon}$ only, define

$$(1 - \alpha)\tilde{\epsilon} = \frac{1}{2}(\epsilon + \tilde{\epsilon}),$$

i.e., the midpoint between ϵ and $\tilde{\epsilon}$. Substituting this into the previous expression leads to the more compact upper bound:

$$\mathbb{E}(\tilde{\mathcal{P}}) \leq \frac{4}{\tilde{\epsilon} - \epsilon} \left(1 + \frac{2\tilde{\epsilon} \ln 2}{\tilde{\epsilon} - \epsilon}\right).$$

□

4.5 Witness Privacy

In this section, we will formalize the notion that an execution of Algorithm 1 does not leak information about the witness. To this end, we are following the definition of *t-Witness Privacy* from definition 11 with $t = 1$. 1-Witness Privacy (WP) requires that even when one prover and the verifier are controlled by an adversary, nothing can be learned about

the honest provers' witness. To show that Algorithm 1 is indeed WP, we must, for any PPT adversary $\mathcal{A}$, construct a WP simulator $\mathcal{S}$ that produces views of $\mathcal{A}$ that are indistinguishable from those of the adversary in a real execution of the protocol. W.L.O.G assume that $\mathcal{I} = \{P_2, V\}$.

Aa single honest prover cannot force the verifier to accept when the other prover is malicious. Therefore, the simulator $\mathcal{S}$ simply has to match the behavior of $\mathcal{A}$. If $\mathcal{A}$ is honestly executing the protocol, then $\mathcal{S}$ must be able to produce an accepting transcript. If $\mathcal{A}$ deviates from an honest execution, $\mathcal{S}$ has to produce transcripts which are rejected by the verifier.

Recall that the simulator $\mathcal{S}$ has access to everything the adversary knows, including its private input $[\![\boldsymbol{S}]\!]^2, [\![\boldsymbol{Q}]\!]^2, [\![\boldsymbol{P}]\!]^2, [\![\boldsymbol{x}]\!]^2, [\![a]\!]^2, [\![b]\!]^2, [\![c]\!]^2$. This is also made explicit in the t-zero-knowledge definition in [LZW+24].

The exact strategy used in the HVZK simulator cannot be replicated here, because even though in the semi-honest verifier case, when the simulator can know the challenge values beforehand, the concrete states of in-the-Head parties of $\mathcal{P}_1$ are unknown. To get around this, recall that the function of this protocol is to securely evaluate $v = (([\![S]\!]^1 + [\![S]\!]^2) \cdot ([\![Q]\!]^1 + [\![Q]\!]^2)) - ([\![P]\!]^1 + [\![P]\!]^2) \cdot F)(r)$ to test that it is the zero polynomial. In particular, if $\mathcal{A}$ follows the computations described, adding all shares of $[\![v]\!]^2$ gives the following:

$$[\![v]\!]^2 = [\![c]\!]^2 + \varepsilon \cdot F(r) \cdot [\![P]\!]^2(r) + \alpha \cdot [\![b]\!]^2 + \beta \cdot [\![a]\!]^2 - \alpha \cdot \beta,$$

where $\alpha = [\![\alpha]\!]^1 + [\![\alpha]\!]^2$ and $\beta = [\![\beta]\!]^1 + [\![\beta]\!]^2$. Notice that this value is independent of the particular states of any specific states of in-the-Head parties and therefore can be computed by $\mathcal{S}$ after receiving $[\![\alpha]\!]^2, [\![\beta]\!]^2$ from $\mathcal{A}$ and simulating $[\![\alpha]\!]^1, [\![\beta]\!]^1$. Since $v = 0$ in an honest execution and $[\![v]\!]^1 + [\![v]\!]^2 = v$, the simulator can, while impersonating $\mathcal{P}_1$, ensure that the in-the-head shares of $[\![v]\!]^1$ sum up to $-[\![v]\!]^2$, given that the adversary acts honestly.

Theorem 3 (1-Witness Privacy (WP))**.** *If the outputs of PRG and commitment Com are indistinguishable from the uniform random distribution, then Algorithm 1 is 1-witness private with passive verifier corruption as defined in Definition 11.*

Algorithm 4. **WP Simulator**$([\![\boldsymbol{S}]\!]^2, [\![\boldsymbol{Q}]\!]^2, [\![\boldsymbol{P}]\!]^2, [\![\boldsymbol{x}]\!]^2, [\![a]\!]^2, [\![b]\!]^2, [\![c]\!]^2)$

Step 0 (Sample seed):
Sample $\mathsf{seed}^1 \xleftarrow{\$} \{0,1\}^\lambda$
Generate $(\mathsf{seed}^1_{i'}, \rho^1_{i'})$ *for all leaf parties via* TreePRG*(seed)*

Step 1 (Sample Challenges):
$\mathsf{CH1} = \{r, \epsilon\} \leftarrow \mathbb{F}^t_{\mathsf{points}}, \quad \mathsf{CH2} = i^* \leftarrow [1, \ldots, N^D]$

Step 2 (Generate Leaf Party States):
Expand root seeds recursively via TreePRG *to get* N^D *leaf states* $\{\mathsf{state}_i\}^1_{i \in [N^D]}$.

