Threshold Computation in the Head: Improved Framework for Post-Quantum Signatures and Zero-Knowledge Arguments

Thibauld Feneuil[1,2] and Matthieu Rivain[1]

[1] CryptoExperts, Paris, France
[2] Sorbonne Université, CNRS, INRIA, Institut de Mathématiques de Jussieu-Paris Rive Gauche, Ouragan, Paris, France
{thibauld.feneuil,matthieu.rivain}@cryptoexperts.com

Abstract. The MPC-in-the-Head paradigm is instrumental in building zero-knowledge proof systems and post-quantum signatures using techniques from secure multi-party computation. Many recent works have improved the efficiency of this paradigm. In this work, we improve the recently proposed framework of MPC-in-the-Head based on threshold secret sharing (to appear at Asiacrypt 2023), here called *Threshold Computation in the Head*. We first address the two main limitations of this framework, namely the degradation of the communication cost and the constraint on the number of parties. Our tweak of this framework makes it applicable to the previous MPCitH schemes (and in particular post-quantum signature candidates recently submitted to NIST) for which we obtain up to 50% timing improvements without degrading the signature size. Then we extend the TCitH framework to support quadratic (or higher degree) MPC round functions instead of being limited to linear functions as in the original framework. We show the benefits of our extended framework with several applications. We first propose a generic proof system for polynomial constraints that outperforms the former MPCitH-based schemes for proving low-degree arithmetic circuits. Then we apply our extended framework to derive improved variants of the MPCitH candidates submitted to NIST. For most of them, we save between 9% and 35% of the signature size. In particular, we obtain 4.2 KB signatures based on the (non-structured) MQ problem. Finally, we propose a generic way to build efficient post-quantum ring signatures from any one-way function. When applying our TCitH framework to this design with the MQ problem, the obtained scheme outperforms all the previous proposals in the state of the art. For instance, our scheme achieves sizes below 6 KB and timings around 10 ms for a ring of 4000 users.

1 Introduction

The MPC-in-the-Head (MPCitH) paradigm introduced in [IKOS07] is a versatile paradigm to build zero-knowledge proof systems from secure multi-party computation (MPC). While not providing (asymptotically) succinct proofs like SNARKs [BCCT12, Gro16], the MPCitH paradigm is particularly efficient for small circuits such as those involved to construct (post-quantum) signature schemes. This was first demonstrated by the Picnic signature scheme, submitted to the NIST PQC process in 2017 [ZCD+17]. In the recent NIST call for additional post-quantum signatures [NIS22], 7 candidates out of the 40 selected for the first round rely on the MPCitH paradigm.

The MPCitH paradigm can be summarized as follows. By emulating an MPC protocol verifying a witness and by opening some (verifier chosen) parties, the prover convinces the verifier they know the witness with soundness error around $1/N$, for N the number of parties involved in the MPC protocol. In the traditional MPCitH approach, the bottleneck in running times comes from the emulation of the N parties. Recent works have shown how this bottleneck can be mitigated. The hypercube technique proposed at Eurocrypt 2023 [AGH+23] improves the "traditional setting" (additive sharing with GGM tree commitments) by decreasing the emulation phase to $1 + \log_2 N$ parties with no extra communication cost. On the other hand, MPCitH based on threshold secret sharing [FR22] (to appear at Asiacrypt 2023), here called Threshold Computation in the Head (TCitH), only requires the emulation of a (small) constant number of parties. Specifically TCitH requires $\ell + 1$ parties for an $(\ell + 1, N)$-threshold sharing (which is 2 parties for $\ell = 1$). However, it suffers some communication penalties which is due to the use of Merkle trees in place of GGM

trees for the commitments of shares (this overhead typically represents 2 KB for non-interactive arguments with 128-bit security). Moreover, the number N of parties (and hence the achievable soundness) is limited by the size of the finite field on which the witness is defined ($N \leq |\mathbb{F}|$), which is an issue when dealing with small fields.

In this work, we improve the TCitH framework in several ways and put forward efficient applications of this framework. Our contributions are the following:

1. *TCitH with GGM trees.* We address the two aforementioned limitations of Threshold Computation in the Head compared to standard MPCitH (with seed trees and hypercube technique). We first show how using techniques from [CDI05], we can generate and commit Shamir's secret shares with the communication cost of a GGM tree (as in the traditional approach) while benefiting the low-cost MPC emulation of TCitH. In this TCitH with GGM tree setting, we further propose a way to mitigate the second limitation of TCitH. Specifically, we show that we can extend TCitH by lifting the MPC protocol into a field extension. We obtain an emulation phase with $1 + \lceil \log N / \log |\mathbb{F}| \rceil$ parties (for a soundness error of $\approx 1/N$). For $\mathbb{F}_2$, this is equivalent to the hypercube technique [AGH+23] while for a larger field our approach is strictly better. In comparison to the original TCitH framework with commitments based on Merkle trees, using this GGM variant saves the extra communication cost but loses the advantage of having a very fast verification algorithm inherited from Merkle trees, we hence get an interesting trade-off.
2. *Extended TCitH framework.* We extend the TCitH framework to threshold MPC protocols locally computing quadratic (or higher degree) functions instead of being restricted to linear functions. Our extended framework comes with two variants depending on the used method to generate and commit the shares: GGM tree (more compact) vs. Merkle tree (faster verification). By supporting higher-degree MPC computation, our extended framework is amenable to many potential applications. We notably apply it to derive succinct arguments for low-degree arithmetic circuits providing a very competitive alternative to post-quantum zero-knowledge arguments for small size arithmetic circuits [AHIV17, DOT21]. Our extended framework is conceptually close to the Ligero proof system [AHIV17] but while the latter targets "average-size computation" our framework achieves better sizes for "small-size computation" typically arising in the design of (post-quantum) signature schemes. We notably focus on low threshold secret sharing and tune the scheme parameters as well as the soundness analysis to this setting.
3. *Applications.* We show how our improved TCitH framework is instrumental to various applications and notably for post-quantum (ring) signatures. We first show that our basic (non-extended) TCitH framework with GGM tree can improve the performances of nearly all the recent NIST candidates based on the MPCitH paradigm. We then apply our extended framework to proposed improved variants of these candidates. For most of them, we save between 9% and 35% of the signature size. In particular, our framework applied to the non-structured multivariate quadratic (MQ) problem provides signature sizes of 4.2 KB which is to compare with the 6.3 KB of MQOM signatures (previously the smallest based on non-structured MQ) and 4.8 KB of Biscuit signatures (MPCitH scheme based on a structured MQ instance) [FR23, BKPV23]. We also apply our TCitH framework to design efficient post-quantum ring signatures from any one-way function. We propose concrete instances relying on the MQ and syndrome decoding (SD) problems. For a ring of 1000 users, both schemes have a running time below 10 ms, while achieving sizes around 5 KB for MQ and 9 KB with SD, which greatly improves the current state of the art.