Step 3 (Generate Leaf Party Commitments and Witness Shares):
for *each* $i' \neq i^*$ ***do***
Compute $\mathsf{com}^1_{i'} \leftarrow \mathsf{Hash}(\mathsf{state}^1_{i'}, \rho^1_{i'})$
if $i' \neq N^D$ ***then***
Expand the leaf party seeds into witness shares
else

Randomly draw:
$[\![\mathbf{x}_A]\!]^1_{N^D}$, $[\![\mathbf{Q}]\!]^1_{N^D}$, $[\![\mathbf{P}]\!]^1_{N^D}$, $[\![\mathbf{a}]\!]^1_{N^D}$, $[\![\mathbf{b}]\!]^1_{N^D}$, $[\![\mathbf{c}]\!]^1_{N^D}$
end if
end for
for $i' = i^*$ ***do***
Draw $\mathsf{com}^1_{i^*}$ *at random*
end for
Compute initial commitment:
$\mathsf{COM}^1 = \mathsf{Hash}(\mathsf{com}^1_1, \ldots, \mathsf{com}^1_{i^*}, \ldots, \mathsf{com}^1_{N^D})$
Send COM^1 *to* $\mathcal{A}$.
Receive COM^2 *from* $\mathcal{A}$.

Step 4 (Generate In-the-Head Party Communications):
Draw $[\![\alpha]\!]^1_{i^*}$, $[\![\beta]\!]^1_{i^*}$ *uniformly at random*
for $k \in \{1, \ldots, D\}$ ***do***
for $i \neq i^*_k$ ***do***
Calculate and accumulate communication shares $[\![\alpha]\!]^1_{k,i}$, $[\![\beta]\!]^1_{k,i}$.
end for
Compute $[\![\alpha]\!]_{k,i^*_k}$, $[\![\beta]\!]_{k,i^*_k}$ *using* $[\![\alpha]\!]^1_{i^*}$, $[\![\beta]\!]^1_{i^*}$.
if $k = 1$ ***then***
Send $[\![\alpha]\!]^1, [\![\beta]\!]^1$ *to* $\mathcal{A}$.
Receive $[\![\alpha]\!]^2, [\![\beta]\!]^2$ *from* $\mathcal{A}$.
Set $\alpha = [\![\alpha]\!]^1 + [\![\alpha]\!]^2$.
end if
for $i \in [N]$ ***do***
if $i \neq i^*_k$ ***then***

$$[\![v]\!]^1_{k,i} = -[\![c]\!]^1_{k,i} + \langle \epsilon, \mathbf{F}(r) \cdot [\![\boldsymbol{P}(r)]\!]^1_{k,i} \rangle + \langle \alpha, [\![b]\!]^1_{k,i} \rangle + \langle \beta, [\![a]\!]^1_{k,i} \rangle - \langle \alpha, \beta \rangle.$$

end if
end for
Set
$$[\![v]\!]^2 = [\![c]\!]^2 + \varepsilon \cdot F(r) \cdot P^2(r) + \alpha \cdot [\![b]\!]^2 + \beta \cdot [\![a]\!]^2 - \alpha \cdot \beta.$$
Set $[\![v]\!]^1_{k,i^*_k} = -[\![v]\!]^2 - \sum_{i=1, i\neq i^*_k}^{N} [\![v]\!]^1_{k,i}$
Set $H^1_k = \mathsf{Hash}([\![\boldsymbol{\alpha}]\!]^1_k, [\![\boldsymbol{\beta}]\!]^1_k, [\![\boldsymbol{v}]\!]^1_k)$.
end for
Send $h^1 = \mathsf{Hash}(H^1_1, \ldots H^1_D)$ *to* $\mathcal{A}$.
Receive h^2 *from* $\mathcal{A}$.

Step 5 (Open views):
Send $\{\mathsf{state}^1_i\}_{i\neq i^*}) \| \mathsf{com}^1_{i^*} \| [\![\alpha]\!]^1_{i^*} \| [\![\beta]\!]^1_{i^*} \| [\![v]\!]^1_{i^*}$ *to* $\mathcal{A}$.
Receive $\mathsf{RSP}2^2$ *from* $\mathcal{A}$.
Output $\{\mathsf{COM}^p, h^p, \{\mathsf{state}^p_i\}_{i\neq \mathbf{i}^*}, \mathsf{com}^p_{\mathbf{i}^*}, [\![\boldsymbol{\alpha}]\!]^p_{\mathbf{i}^*}, [\![\boldsymbol{\beta}]\!]^p_{\mathbf{i}^*}, [\![\boldsymbol{v}]\!]^p_{\mathbf{i}^*}\}_{p\in\{1,2\}}$, (r, ϵ), $\mathbf{i}^*$

Proof Sketch. To see that transcripts produced by the WP Simulator are computationally indistinguishable from a real execution, first consider the "honest adversary" scenario, where $\mathcal{A}$, given their witness shares, performs all computations honestly. Then, the transcript is accepting with probability 1 by construction, and the messages are $(t, \epsilon_{\mathsf{PRG}} + \epsilon_{\mathsf{COM}})$

indistinguishable as shown in the HVZK simulator.

In the case where the adversary deviates from the protocol in any way, and are unable to find a commitment collision, the success probability is bounded by Lemma 4. □

5 Two Party Signatures From SDitH

We obtain a two-party signature scheme by applying the Fiat-Shamir transform to Algorithm 1. To describe a signature scheme, we will describe the of multi-party algorithms KeyGen, Sign, Verify. The signing algorithm and the security proof are closely following the work of Aguilar-Melchor *et al.* [AMGH+23].