Finally, we provide a comparative analysis of our TCitH framework and the recently proposed *VOLE-in-the-Head* framework which draws strong links between the two approaches.

2 Preliminaries

In this paper, we shall use the standard cryptographic notions of pseudorandom generator (PRG), collision resistant hash function, (binding and hiding) commit scheme, Merkle tree, and zero-knowledge proof of knowledge. While we do not reintroduce these notions here, the reader is referred to [FR22] for their formal definitions with similar notations.

2.1 Secret Sharing

Along the paper, the sharing of a value s is denoted $[\![s]\!] := ([\![s]\!]_1, \ldots, [\![s]\!]_N)$ with $[\![s]\!]_i$ denoting the share of index i for every $i \in [1 : N]$. For any subset of indices $J \subseteq [N]$, we shall further denote $[\![s]\!]_J := \left([\![s]\!]_i\right)_{i \in J}$.

A (t, N)-*threshold secret sharing scheme* is a method to share a value $s \in \mathbb{F}$ into a sharing $[\![s]\!]$ such that s can be reconstructed from any t shares while no information is revealed on the secret from the knowledge of $t-1$ shares. A *linear secret sharing scheme* (LSSS) is a secret sharing scheme such that for any two sharings $[\![s_1]\!]$, $[\![s_2]\!]$ and any two values $a_1, a_2 \in \mathbb{F}$, computing $a_1 \cdot [\![s_1]\!] + a_2 \cdot [\![s_2]\!]$ yields a sharing of $a_1 s_1 + a_2 s_2$.

The *additive secret sharing* is an (N, N)-threshold LSSS for which $s = \sum_{i=1}^{N} [\![s]\!]_i$. The Shamir's secret sharing is formally defined hereafter.

The *Shamir's secret sharing* over $\mathbb{F}$ of degree ℓ is an $(\ell+1, N)$-threshold LSSS for which a sharing $[\![s]\!]$ of $s \in \mathbb{F}$ is built as follows:

- sample $r_1, \ldots, r_\ell$ uniformly in $\mathbb{F}$,
- build the polynomial P_s as $P_s(X) := s + \sum_{i=1}^{\ell} r_i X^i$,
- build the shares $[\![s]\!]_i$ as evaluations $P_s(e_i)$ of P_s for each $i \in [1 : N]$, where $e_1, \ldots, e_N$ are public non-zero distinct points of $\mathbb{F}$.[3]

For any subset $J \subseteq [N]$, s.t. $|J| = \ell + 1$, the recovery of s from $[\![s]\!]_J$ works by interpolating the polynomial P_s from the $\ell + 1$ evaluation points $[\![s]\!]_J = (P_s(e_i))_{i \in J}$ and outputing the constant term of P_s.

Along this paper, we shall consider sharings of vectors. A sharing $[\![v]\!]$ of a vector $v \in \mathbb{F}^{|v|}$ is simply a vector of $|v|$ independent sharings of the coordinates of v. The polynomial underlying the Shamir's secret sharing of a vector v is a hence vector polynomial $P_v(X) \in (\mathbb{F}[X])^{|v|}$. We shall denote coeff_i the function mapping a polynomial P_v to its degree-i coefficient. If P_v is a vector polynomial, then $\mathsf{coeff}_i(P_v) \in \mathbb{F}^{|v|}$ is naturally defined as the vector of the degree-i coefficients of the coordinates of v.

2.2 The MPC-in-the-Head Paradigm

The MPC-in-the-Head (MPCitH) paradigm was introduced by Ishai, Kushilevitz, Ostrovsky and Sahai in [IKOS07] to build zero-knowledge proofs from secure multi-party computation (MPC) protocols. We first recall the general principle of this paradigm before introducing a formal model for the underlying MPC protocols and their transformation into zero-knowledge proofs.[4]

Assume we want to build a zero-knowledge proof of knowledge of a witness w for a statement x such that $(x, w) \in \mathcal{R}$ for some relation $\mathcal{R}$. To proceed, we shall use an MPC protocol in which N parties $\mathcal{P}_1, \ldots, \mathcal{P}_N$ securely and correctly evaluate a function f on a secret witness w with the following properties:

- each party $\mathcal{P}_i$ takes a share $[\![w]\!]_i$ as input, where $[\![w]\!]$ is a sharing of w;
- the function f outputs Accept when $(x, w) \in \mathcal{R}$ and Reject otherwise;
- the protocol is ℓ-private in the semi-honest model, meaning that the views of any ℓ parties leak no information about the secret witness.

We can use this MPC protocol to build a zero-knowledge proof of knowledge of a witness w satisfying $(x, w) \in \mathcal{R}$. The prover proceeds as follows:

- they build a random sharing $[\![w]\!]$ of w;
- they simulate locally ("in her head") all the parties of the MPC protocol;
- they send a commitment of each party's view to the verifier, where such a view includes the party's input share, its random tape, and its received messages (the sent messages can further be deterministically derived from those elements);
- they send the output shares $[\![f(w)]\!]$ of the parties, which should correspond to a sharing of Accept.

[3] We also consider the "evaluation point" $e_i = \infty$. For this special evaluation point, $P_s(\infty)$ is defined as the leading degree coefficient of P_s, namely $P_s(\infty) = r_\ell$.

[4] The following formalism is heavily borrowed from [FR22].

Then the verifier randomly chooses ℓ parties and asks the prover to reveal their views. After receiving them, the verifier checks that they are consistent with an honest execution of the MPC protocol and with the commitments. Since only ℓ parties are opened, the revealed views leak no information about the secret witness w, which ensures the zero-knowledge property. On the other hand, the random choice of the opened parties makes the cheating probability upper bound by $1 - \binom{N-2}{\ell-2}/\binom{N}{\ell}$, which ensures the soundness of the proof.

The MPCitH paradigm simply requires the underlying MPC protocol to be secure in the semi-honest model (and not in the malicious model), meaning that the parties are assumed to be honest but curious: they follow honestly the MPC protocol while trying to learn secret information from the received messages.