Key Generation. For simplicity, we will delegate key generation to a trusted party. The trusted party samples $H \xleftarrow{\$} \mathbb{F}_{SD}^{(m-k)\times m}$ and low weight $[\![\boldsymbol{x}]\!]^1, [\![\boldsymbol{x}]\!]^2 \in \mathbb{F}_{SD}^{m-k}$ and sends them to signers $\mathcal{P}_1, \mathcal{P}_2$ respectively. The public key is set as $y \coloneqq ([\![\boldsymbol{x}]\!]^1 + [\![\boldsymbol{x}]\!]^2)H$. By carefully selecting the field $\mathbb{F}_{SD}$ and the weight parameter ω, we can ensure that syndrome decoding is sufficiently difficult for partial syndrome decoding instances $[\![\boldsymbol{x}]\!]^1 H$, $[\![\boldsymbol{x}]\!]^2 H$ as well as the full instance $([\![\boldsymbol{x}]\!]^1 + [\![\boldsymbol{x}]\!]^2)H$.

5.1 Signing Algorithm

The signing algorithm is obtained directly by replacing verifiers challenges in 1 by calls to a hash function. For our security proof, we will model hash functions in the random oracle model (ROM). In ROM, hash functions are modeled as truly random deterministic functions. To simplify estimating the probability of input collisions to hashes in the security analysis, both provers also include a salt in their hashes. Additionally, the signature scheme is parametrized by τ - number of parallel SDitH executions and t - the number of challenge points in verifiers challenge 1. Each signature uses $t \cdot \tau$ Beaver triples.

At this point, it is necessary to consider the Kales and Zaverucha attack [KZ20] on Fiat-Shamir transformed schemes with 5 or more rounds [AMGH+23]. The soundness of the non-interactive scheme is lowered because in order to pass a verifiers checks, only one of the transformed challenges has to be guessed.

The cost of a forgery relies on finding a repetition rate τ' that minimizes the cost of a forgery:

$$\text{cost}_{\text{forge}} \coloneqq \min_{0 \le \tau' \le \tau} \left\{ \frac{1}{\sum_{i=\tau'}^{\tau} \binom{\tau}{i} p_f^i (1-p)^{\tau - i}} + (N^D)^{\tau - \tau'} \right\}. \tag{3}$$

Algorithm 5. **Sign.** Prover $\mathcal{P}_p$ knows $[\![\boldsymbol{S}]\!]^p, [\![\boldsymbol{Q}]\!]^p, [\![\boldsymbol{P}]\!]^p, [\![a]\!]^p, [\![a]\!]^p, [\![b]\!]^p, [\![c]\!]^p$

Both parties sample a salt $\mathtt{salt}^p \xleftarrow{\$} \{0,1\}^{2\lambda}$.
Round 1.
for party $p \in \{1,2\}$ repetition $e \in [\tau]$ **do**
  Sample a root seed: $\mathtt{seed}^{e,p} \xleftarrow{\$} \{0,1\}^{\lambda}$
  Derive leaf seeds $\mathtt{seed}_{i'}^{e,p}$ using TreePRG.
  for each leaf $i' = 1$ to N^D **do**
    if $i' \neq N^D$ **then**
      $\mathtt{state}_{i'} = \{[\![c]\!]_{i'}^{e,p}, [\![\boldsymbol{x}_A]\!]_{i'}^{e,p}, [\![\boldsymbol{Q}]\!]_{i'}^{e,p}, [\![\boldsymbol{P}]\!]_{i'}^{e,p}\} \leftarrow \mathtt{PRG}(\mathtt{seed}_{i'})$
    else
      $[\![\boldsymbol{x}_A]\!]_{N^D}^{e,p} = [\![\boldsymbol{x}_A]\!]^{e,p} - \sum_{i=1}^{N^D-1} [\![\boldsymbol{x}_A]\!]_i^{e,p}$
      $[\![\boldsymbol{Q}]\!]_{N^D}^{e,p} = [\![\boldsymbol{Q}]\!]^{e,p} - \sum_{i=1}^{N^D-1} [\![\boldsymbol{Q}]\!]_i^{e,p}$
      $[\![\boldsymbol{P}]\!]_{N^D}^{e,p} = [\![\boldsymbol{P}]\!]^{e,p} - \sum_{i=1}^{N^D-1} [\![\boldsymbol{P}]\!]_i^{e,p}$

$\mathsf{state}_{i'} = \{[\![a]\!]^{e,p}_{N^D}, [\![b]\!]^{e,p}_{N^D}, [\![c]\!]^{e,p}_{N^D}, [\![\boldsymbol{x}_A]\!]^{e,p}_{N^D}, [\![\boldsymbol{Q}]\!]^{e,p}_{N^D}, [\![\boldsymbol{P}]\!]^{e,p}_{N^D}\}$
end if
for each main party index p in $\{(1,i_1),(2,i_2),\ldots,(D,i_D)\}$ **do**
$[\![\boldsymbol{x}_A]\!]^{e,p}_{p} += [\![\boldsymbol{x}_A]\!]^{e,p}_{i'}, [\![\boldsymbol{Q}]\!]^{e,p}_{p} += [\![\boldsymbol{Q}]\!]^{e,p}_{i'}, [\![\boldsymbol{P}]\!]^{e,p}_{p} += [\![\boldsymbol{P}]\!]^{e,p}_{i'}$
$[\![a]\!]^{e,p}_{p} += [\![a]\!]^{e,p}_{i'}, [\![b]\!]^{e,p}_{p} += [\![b]\!]^{e,p}_{i'}, [\![c]\!]^{e,p}_{p} += [\![c]\!]^{e,p}_{i'}$
end for
$\mathsf{com}^{e,p}_{i'} = \mathsf{Hash}_0(\mathsf{salt}^1, \mathsf{salt}^2, e, i', \mathsf{state}_{i'})$.
end for
$\mathsf{COM}^{e,p}_1 = \mathsf{Hash}_1(\mathsf{salt}^1, \mathsf{salt}^2, e, \mathsf{com}^{e,p}_1, \ldots, \mathsf{com}^{e,p}_{N^D})$
Broadcast $\mathsf{COM}^p_1, \mathsf{salt}^p$.
end for