2.3 General Model for MPCitH-Friendly MPC Protocols

Several simple MPC protocols have been proposed that yield fairly efficient zero-knowledge proofs and signature schemes in the MPCitH paradigm, see for instance [KZ20b, BD20, BN20, BDK+21, FJR22]. These protocols lie in a specific subclass of MPC protocols in the semi-honest model which is formalized in [FR22]. In this model, an MPC protocol performs its computation on a base finite field $\mathbb{F}$ so that all the manipulated variables (including the witness w) are tuples of elements from $\mathbb{F}$. In what follows, the sizes of the different tuples involved in the protocol are kept implicit for the sake of simplicity. The parties take as input an additive sharing $[\![w]\!]$ of the witness w (one share per party), which is defined as

$$[\![w]\!] = ([\![w]\!]_1, \ldots, [\![w]\!]_N) \quad \text{s.t.} \quad w = \sum_{i=1}^{N} [\![w]\!]_i \ .$$

Then the parties compute one or several rounds in which they perform three types of actions:

Receiving randomness: The parties receive a random value (or random tuple) ε from a randomness oracle $\mathcal{O}_R$. When calling this oracle, all the parties get the same random value ε.

Receiving hint: The parties receive a fresh sharing $[\![\beta]\!]$ (one share per party) from a hint oracle $\mathcal{O}_H$. The hint β can depend on the witness w and the previous random values sampled from $\mathcal{O}_R$.

Computing & broadcasting: The parties locally compute $[\![\alpha]\!] := [\![\varphi(v)]\!]$ from a sharing $[\![v]\!]$ where φ is an $\mathbb{F}$-linear function, then broadcast all the shares $[\![\alpha]\!]_1, \ldots, [\![\alpha]\!]_N$ to publicly reconstruct $\alpha := \varphi(v)$.

After t rounds of the above actions, the parties finally output Accept if and only if the publicly reconstructed values $\alpha^1, \ldots, \alpha^t$ satisfy the relation $g(\alpha^1, \ldots, \alpha^t) = 0$ for a given function g. As formalized in [FR22], such an MPC protocol has a *false positive probability*. Namely, given the sharing of an *invalid* witness as input, the protocol might still output Accept with some probability p over the random choice of the values $\varepsilon^1, \ldots, \varepsilon^t$ from the randomness oracle $\mathcal{O}_R$. We refer to [FR22] for a formal definition or to Section 4 below where we extend this MPC model.

2.4 MPCitH Transform based on Additive Sharing and GGM Trees

Any MPC protocol complying with the above description gives rise to a practical short-communication zero-knowledge proof in the MPCitH paradigm. The resulting zero-knowledge proof is described in Protocol 5 (in Appendix A): after sharing the witness w, the prover emulates the MPC protocol "in her head", commits the parties' inputs, and sends a hash digest of the broadcast communications; finally, the prover reveals the requested parties' inputs as well as the broadcast messages of the unopened party, thus enabling the verifier to emulate the computation of the opened parties and to check the overall consistency.

Soundness. Assuming that the underlying MPC protocol follows the model of Section 2.3 with a false positive probability p, the soundness error of Protocol 5 is

$$\frac{1}{N} + \left(1 - \frac{1}{N}\right) \cdot p \ . \tag{1}$$

The above formula results from the fact that a malicious prover might successfully cheat with probability $1/N$ by corrupting the computation of one party or with probability p by making the MPC protocol produce a false positive. This soundness has been formally proven in [FR22] for the general MPC model recalled above as well as for several specific MPC protocols in other previous works – see, e.g., [DOT21, BN20, FJR22].

Pseudorandomness and GGM trees. The communication of Protocol 5 includes:

- the input shares $(\llbracket w \rrbracket_i, \llbracket \beta^1 \rrbracket_i, \ldots, \llbracket \beta^t \rrbracket_i)$ of the opened parties. In practice, a seed $\mathsf{seed}_i \in \{0,1\}^\lambda$ is associated with each party so that for each committed variable v (among the witness w and the hints $\beta^1, \ldots, \beta^t$) the additive sharing $\llbracket v \rrbracket$ is built as

$$\begin{cases} \llbracket v \rrbracket_i \leftarrow \mathrm{PRG}(\mathsf{seed}_i) \text{ for } i \neq N \\ \llbracket v \rrbracket_N = v - \sum_{i=1}^{N-1} \llbracket v \rrbracket_i. \end{cases}$$

  Thus, instead of committing $(\llbracket w \rrbracket_i, \llbracket \beta^1 \rrbracket_i)$, the initial commitments simply include the seeds for $i \neq N$, and com_i^j becomes useless for $j \geq 2$ and $i \neq N$. Formally, we have:

$$\mathsf{com}_i^j = \begin{cases} \mathrm{Com}(\mathsf{seed}_i; \rho_i^1) & \text{for } j = 1 \text{ and } i \neq N \\ \mathrm{Com}(\llbracket w \rrbracket_N, \llbracket \beta^1 \rrbracket_N; \rho_N^1) & \text{for } j = 1 \text{ and } i = N \\ \emptyset & \text{for } j > 1 \text{ and } i \neq N \\ \mathrm{Com}(\llbracket \beta^j \rrbracket_N; \rho_N^j) & \text{for } j > 1 \text{ and } i = N \end{cases}$$

  Some coordinates of the β^j might be uniformly distributed over $\mathbb{F}$ (remember that the β^j are tuples of $\mathbb{F}$ elements). We denote β^{unif} the sub-tuple composed of those uniform coordinates. In this context, the last share $\llbracket \beta^{\mathrm{unif}} \rrbracket_N$ can be built as $\llbracket \beta^{\mathrm{unif}} \rrbracket_N \leftarrow \mathrm{PRG}(\mathsf{seed}_N)$ so that a seed seed_N can be committed in com_N^1 (instead of committing $\llbracket \beta^{\mathrm{unif}} \rrbracket_N$). This way the prover can save communication by revealing seed_N instead of $\llbracket \beta^{\mathrm{unif}} \rrbracket_N$ whenever the latter is larger;
- the messages $\llbracket \alpha^1 \rrbracket_{i^*}, \ldots, \llbracket \alpha^t \rrbracket_{i^*}$ broadcasted by the unopened party. Let us stress that one can sometimes save communication by sending only some elements of $\llbracket \alpha^1 \rrbracket_{i^*}, \ldots, \llbracket \alpha^t \rrbracket_{i^*}$ and use the relation $g(\alpha^1, \ldots, \alpha^t) = 0$ to recover the missing ones;
- the hash digests $h_1, \ldots, h_{t+1}$ and the unopened commitments $\mathsf{com}_{i^*}^1, \ldots, \mathsf{com}_{i^*}^t$ (as explained above, we have $\mathsf{com}_{i^*}^j = \emptyset$ for $j > 1$ if $i^* \neq N$).

As suggested in [KKW18], instead of revealing the $(N-1)$ seeds of the opened parties, one can generate them from a GGM tree [GGM84] (a.k.a. a *tree PRG* or *seed tree*). Such a tree is a pseudorandom generator that expands a root seed mseed into N subseeds in a structured way. The principle is to label the root of a binary tree of depth $\lceil \log_2 N \rceil$ with mseed. Then, one inductively labels the children of each node with the output of a standard PRG applied to the node's label. The subseeds $(\mathsf{seed}_i)_{i \in [1:N]}$ are defined as the labels of the N leaves of the tree. Such a seed tree makes it possible to reveal all the subseeds but one by only revealing $\log_2(N)$ labels of the tree. The principle is to reveal the *sibling path* of the seed seed_{i^*} which one wants to keep secret (i.e., all the labels on the siblings of the path from seed_{i^*} to the root). Those labels allow the verifier to reconstruct the $N-1$ seeds $(\mathsf{seed}_i)_{i \in [1:N] \setminus \{i^*\}}$. Using GGM trees, the prover hence only needs to communicate $\log_2 N$ λ-bit seeds to the verifier.

2.5 Threshold Computation in the Head: Original Framework

In [FR22], the authors suggest building proof systems from the MPC-in-the-Head framework using a threshold secret sharing scheme instead of using additive sharing as the wide majority of previous works. Their approach leads to faster running times compared to the rest of the state of the art. For an MPC protocol complying to the model described in Section 2.3, the first step of the transform consists in replacing the additive sharings handled by the protocol by $(\ell+1, N)$-threshold linear secret sharings (e.g. Shamir's secret sharings). Since the MPC protocols in this model only require linearity and ℓ-privacy from the sharings, this transformation is straightforward. Then, the TCitH transform compiles this MPC protocol into the following proof of knowledge:

1. The prover generates a $(\ell+1, N)$-threshold sharing $[\![w]\!]$ of w, and they commit each share independently: for all $i \in \{1, \dots, N\}$, $\mathsf{com}_i \leftarrow \mathrm{Com}([\![w]\!]_i, \rho_i)$ where ρ_i is some commitment randomness. They send a hash digest h_1 of all $\{\mathsf{com}_i\}_{i\in[1:N]}$ to the verifier.
2. The prover emulates a subset $S \subset \{1, \dots, N\}$ of $\ell+1$ parties and they send a hash digest h_2 of the values which have been broadcast by these parties to the verifier.
3. The verifier samples a random subset $I \subset \{1, \dots, N\}$ of ℓ parties.
4. The prover opens the commitment of all the parties in I, namely they send $([\![w]\!]_i, \rho_i)_{i\in I}$. The prover further sends additional information to enable the verifier to recompute h_1. Additionally, the prover sends broadcast shares of an unopened party $i^* \in S\backslash I$.
5. The verifier checks that the open commitments are consistent with the corresponding hash and that the revealed parties are consistent with the hash of the broadcast values.

Since only a small number of commitments need to be open in the TCitH framework, one relies on a Merkle tree instead of a GGM tree. Namely, the commitment h_1 is computed as the root of a Merkle tree with leaves com_i. Then, while opening the commitments, the prover further sends the authentication paths of the opened leaves $\{\mathsf{com}_i\}_{i\in I}$ to allow the verifier to recompute and check h_1. In [FR22], the soundness error of the obtained proof system is shown to be

$$\frac{1}{\binom{N}{\ell}} + p \cdot \frac{\ell(N-\ell)}{\ell+1} \ , \tag{2}$$

where p is the false positive probability of the underlying multiparty protocol. Since ℓ is typically a small constant (for example, $\ell \in \{1, 2, 3\}$), the MPC emulation is far cheaper than in the traditional MPCitH framework in which we used to emulate all the N parties.

The hypercube technique introduced in [AGH+23] already reduces the cost of emulating the MPC protocol to $1 + \log_2 N$ parties (instead of N) without impacting the communication cost. On the other hand, the original TCitH framework [FR22] decreases the emulation cost even more, to a constant number of parties, but the communication is slightly larger. This is for two reasons:

- TCitH commitments are based on a Merkle tree for which an opening is twice larger than with a GGM tree. This is because a Merkle authentication path is made of $\log N$ hash digests of 2λ bits while a GGM sibling path is made of $\log N$ seeds of λ bits.
- There is a soundness loss since a malicious prover can commit an invalid sharing (see Equation (2) *vs.* Equation (1)).

In the next section, we show how we can use GGM trees for commitments in the TCitH framework, thus achieving a constant number of party emulations (of $\ell+1$) without communication penalty.

3 TCitH with Pseudorandom Sharing and GGM Trees

3.1 General Technique

In this section, we only work on the case of the Shamir's secret sharing for the sake of simplicity (and since we will use this sharing scheme in practice). Let us stress that the results of this section further apply to any linear secret sharing scheme (LSSS).

Sharing generation. We propose to generate the Shamir's secret sharings involved in TCitH framework in a *pseudorandom way* using the technique described in [CDI05]. The first step consists in additively sharing the secret w into $\binom{N}{\ell}$ shares, each labelled by a different set from $\{T \subset [1:N], |T| = \ell\}$:

$$w = \sum_{T \subset [1:N], |T| = \ell} s_T.$$

This is also known as an ℓ-private *replicated secret sharing* [ISN89, CDI05].

We can optimize the generation of this additive sharing using a GGM seed tree as in the MPCitH transform with additive sharing described in Section 2.4. Then, for every $i \in [1:N]$, the i^{th} party receives the additive shares $\{s_T\}_{i \notin T}$ and converts them into one Shamir's share $[\![w]\!]_i$.