Round 2. Both parties
Calculate $h_2 = \mathsf{Hash}_2(m, \mathsf{salt}^1, \mathsf{salt}^2, \mathsf{COM}^{1,1}_1, \ldots, \mathsf{COM}^{1,\tau}_1, \mathsf{COM}^{2,1}_1, \ldots, \mathsf{COM}^{2,\tau}_1)$
Expand h_2 to τ challenges. $\{\boldsymbol{r}_e, \boldsymbol{\epsilon}_e\}_{e\in[\tau]} \leftarrow \mathsf{PRG}(h_2)$, where $(\boldsymbol{r}_e, \boldsymbol{\epsilon}_e) \in \mathbb{F}^t_{points} \times \mathbb{F}^t_{points}$.

Round 3.
for party $p \in \{1,2\}$, repetition $e \in [\tau]$ **do**
for dimension $k \in [D]$ **do**
for main party index $i \in [N]$ **do**
Set $[\![\boldsymbol{\alpha}]\!]^{e,p}_{k,i} = \boldsymbol{\epsilon}[\![\boldsymbol{Q}(r)]\!]^{e,p}_{k,i} + [\![\boldsymbol{a}]\!]^{e,p}_{k,i}$.
Set $[\![\boldsymbol{\beta}]\!]^{e,p}_{k,i} = [\![\boldsymbol{S}(r_e)]\!]^{e,p}_{k,i} + [\![\boldsymbol{b}]\!]^{e,p}_{k,i}$.
end for
Broadcast $[\![\boldsymbol{\alpha}]\!]^{e,p}, [\![\boldsymbol{\beta}]\!]^{e,p}$.
Set $\boldsymbol{\alpha}^e = \boldsymbol{\alpha}^{e,1} + \boldsymbol{\alpha}^{e,2}$.
for main party index $i \in [N]$ **do**

$$[\![\boldsymbol{v}]\!]^{e,p}_{k,i} = -[\![\boldsymbol{c}]\!]^{e,p}_{k,i} + \langle \boldsymbol{\epsilon_e}, \mathbf{F}(\boldsymbol{r}_e) \cdot [\![\boldsymbol{P}(\boldsymbol{r}_e)]\!]^{e,p}_{k,i} \rangle + \langle [\![\boldsymbol{\alpha}]\!]^e, [\![\boldsymbol{b}]\!]^{e,p}_{k,i} \rangle + \langle [\![\boldsymbol{\beta}]\!]^e, [\![\boldsymbol{a}]\!]^{e,p}_{k,i} \rangle - \langle [\![\boldsymbol{\alpha}]\!]^e, [\![\boldsymbol{\beta}]\!]^e \rangle.$$

end for
end for
Set $H^{e,p}_k = \mathsf{Hash}_3(\mathsf{salt}^1, \mathsf{salt}^2, e, p, [\![\boldsymbol{\alpha}]\!]^{e,p}_k, [\![\boldsymbol{\beta}]\!]^{e,p}_k, [\![\boldsymbol{v}]\!]^{e,p}_k)$.
Broadcast $h^p_4 = \mathsf{Hash}_4(m, \mathsf{salt}^1, \mathsf{salt}^2, p, H^{1,p}_1, \ldots H^{\tau,p}_D)$.
end for

Round 4. Both parties
Compute $h_4 = \mathsf{Hash}_5(m, \mathsf{salt}^1, \mathsf{salt}^2, h^1_4 \mid h^2_4)$.
Expand h_4 to τ challenge indices. $\{i^{e,*}\}_{e\in[\tau]} \leftarrow \mathsf{PRG}(h_4)$.

Round 5. Party p broadcasts
$\left(\mathsf{state}^{e,p}_{i\neq i^*} \mid \mathsf{com}^{e,p}_{i^*} \mid \{[\![\boldsymbol{\alpha}]\!]^{e,p}_{i^{e,*}}, [\![\boldsymbol{\beta}]\!]^{e,p}_{i^{e,*}}, [\![\boldsymbol{v}]\!]^{e,p}_{i^{e,*}}\}\right)_{e\in[\tau]}$.
$\sigma = \mathsf{salt}^1 \mid \mathsf{salt}^2 \mid h_2 \mid h_4 \mid \left(\{\mathsf{state}_i\}^{e,p}_{i\neq i^*} \mid \mathsf{com}^{e,p}_{i^*} \mid \{[\![\boldsymbol{\alpha}]\!]^{e,p}, [\![\boldsymbol{\beta}]\!]^{e,p}, [\![\boldsymbol{v}]\!]^{e,p}\}\right)_{e\in[\tau],p\in\{1,2\}}$