Let us denote S_ℓ^N all the subsets of $[1:N]$ of size ℓ, and let us take such a subset $T_0 \in S_\ell^N$. We also denote $|w|$ the length of the secret vector w. Formally, the sharing generation works as follows:

1. We sample a root seed $\mathsf{rseed} \in \{0,1\}^\lambda$.
2. We expand this root seed through a GGM tree to obtain $\binom{N}{\ell}$ leaf seeds $\{\mathsf{seed}_T\}_{T \in S_\ell^N}$.
3. For all T, we expand $s_T \in \mathbb{F}^{|w|}$ from the seed seed_T using a pseudorandom generator:

$$s_T \leftarrow \mathrm{PRG}(\mathsf{seed}_T).$$

4. We compute the *auxiliary value* Δw as $\Delta w := w - \sum_{T \in S_\ell^N} s_T$. We thus have

$$w = \Delta w + \sum_{T \in S_\ell^N} s_T.$$

Let us denote $P_T \in \mathbb{F}[X]$ the unique degree-ℓ polynomial[5] such that

$$\begin{cases} P_T(0) = 1 \\ P_T(e_j) = 0 \text{ for all } j \in T \end{cases}$$

where $\{e_j\}_j$ are the parties' evaluation points of the Shamir secret sharing scheme. For $i \in [1:N]$, we compute $[\![w]\!]_i \in \mathbb{F}^{|w|}$ as

$$[\![w]\!]_i := \sum_{T \in S_\ell^N, i \notin T} s_T \cdot P_T(e_i) + \begin{cases} \Delta w \cdot P_{T_0}(e_i) & \text{if } i \notin T_0, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Correctness. Let us analyze the sharing $\{[\![w]\!]_i\}_i$ obtained using the above procedure. We define the polynomial $P_w(X) \in \left(\mathbb{F}[X]\right)^{|w|}$ as

$$P_w(X) \;:=\; \Delta w \cdot P_{T_0}(X) + \sum_{T \in S_\ell^N} s_T \cdot P_T(X)\ , \tag{4}$$

such that $[\![w]\!]_i = P(e_i)$ for all i. Since all polynomials $\{P_T\}_T$ are obtained by interpolation of $\ell + 1$ points, they are of degree (at most) ℓ. We hence directly have that P is a degree-ℓ polynomial. Moreover, we have

$$\begin{aligned} P(0) &= \Delta w \cdot P_{T_0}(0) + \sum_{T \in S_\ell^N} s_T \cdot P_T(0) \\ &= \Delta w + \sum_{T \in S_\ell^N} s_T = w\ . \end{aligned}$$

We thus deduce that the shares $\{[\![w]\!]_i\}_i$ forms a valid Shamir's secret sharing of w, since they are evaluations of a degree-ℓ polynomial with w as constant term.

[5] If there is a $j \in T$ such that $e_j = \infty$, P_T is of degree $\ell - 1$.

Remark 1. Steps 1–4 consist in generating the ℓ-private replicated secret sharing of w, using a GGM tree. Then Step 5 consists in converting this additive sharing into a Shamir's sharing.

Remark 2. As mentioned previously, the generation process can be generalized for any LSSS. For $T \in S_\ell^N$, let us denote $v^{(T)}$ the sharing of 1 such that the i^{th} share is zero for all $i \in T$ (it necessarily exists thanks to the privacy property of the LSSS). Then, we can build a pseudo-random sharing of this sharing scheme using the same procedure as before except that we compute $[\![w]\!]_i$ as

$$[\![w]\!]_i := \Delta w \cdot v_i^{(T_0)} + \sum_{T \in S_\ell^N, i \notin T} s_T \cdot v_i^{(T)}.$$

Protocol description. We rely on the same zero-knowledge protocol as in the original TCitH framework tweaked with the above sharing generation. Instead of committing $\{[\![w]\!]_i\}_i$, the prover commits $\{\mathsf{seed}_T\}_{T \in S_\ell^N}$ and Δw. To open a party i, the prover then needs to reveal all the seeds $\{\mathsf{seed}_T\}_{T \in S_\ell^N, i \notin T}$ and Δw (if $i \notin T_0$), from which the verifier can recompute the share $[\![w]\!]_i$ thanks to Equation (3). In practice, the prover should reveal a subset I of ℓ parties, so they will reveal

$$\begin{cases} \text{all the seeds } \{\mathsf{seed}_T\}_{T \neq I}, \\ \Delta w \text{ if } I \neq T_0. \end{cases}$$

In other words, it means that the prover should reveal all the seeds except seed_I. To proceed, they just need to reveal the *sibling path* of the hidden leaf seed_I in the GGM tree.

Protocol 1 formally describes the zero-knowledge protocol obtained by applying the above TCitH framework with GGM tree to the general MPC protocol formalized in [FR22] and overviewed in Section 2.3, where $\{\Phi^j\}_j$ are the functions locally applied to derive the broadcast shares and ψ^j are the functions defining the hints.[6]

Computing on polynomial coefficients. As briefly mentioned in [FR22], instead of emulating a subset of $\ell+1$ parties (*i.e.* applying the MPC computation for $\ell+1$ shares), the prover can directly emulate the MPC protocol *on the* $\ell+1$ *coefficients of the polynomial* P_w (underlying the Shamir's secret sharing $[\![w]\!]$). Since the constant term of the polynomial corresponds to the plain secret value, emulating the MPC protocol on the constant coefficient is equivalent to computing the function underlying the MPC protocol (the function f in Equation (5)) on the plain input witness. Such "emulation" is often cheaper than a party emulation, since, in the MPCitH context, the prover already know some expected values α^j (which satisfy $g(\alpha^1, \ldots, \alpha^t) =$ Accept) and can thus save some computation.

Protocol routines. Protocol 1 is based on the following three routines that deal with sharings:

- GenerateSharingPolynomial takes as inputs an auxiliary value and the expanded randomness (*i.e.* the randomness expanded from all the seeds), and outputs the corresponding Shamir's polynomial (the polynomial involved in the Shamir's secret sharing). Formally, the call $\mathsf{GenerateSharingPolynomial}(\Delta x, (s_T)_{T \in S_\ell^N})$ outputs the polynomial P_x defined as

$$P_x(X) = \Delta x \cdot P_{T_0}(X) + \sum_T s_T \cdot P_T(X) \ \in \big(\mathbb{F}[X]\big)^{|x|}.$$

[6] A formal description of the general MPC protocol is also provided in Section 4.1 (see Protocol 2). The zero-knowledge protocol described here (Protocol 1) is an application of our TCitH framework with GGM tree to this general MPC protocol with the restriction that the Φ^j round functions are made of inner functions $\varphi^{j,k}$ which are linear.

1. The prover samples a root seed $\mathsf{rseed} \in \{0,1\}^\lambda$ and expands it through a GGM tree to obtain $\binom{N}{\ell}$ leaf seeds $\{\mathsf{seed}_T\}_{T \in S_\ell^N}$.
2. The prover expands each leaf seeds, $s_T^0 \leftarrow \mathrm{PRG}(\mathsf{seed}_T)$ for all T, builds the auxiliary value $\Delta w := w - \sum_T s_T^0$, and computes
$$P_w \leftarrow \mathsf{GenerateSharingPolynomial}(\Delta w, \{s_T^0\}_{T \in S_\ell^N}).$$
3. The prover emulates "in her head" the MPC protocol.
   For $j = 1$ to t:
   (a) the prover computes
   $$\beta^j = \psi^j(w, \{\varepsilon^k\}_{k<j}; r^j)$$
   using fresh uniform random tape r^j.
   (b) the prover expands more randomness $\{s_T^j\}_{T \in S_\ell^N}$ from the leaf seeds (such that $|s_T^j| = |\beta^j|$ for all T), builds the auxiliary value $\Delta\beta^j := \beta^j - \sum_T s_T^j$, and computes
   $$P_{\beta^j} \leftarrow \mathsf{GenerateSharingPolynomial}(\Delta\beta^j, \{s_T^j\}_{T \in S_\ell^N}).$$
   (c) the prover expands some commitment randomness ρ_T^j from the leaf seeds (such that $|\rho_T^j| = \lambda$ for all T) and computes the commitments
   $$\mathsf{com}_T^j := \begin{cases} \mathrm{Com}(\mathsf{seed}_T; \rho_T^j) & \text{if } j = 1 \text{ and } T \neq T_0 \\ \mathrm{Com}(\mathsf{seed}_T, \Delta w, \Delta\beta^j; \rho_T^j) & \text{if } j = 1 \text{ and } T = T_0 \\ \emptyset & \text{if } j > 1 \text{ and } T \neq T_0 \\ \mathrm{Com}(\Delta\beta^j; \rho_T^j) & \text{if } j > 1 \text{ and } T = T_0 \end{cases}$$
   for all $T \in S_\ell^N$.
   (d) the prover sends
   $$h_j := \begin{cases} \mathrm{Hash}(\{\mathsf{com}_T^1\}_T) & \text{if } j = 1 \\ \mathrm{Hash}(\{\mathsf{com}_T^j\}_T, P_{\alpha^{j-1}}) & \text{if } j > 1 \end{cases}$$
   to the verifier;
   (e) the verifier picks at random a challenge ε^j and sends it to the prover;
   (f) the prover compute the plain broadcast
   $$\alpha^j := \mathsf{coeff}_0(P_{\alpha^j}) = \Phi^j\big(w, (\beta^k)_{k \leq j}\big).$$
   (g) the prover computes, for $i \in [1:\ell]$,
   $$\mathsf{coeff}_i(P_{\alpha^j}) := \Phi^j\big(\mathsf{coeff}_i(P_w), (\mathsf{coeff}_i(P_{\beta^k}))_{k \leq j}\big).$$
   The prover further computes $h_{t+1} := \mathrm{Hash}(P_{\alpha^t})$ and sends it to the verifier.
4. The verifier picks at random a subset $I \subset [1:N]$ of ℓ parties (*i.e.* $|I| = \ell$) and sends it to the prover.
5. The prover reveals the views of all the parties in I, namely they send the sibling path of seed_I in the GGM tree to the verifier, together with $\Delta w, \{\Delta\beta^j\}_{j \in [1:t]}$ when $I \neq T_0$. The prover further sends the digests of the unopened commitments $\{\mathsf{com}_I^j\}_{j \in [1:t]}$ and the plain broadcast values $\{\alpha^j\}_{j \in [1:t]}$.
6. The verifier recomputes the commitments com_T^j for $T \neq T_0$ and $j \in [1:t]$ from the sibling path and the auxiliary values (in the same way as the prover). They expand all the randomness $(s_T^0, s_T^1, \ldots, s_T^t)_{T \neq I}$ from seeds and build the share of the open parties: for all $i \in I$,
   $$\begin{cases} [\![w]\!]_i = \mathsf{GeneratePartyShare}_i(\{s_T^0\}_{T: i \notin T}, \Delta w) \\ [\![\beta^j]\!]_i = \mathsf{GeneratePartyShare}_i(\{s_T^j\}_{T: i \notin T}, \Delta\beta^j) \text{ for all } j \in [1:t] \end{cases}$$
   where Δw and $\{\Delta\beta^j\}_j$ are omitted if not provided by the prover. The verifier can emulate the MPC protocol on the open parties: for all $i \in I$ and $j \in [1:t]$,
   $$[\![\alpha^j]\!]_i := \Phi^j\big([\![w]\!]_i, ([\![\beta^k]\!]_i)_{k \leq j}\big)$$
   Finally, the verifier can recompute the Shamir's polynomials P_{α^j} of all $\{[\![\alpha^j]\!]\}_j$: for all $j \in [1:t]$,
   $$P_{\alpha^j} = \mathsf{RecomputeSharing}_I(\alpha^j, ([\![\alpha^j]\!]_i)_{i \in I}).$$
7. The verifier accepts if and only if:
   (a) the views of the opened parties are consistent with each other, with the committed input shares and with the hash digest of the broadcast messages, *i.e.* for $j = 1$ to $t+1$,
   $$h_j \stackrel{?}{=} \begin{cases} \mathrm{Hash}(\{\mathsf{com}_T^1\}_T) & \text{if } j = 1 \\ \mathrm{Hash}(\{\mathsf{com}_T^j\}_T, P_{\alpha^{j-1}}) & \text{if } 2 \leq j \leq t \\ \mathrm{Hash}(P_{\alpha^t}) & \text{if } j = t+1; \end{cases}$$
   (b) the output of the opened parties are Accept, *i.e.*
   $$g(\alpha^1, \ldots, \alpha^t) \stackrel{?}{=} 0\ .$$

Protocol 1: Zero-knowledge protocol: application of the TCitH framework with GGM tree to the general MPC protocol of [FR22].