Given: Input σ

Algorithm 6. **Verify**

Parse $\sigma = \mathtt{salt}^1 \mid \mathtt{salt}^2 \mid h_2 \mid h_4 \mid \left(\mathtt{state}^{e,p}_{i\neq i^*} \mid \mathtt{com}^{e,p}_{i^*} \mid \{[\![\boldsymbol{\alpha}]\!]^{e,p}, [\![\boldsymbol{\beta}]\!]^{e,p}, [\![\boldsymbol{v}]\!]^{e,p}\}\right)_{e\in[\tau],p\in\{1,2\}}$
Expand states to $([\![\boldsymbol{S}]\!]^{e,p}_i, [\![\boldsymbol{Q}]\!]^{e,p}_i, [\![\boldsymbol{P}]\!]^{e,p}_i, [\![\boldsymbol{a}]\!]^{e,p}_i, [\![\boldsymbol{b}]\!]^{e,p}_i, [\![\boldsymbol{c}]\!]^{e,p}_i)_{i\in[N^D], i\neq i^*, e\in[\tau], p\in\{1,2\}}$
for $i \neq i^*, p \in \{1,2\}, e \in [\tau]$ **do**
  $\mathtt{com}^{e,p}_i = \mathtt{Hash}_0(\mathtt{salt}^1, \mathtt{salt}^2, e, p, i', \mathtt{state}_{i'})$.
end for
for $p \in \{1,2\}, e \in [\tau]$ **do**
  Calculate $\mathtt{COM}^p_1 \neq \mathtt{Hash}_1(\mathtt{salt}^1, \mathtt{salt}^2, e, p, \mathtt{com}^{1,p}_1, \ldots, \mathtt{com}^{\tau,p}_{N^D})$
end for

Calculate $h'_2 = \mathtt{Hash}_2(m, \mathtt{salt}^1, \mathtt{salt}^2, \mathtt{COM}^{1,1}_1, \ldots, \mathtt{COM}^{1,\tau}_1, \mathtt{COM}^{2,1}_1, \ldots, \mathtt{COM}^{2,\tau}_1)$
Expand hash $\{\boldsymbol{r}_e, \boldsymbol{\epsilon}_e\}_{e\in[\tau]} \leftarrow \mathtt{PRG}(h_2)$.

for party $p \in \{1,2\}$, repetition $e \in [\tau]$ **do**
  for dimension $k \in [D]$ **do**
    for main party index $i \in [N], i \neq i^*_k$ **do**
      Set $[\![\boldsymbol{\alpha}]\!]^{e,p}_{k,i} = \boldsymbol{\epsilon}[\![\boldsymbol{Q}(r)]\!]^{e,p}_{k,i} + [\![\boldsymbol{a}]\!]^{e,p}_{k,i}$.
      Set $[\![\boldsymbol{\beta}]\!]^{e,p}_{k,i} = [\![\boldsymbol{S}(r)]\!]^{e,p}_{k,i} + [\![\boldsymbol{b}]\!]^{e,p}_{k,i}$.
    end for
    Calculate $[\![\boldsymbol{\alpha}]\!]^{e,p}_k, [\![\boldsymbol{\beta}]\!]^{e,p}_k$.
    Set $[\![\boldsymbol{\alpha}]\!]^{e}_k = [\![\boldsymbol{\alpha}]\!]^{e,1}_k + [\![\boldsymbol{\alpha}]\!]^{e,2}_k$.
    Set $[\![\boldsymbol{\beta}]\!]^{e}_k = [\![\boldsymbol{\beta}]\!]^{e,1}_k + [\![\boldsymbol{\beta}]\!]^{e,2}_k$.
    for main party index $i \in [N]$ **do**
      if $i^*_k = i$ **then** $[\![v]\!]^{e,p}_{k,i} = [\![\boldsymbol{v}]\!]^{e,p}_{i^*} + \sum^N_{j\in[N], j\neq i^*_k} [\![\boldsymbol{v}]\!]^{e,p}_{k,j}$
      else

$$[\![\boldsymbol{v}]\!]^{e,p}_{k,i} = -[\![\boldsymbol{c}]\!]^{e,p}_{k,i} + \langle \epsilon, \mathbf{F}(r) \cdot [\![\boldsymbol{P}(r)]\!]^{e,p}_{k,i} \rangle + \langle [\![\boldsymbol{\alpha}]\!]^{e}_k, [\![\boldsymbol{b}]\!]^{e,p}_{k,i} \rangle + \langle [\![\boldsymbol{\beta}]\!]^{e}_k, [\![\boldsymbol{a}]\!]^{e,p}_{k,i} \rangle - \langle [\![\boldsymbol{\alpha}]\!]^{e}_k, [\![\boldsymbol{\beta}]\!]^{e}_k \rangle.$$

      end if
    end for
    Set $\tilde{H}^{e,p}_k = \mathtt{Hash}_3([\![\boldsymbol{\alpha}]\!]^{e,p}_k, [\![\boldsymbol{\beta}]\!]^{e,p}_k, [\![\boldsymbol{v}]\!]^{e,p}_k)$.
  end for
  $h^p_4 = \mathtt{Hash}_4(m, \mathtt{salt}^1, \mathtt{salt}^2, p, H^{1,p}_1, \ldots H^{\tau,p}_D)$
end for
$h'_4 = \mathtt{Hash}_5(m, \mathtt{salt}^1, \mathtt{salt}^2, h^1_4 \mid h^2_4)$.

if for all $e \in [\tau], k \in [D]$ $\sum^N_{i=1, p\in\{1,2\}} [\![v]\!]^{e,p}_{k,i} = 0$ **and** $h'_2 = h_2$ **and** $h'_4 = h_4$ **then**
  ACCEPT
else
  REJECT
end if

5.2 Proof of Security

Theorem 4 (EU-CMA)**.** *The probability of an existential forgery under a chosen message attack of the two-party signatures described in Algorithm 5 is bounded from above by:*

$$\Pr(\mathit{forge}) \leq \frac{2 \cdot (q + \tau N^D q_S)^2}{2^\lambda} + \frac{q_S(q_S + 6q)}{2^\lambda} + \tau \cdot q_S \cdot \epsilon_{\mathsf{PRG}} + \epsilon_{SD} + q_2 \cdot p^{\tau'} + q_4 \cdot \left(\frac{1}{N^D}\right)^{\tau-\tau'}$$

Proof. The security proof closely follows the proof of security of the hypercube SDitH signatures by Aguilar-Melchor *et al.* [AMGH+23], which in turn was inspired by previous works such as Feneuil *et al.* [FJR22].

First, we will argue that we can effectively simulate the view of an adversary that controls one of the signing parties. To do this, we quantify the differences between an execution of the signing algorithm and the adversary interacting with the WP simulator. Assume that adversary is acting in place of $\mathcal{P}_2$.

Game 1 In this game, $\mathcal{A}$ interacts with an honest party to create a signature. This process is interactive with $\mathcal{A}$ being expected to send $\mathtt{salt}^2, \mathtt{COM}_1^2, \mathtt{COM}_2^2, \{\boldsymbol{\alpha}, \boldsymbol{\beta}\}_{e\in[\tau]}^e$ and, in the final round

$$\left(\mathtt{state}_{i\neq i^*}^{e,2} \mid \mathtt{com}_{i^*}^{e,2} \mid \left\{[\![\boldsymbol{\alpha}]\!]_{i^{e,*}}^{e,2}, [\![\boldsymbol{\beta}]\!]_{i^{e,*}}^{e,2}, [\![\boldsymbol{v}]\!]_{i^{e,*}}^{e,2}\right\}\right)_{e\in[\tau]}.$$

We aim to give an upper bound for $\Pr_1(\text{forge})$.

Game 2 In game 2, the protocol is exactly as game 1, except game 2 aborts if there is a hash collision in the outputs of $\mathsf{Hash}_0, \mathsf{Hash}_1, \mathsf{Hash}_3, \mathsf{Hash}_4$ for the honest party $\mathcal{P}_1$. The number of queries to $\mathsf{Hash}_0, \mathsf{Hash}_1, \mathsf{Hash}_3$ and Hash_4 are bounded from above by $q+\tau\cdot N^D\cdot q_S$, where $q = \max\{q_0, q_1, q_2, q_3, q_4, q_5\}$, q_i is the number of times Hash_i was queried, and q_S is the number of times game 2 is invoked. A single invocation contains $\tau \cdot N^D$ queries to Hash_0, τ queries to Hash_1. It also contains a single query to Hash_2, $\tau \cdot D$ queries to Hash_3 and one query to both Hash_4 and Hash_5. Recall the coarse birthday problem approximation: probability of a collision after k queries to 2λ-bit random function is $\frac{k}{2\cdot 2^\lambda}$. Utilizing this, the probability of a hash collision is bounded from above by

$$|\Pr_1(\text{forge}) - \Pr_2(\text{forge})| \leq \frac{4\cdot(q+\tau N^D q_S)^2}{2\cdot 2^\lambda}$$

Game 3 Game 3 differs from game 2 in that game 3 also aborts when an input query is repeated for any Hash_i. For an input query to be duplicated, at least the salt has to be the same. That means, that on the last signing query, the probability of an input collision for any of the hashes is at most $\frac{q_S-1+q_0+q_1+q_2+q_3+q_4+q_5}{2^\lambda}$. A coarse upper bound for an input collision on any of the hashes can therefore be given as $\frac{q_S(q_S+q_0+q_1+q_2+q_3+q_4+q_5)}{2^\lambda}$. Therefore,

$$|\Pr_2(\text{forge}) - \Pr_3(\text{forge})| \leq \frac{q_S(q_S+q_0+q_1+q_2+q_3+q_4+q_5)}{2^\lambda} \leq \frac{q_S(q_S+6q)}{2^\lambda}.$$

Game 4 In game 4, the hashes h_2, h_5 are replaced with uniform randomness. The verifiers challenges are still obtained by the pseudorandom expansion of h_2, h_5. In the random oracle model, hash functions differ from random sampling only in that they are deterministic. This difference is apparent only when a random oracle is queried with the same value twice. In this case game 3 aborts. Therefore,

$$\Pr_3(\text{forge}) = \Pr_4(\text{forge}).$$

Game 5 In game 5, we replace all $\mathtt{com}_{i^*}^{1,e}$ with uniform randomness for all $e \in [\tau]$. Because $\mathtt{com}_{i^*}^p = \mathsf{Hash}_0(\mathtt{salt}^1, \mathtt{salt}^2, e, p, i, \mathtt{state}_{i^*})$ includes an unique tuple of indices (e, i), which is unique within a signing query. In game 4, every (non-aborted) signing query and queries to $\mathsf{Hash}_0(\cdot)$ have a unique salt. Therefore, there are no duplicate inputs given to $\mathsf{Hash}_0(\cdot)$ and

$$\Pr_4(\text{forge}) = \Pr_5(\text{forge}).$$

Game 6 In game 6, we replace $\mathtt{COM}_1^1$ with uniform randomness. Similarly, this differs from 5 only if Hash_1 received a duplicate input. Because all the outputs of Hash_0, were made to be distinct in game 2, this is impossible. Therefore,

$$\Pr_5(\text{forge}) = \Pr_6(\text{forge}).$$

Game 7 In game 7, the witness shares are generated like in WP simulator, over which $\mathcal{A}$ has an advantage of at most ϵ_{PRG}. Taking into account the τ parallel runs and q_S queries gives the adversary an advantage of

$$|\Pr_6(\text{forge}) - \Pr_7(\text{forge})| \leq \tau \cdot q_S \cdot \epsilon_{\mathsf{PRG}}.$$