– $\mathsf{GeneratePartyShare}_i$ (for some $i \in [1:N]$) builds the share of the i^{th} party, using Equation (3). It takes as inputs the randomness $(s_T)_{T:i\notin T}$, together with the auxiliary value when necessary. Formally, the call $\mathsf{GeneratePartyShare}_i((s_T)_{T:i\notin T}, \Delta x)$ outputs

$$[\![x]\!]_i := \sum_{T:i\notin T} s_T \cdot P_T(e_i) + \begin{cases} \Delta x \cdot P_{T_0}(e_i) & \text{if } i \notin T_0, \\ 0 & \text{otherwise.} \end{cases}$$

– $\mathsf{RecomputeSharing}_I$ builds the Shamir's polynomial P_x from a plain value x and ℓ party shares $([\![x]\!]_i)_{i\in I}$. It simply performs Lagrange interpolation with the points $P_x(0) = x$ and $P_x(e_i) = [\![x]\!]_i$ for all $i \in I$. Formally, the call $\mathsf{RecomputeSharing}_I(x, ([\![x]\!]_i)_{i\in I})$ outputs the polynomial $P_x \in \big(\mathbb{F}[X]\big)^{|x|}$ defined as

$$P_x(X) := x \cdot \prod_{j\in I'} \frac{X - e_j}{-e_j} + \sum_{i\in I'} \left([\![x]\!]_i \cdot \frac{X}{e_i} \cdot \prod_{j\in I', j\neq i} \frac{X - e_j}{e_i - e_j} \right) + c_\infty \cdot X \cdot \prod_{j\in I'} (X - e_j)$$

where i_∞ is the index such that $e_{i_\infty} = \infty$, and

$$\begin{cases} I' := I \setminus \{i_\infty\} \text{ and } c_\infty := [\![x]\!]_{i_\infty} & \text{if } i_\infty \in I, \\ I' := I \text{ and } c_\infty := 0 & \text{if } i_\infty \notin I. \end{cases}$$

Soundness and zero-knowledge analysis. Let us analyze the soundness of Protocol 1. From a high-level point of view, we just changed how the shares of the Shamir's secret sharing are built. In the original TCitH framework, one cannot force the prover to commit a valid sharing (*i.e.*, where the shares are the evaluations of the same degree-ℓ polynomial). This degree of freedom impacts the soundness of the scheme: the false positive probability p is scaled by a factor $\ell(N-\ell)/(\ell+1)$ in the soundness error (see Equation (2)) which constrains the protocol to have a low p or to suffer an important loss in soundness. The situation is different here: a malicious prover shall commit $\binom{N}{\ell}$ seeds $\{\mathsf{seed}_T\}_T$ with an auxiliary value Δw. These values always define a valid Shamir's secret sharing with underlying polynomial

$$P_w(X) = \Delta w \cdot P_{T_0}(X) + \sum_T s_T \cdot P_T(X)$$

where $s_T \leftarrow \mathrm{PRG}(\mathsf{seed}_T)$ for all T. While this sharing might not correspond to a valid witness, it is a valid Shamir's secret sharing of a (possibly invalid) witness w. Namely, all the sets of $\ell+1$ shares among the $[\![w]\!]_i$'s encode the same (possibly invalid) witness w. Thus, a malicious prover has no way of committing to something that is not a valid Shamir's secret sharing. For this reason, the soundness error ϵ is

$$\epsilon := \frac{1}{\binom{N}{\ell}} + p \cdot \left(1 - \frac{1}{\binom{N}{\ell}}\right),$$

which for $\ell = 1$ matches the soundness error of the MPCitH framework with additive sharing (see Equation (1)). The obtained soundness is hence better than the original TCitH framework for which the soundness error is degraded by the fact that a malicious prover might commit an invalid Shamir's secret sharing. This result is formally stated in Theorem 1 in the next section (for an extension of the TCitH framework).

Regarding the zero-knowledge property, it simply holds from the following observation: the seed seed_I remains hidden and the plain witness w is masked by the hidden value $s_I := \mathrm{PRG}(\mathsf{seed}_I)$. This ensures that the proof system leaks no information about the witness (provided that the underlying MPC protocol is ℓ-private in the semi-honest model).

Performances. Let us analyze the communication cost of Protocol 1. The prover sends

– $t+1$ hash digests $h_1, \ldots, h_{t+1}$, which cost $(t+1) \cdot 2\lambda$ bits;

– the sibling path of seed_I in a seed tree with $\binom{N}{\ell}$, which costs $\lambda \cdot \log_2 \binom{N}{\ell}$ bits;
– the auxiliary values $(\Delta x, \{\Delta\beta^j\}_{j\in[1:t]})$ when $I \neq T_0$;
– the plain broadcast values $\alpha^1, \ldots, \alpha^t$;
– some commitment digests $\{\mathsf{com}_I^j\}_{j\in[1:t]}$, which cost $t \cdot 2\lambda$ bits when $I = T_0$ and 2λ bits otherwise (since com_I^j is $\emptyset$ for $j > 0$ and $I \neq T_0$).

We obtain a total communication cost of

– when $I \neq T_0$,

$$\mathsf{Cost} = \underbrace{(t+1)\cdot 2\lambda}_{h_1,h_2,\ldots,h_{t+1}} + (\underbrace{\mathsf{inputs}}_{\Delta w,\Delta\beta^1,\ldots,} + \underbrace{\mathsf{comm}}_{\alpha^1,\ldots,\alpha^t} + \underbrace{\lambda \cdot \log_2 \binom{N}{\ell}}_{\mathsf{seed}_T \text{ for } T\neq I} + \underbrace{2\lambda}_{\mathsf{com}_I^1}).$$

– when $I = T_0$,

$$\mathsf{Cost} = \underbrace{(t+1)\cdot 2\lambda}_{h_1,h_2,\ldots,h_{t+1}} + (\underbrace{\mathsf{comm}}_{\alpha^1,\ldots,\alpha^t} + \underbrace{\lambda \cdot \log_2 \binom{N}{\ell}}_{\mathsf{seed}_T \text{ for } T\neq I} + \underbrace{t\cdot 2\lambda}_{\mathsf{com}_I^1,\ldots,\mathsf{com}_I^t}).$$

where inputs denote the bitsize of $(\Delta w, \Delta\beta^1, \ldots, \Delta\beta^t)$, and where comm denotes the bitsize of $(\alpha^1, \ldots, \alpha^t)$.

To achieve a soundness error of $2^{-\lambda}$, one must repeat the protocol τ times such that $\epsilon^\tau < 2^{-\lambda}$. The resulting averaged cost (in bits) is the following:

$$\mathsf{Cost} = (t+1)\cdot 2\lambda + \tau \cdot \left(\frac{\binom{N}{\ell} - 1}{\binom{N}{\ell}} \cdot \mathsf{inputs} + \mathsf{comm} + \lambda \cdot \log_2 \binom{N}{\ell} + \frac{\binom{N}{\ell} - 1 + t}{\binom{N}{\ell}} \cdot 2\lambda \right).$$

Comparison. Let us first consider the case $\ell = 1$. We can check that Protocol 1 (with $\ell = 1$) achieves *exactly the same communication cost and soundness* as the MPCitH transformation with additive sharing and GGM tree (see, e.g., [FR22, Section 3.2]). Moreover, in Protocol 1,

– the prover emulates $\ell + 1 = 2$ parties and
– the verifier emulates $\ell = 1$ party.