Game 8. In game 8, we abort if any execution of the protocol defines a correct witness. An execution $e^* \in [\tau]$ of a query to Hash_4

$$h_4 = \mathsf{Hash}_5(m, h_2, \mathtt{salt}^1, \mathtt{salt}^2, h_4^1 \mid h_4^2)$$

is said to define a correct witness if the following criteria are satisfied:

- $h_4^1 = \mathsf{Hash}_4(m, \mathtt{salt}^1, \mathtt{salt}^2, p, H_1^{1,1}, \ldots H_D^{\tau,1})$
- h_4^2 was sent by $\mathcal{A}$.
- Each of the $H_k^{1,e}$ are the output of a query to Hash_3

$$H_k^{1,e} = \mathsf{Hash}_3(\mathtt{salt}^1, \mathtt{salt}^2, e, \{[\![\boldsymbol{\alpha}]\!]^{1,e}, [\![\boldsymbol{\beta}]\!]^{1,e}, [\![\boldsymbol{v}]\!]^{1,e}\}_k)$$

- h_2 is the output of a query to Hash_2

$$h_2 = \mathsf{Hash}_2(m, \mathtt{salt}^1, \mathtt{salt}^2, \mathtt{COM}_1^{1,1}, \ldots, \mathtt{COM}_1^{1,\tau}, \mathtt{COM}_1^{2,1}, \ldots, \mathtt{COM}_1^{2,\tau})$$

- Each $\mathtt{COM}_1^{e,1}$ input to h_2 was generated by a prior query to Hash_1

$$\mathtt{COM}_1^{e,1} = \mathsf{Hash}_1(\mathtt{salt}^1, \mathtt{salt}^2, e, \mathtt{com}_1^{e,1}, \ldots, \mathtt{com}_{N^D}^{e,1})$$

- Each $\mathtt{com}_{i'}^{e,1}$ input to an above instance of $\mathtt{com}^{e,1}$ was generated by a prior query to Hash_0

$$\mathtt{com}_{i'}^{e,1} = \mathsf{Hash}_0(\mathtt{salt}^1, \mathtt{salt}^2, e, i', \mathtt{state}_{i'}^{e,1})$$

- The vector $\mathbf{x}^1$ defined by the leaf party states $\{\mathtt{state}_{i'}^{e,1}\}_{i \in [N^D]}$, has small weight $wt(\mathbf{x}^1) \leq \omega$ and syndrome $\mathbf{H}\mathbf{x}^2 + \mathbf{H}\mathbf{x}^1 = \boldsymbol{y}$.

In the event (we call *solve*) that such an execution exists, where the message m has not already been queried to the signing oracle, it is possible to extract the correct witness $\mathbf{x}$ from $\{\mathtt{state}_i^e\}_{i \in [ND]}$, which implies solving the underlying hard problem, so

$$\Pr_8(\text{solve}) \leq \epsilon_{SD}.$$

We claim that finding a forgery without solving the underlying problem means that Game 8 gives

$$\Pr_8(\text{forge} \wedge \overline{\text{solve}}) \leq q_2 \cdot p^{\tau'} + q_4 \cdot \left(\frac{1}{N^D}\right)^{\tau - \tau'}$$

with $p^{\tau'} = \sum_{i=\tau'}^{\tau} \binom{\tau}{i} p^i (1-p)^{\tau - i} \geq p^{\tau'}$, and τ' being the optimal number of false positives to find to minimize the cost of forgery as described in Equation 3. In this case, solve does not happen, so there is no successful execution e^* which gives a correct witness. Then to have obtained a forgery via query to Hash_5, the adversary must have either:

- Found a false positive polynomial and challenge point combination

$$\mathbf{S} \cdot \mathbf{Q} \neq \mathbf{F} \cdot \mathbf{P}, \text{ but } \mathbf{S} \cdot \mathbf{Q}(r_k) = \mathbf{F} \cdot \mathbf{P}(r_k), \text{ for } k \in [t],$$

  at challenge points r_k, which happens with probability p.

- Else (with probability $1 - p$) successfully cheat on a leaf party i' which will be successfully challenged with probability $1/N^D$.

- Or equivalently (also with probability $1 - p$) cheat on one out of N main parties, independently in each of the D protocols, with probability $(1/N)^D$, which is equivalent to cheating on one of the N^D leaf parties.

This must happen independently for each of the τ iterations of the protocol. □

6 Conclusion

After introducing the necessary cryptographic background, we proposed a straightforward construction for two-prover zero-knowledge proofs from hypercube syndrome decoding in-the-head. We proved its honest verifier zero-knowledge, knowledge soundness and witness privacy properties in the simulation based paradigm.

This two-prover zero-knowledge proof was then transformed to a two-party signature scheme using the Fiat-Shamir transformation. In the unforgeability proof for the signature, we used parts of the witness privacy simulator. Hopefully, this can be a helpful construction for future security proofs for multi-party signatures.

To the authors knowledge, this signature has the lowest communication overhead among two-prover MPCitH signature schemes, while being double the original signature in size.

Future work. To compare to other signature schemes, concrete signatures sizes and communication costs should be calculated, hopefully accompanied by an implementation of this scheme. An interesting question which naturally arises from two-party signatures is whether a similar approach to threshold signatures from MPCitH could be generalized to provide $O(n \cdot |\sigma|)$ size signatures for n parties, or to apply the two-party variant to other MPCitH schemes.

This thesis ignored the complexity of a malicously secure protocol which will replace the trusted setup in an implementation. To this end, we will note that both provers independently sampling low weight vectors $\boldsymbol{x}_1, \boldsymbol{x}_2$ and then calculating S, Q as in the original SDitH protocol leads to an additive sharing of a global witness $S_1 + S_2$ and a multiplicative sharing of a global witness $Q_1 \cdot Q_2$ with the desired properties.

Interesting future work also includes investigating the threshold linear sharing variant of SDitH [FR22] and assess its merits in the multi-party setting.

References

[ABB+25] Jean-Philippe Aumasson, Daniel J. Bernstein, Ward Beullens, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Andreas Hülsing, Panos Kampanakis, Stefan Kölbl, Tanja Lange, Martin M. Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe, and Bas Westerbaan. SPHINCS+: Stateless hash-based signature scheme. `https://sphincs.org/`, 2025. Accessed: 2025-05-15.

[AMGH+23] Carlos Aguilar-Melchor, Nicolas Gama, James Howe, Andreas Hülsing, David Joseph, and Dongze Yue. The return of the sdith. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 564–596. Springer, 2023.

[Bea92] Donald Beaver. Efficient multiparty protocols using circuit randomization. In Joan Feigenbaum, editor, *Advances in Cryptology — CRYPTO '91*, pages 420–432, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

[Beu21] Ward Beullens. MAYO: Practical post-quantum signatures from oil-and-vinegar maps. Cryptology ePrint Archive, Paper 2021/1144, 2021.

[BMvT78] E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems (corresp.). *IEEE Transactions on Information Theory*, 24(3):384–386, 1978.

[CC24] Eliana Carozza and Geoffroy Couteau. On threshold signatures from MPC-in-the-head. Cryptology ePrint Archive, Paper 2024/1897, 2024.

[CZC+21] Hongrui Cui, Kaiyi Zhang, Yu Chen, Zhen Liu, and Yu Yu. Mpc-in-multi-heads: A multi-prover zero-knowledge proof system. In Elisa Bertino, Haya Shulman, and Michael Waidner, editors, *Computer Security – ESORICS 2021*, pages 332–351, Cham, 2021. Springer International Publishing.

[DKR24] Jack Doerner, Yashvanth Kondi, and Leah Namisa Rosenbloom. Sometimes you can't distribute random-oracle-based proofs. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024*, pages 323–358, Cham, 2024. Springer Nature Switzerland.

[DLL+17] Leo Ducas, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehle. CRYSTALS – dilithium: Digital signatures from module lattices. Cryptology ePrint Archive, Paper 2017/633, 2017.

[EB21] Andre Esser and Emanuele Bellini. Syndrome decoding estimator. Cryptology ePrint Archive, Paper 2021/1243, 2021.

[FHK+25] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU. `https://falcon-sign.info/`, 2025. Accessed: 2025-05-15.

[FJR22] Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022*, pages 541–572, Cham, 2022. Springer Nature Switzerland.

[FR22] Thibauld Feneuil and Matthieu Rivain. Threshold linear secret sharing to the rescue of MPC-in-the-head. Cryptology ePrint Archive, Paper 2022/1407, 2022.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, August 1986.

[IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 21–30, 2007.

[KZ20] Daniel Kales and Greg Zaverucha. An attack on some signature schemes constructed from five-pass identification schemes. Cryptology ePrint Archive, Paper 2020/837, 2020.

[Lin17] Yehuda Lindell. *How to Simulate It – A Tutorial on the Simulation Proof Technique*, pages 277–346. Springer International Publishing, Cham, 2017.

[LN17] Yehuda Lindell and Ariel Nof. A framework for constructing fast mpc over arithmetic circuits with malicious adversaries and an honest-majority. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 259–276, New York, NY, USA, 2017. Association for Computing Machinery.

[LZW+24] Xuanming Liu, Zhelei Zhou, Yinghao Wang, Bingsheng Zhang, and Xiaohu Yang. Scalable collaborative zk-SNARK: Fully distributed proof generation and malicious security. Cryptology ePrint Archive, Paper 2024/143, 2024.

[McE78] Robert J McEliece. A public-key cryptosystem based on algebraic. *Coding Thv*, 4244(1978):114–116, 1978.

[Nat17] National Institute of Standards and Technology. Post-quantum cryptography standardization: Round 1 submissions, 2017. Accessed: 2025-05-15.

[OB22] Alex Ozdemir and Dan Boneh. Experimenting with collaborative zk-SNARKs: Zero-Knowledge proofs for distributed secrets. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 4291–4308, Boston, MA, August 2022. USENIX Association.

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of cryptology*, 13:361–396, 2000.

[Ste94] Jacques Stern. A new identification scheme based on syndrome decoding. In Douglas R. Stinson, editor, *Advances in Cryptology — CRYPTO' 93*, pages 13–21, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Hans Kristjan Veri**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Always Two, There Are: Towards Two-Party SDitH,

   supervised by Toomas Krips.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Hans Kristjan Veri
15/05/2025