This is to be compared with $1 + \log_2 N$ (for the prover) and $\log_2 N$ (for the verifier) using the MPCitH transform with additive sharing speed up with the hypercube technique of [AGH+23]. We thus obtain a proof system with the same communication and soundness but always faster than the recent MPCitH schemes accelerated with the hypercube technique.

Moreover, our result can be argued to be optimal in terms of party emulation: the verifier could not verify less than one party (to get a sound proof) and the prover must emulate strictly more parties than those opened to the verifier (to achieve the zero-knowledge property). We present in Section 3.3 a detailed comparison between the TCitH framework using pseudo-random sharings and GGM tree (abbreviated *TCitH-GGM* in the following) and the former approach based on a Merkle tree recalled in Section 2.5 (and abbreviated *TCitH-MT* in the following).

Remark 3. When repeating the protocol τ times to achieve a negligible soundness error, we obtain a proof system that emulates 2τ parties in total for the prover. However, if the used MPC protocol has a negligible false positive probability p, we can use the trick proposed in the Limbo proof system [DOT21] which consists in using the same MPC challenges $\varepsilon^1, \ldots, \varepsilon^t$ across the τ parallel executions. In that case, we get exactly the same plain values for the hints and the broadcast. Since one of the two party executions per repetition is the plain MPC computation, we can make it once for all the repetitions. The overall MPC emulations for the prover thus consist in emulating only $1 + \tau$ parties.

Case of $\ell > 1$. To compare the cases $\ell = 1$ and $\ell > 1$, let us analyze the communication cost and the running times with respect to a given soundness error $2^{-\lambda_0}$ for a single repetition:

- *Communication cost.* We can assume that

$$\frac{\binom{N}{\ell} - 1}{\binom{N}{\ell}} \approx 1 \quad \text{and} \quad \frac{\binom{N}{\ell} - 1 + t}{\binom{N}{\ell}} \approx 1$$

  for all ℓ. Moreover, we can observe that the seed trees have 2^{λ_0} leaves in both cases. We thus get that the communication cost is the same for any ℓ (up to the above approximations).

- *Running times.* The size of the seed trees is the same in both cases and there is the same number of commitments. The difference in the computation mainly comes from the MPC emulation: we need to emulate $1 + \ell$ parties (for the prover).

To sum up, taking $\ell > 1$ leads to slower schemes while keeping the communication cost unchanged. So the best choice is always to take the minimal value for ℓ.

The only good reason to take $\ell > 1$ would be to bypass the constraint on the number of parties. Remind that we have the limitation that the number N of parties should be less than the field size ($N \leq |\mathbb{F}|$) which, for a small field, might prevent reaching the target per-repetition soundness error $2^{-\lambda_0}$. While increasing ℓ, we can thus amplify the single repetition soundness with a limited N. While this approach is relevant, we show another way to handle the case of the small fields in the next section which achieves better soundness-performance trade-offs.

3.2 Lifting in a Field Extension

As explained previously, the TCitH framework suffers the constraint that the number N of parties should be smaller than the field size: $N \leq |\mathbb{F}|$ (or $N \leq |\mathbb{F}| + 1$ in some cases) as long as $\ell < N - 1$ (see Lemma 1 in [FR22]).[7] This limitation is an issue when dealing with statements defined over small fields (and in particular the binary field $\mathbb{F}_2$). A natural idea to overcome this limitation is to lift the sharing in a field extension.

Lifting in a field extension. Let us take η such that $N \leq |\mathbb{F}|^\eta$ and consider the field extension $\mathbb{K} \equiv \mathbb{F}[\delta]/f(\delta)$ where f is a public irreducible degree-η polynomial. We tweak a bit the sharing generation of Section 3.1. After expanding all $s_T \in \mathbb{F}^{|w|}$ for all $T \in S_\ell^N$, we still compute the auxiliary value Δw as

$$\Delta w := w - \sum_{T \in S_\ell^N} s_T \in \mathbb{F}^{|w|},$$

but we compute the shares $[\![w]\!]_i$ using Equation (3) with parties' evaluation points $\{e_j\}$ living in the field extension $\mathbb{K}$ (instead of living in $\mathbb{F}$). As consequence, the shares $\{[\![w]\!]_i\}_i$ live in $\mathbb{K}^{|w|}$ instead of $\mathbb{F}^{|w|}$. Let us stress that the security properties still hold using this tweak:

- Zero-knowledge: the seed seed_I remains hidden as previously, and so the plain value w is masked by the hidden value $s_I := \mathrm{PRG}(\mathsf{seed}_I)$.
- Soundness: the extractor of the proof of Theorem 1 (see Section 4) outputs the witness even if the polynomials live in a field extension.

[7] Note that $\ell = N - 1$ is equivalent to a trivial linear secret sharing (e.g., the additive secret sharing) and is not of interest for the TCitH framework which benefits from small values of ℓ.

Performances. The communication cost remains *unchanged* since the proof transcript only contains auxiliary values and plain values which still live in the base field $\mathbb{F}$. Regarding the computational cost for the prover, we can remark that:

- The cost of running the plain protocol (Step 3(f) of Protocol 1) remains unchanged, since the plain values still live in $\mathbb{F}$;
- The cost of running the MPC protocol on the ℓ other coefficients of the Shamir's polynomials (Step 3(g) of Protocol 1) is bigger. It is exactly η times bigger as we can decompose this computation into η smaller independent computations living in the base field. Indeed, by denoting $A_0, \ldots, A_j$ the matrices and b the vector underlying the definition of φ^j, which is $\varphi^j : (v_0, \ldots, v_j) \mapsto A_0 \cdot v_0 + \cdots + A_j \cdot v_j + b$ and by denoting $x_{|d}$ the $\mathbb{F}$-coordinate of $x \in \mathbb{K}$ corresponding to the coefficient of the term δ^{d-1} when decomposing $x \in \mathbb{K}$ in the $\mathbb{F}$-basis $(1, \delta, \ldots, \delta^{\eta-1})$, we have:

$$\begin{aligned}
\mathsf{coeff}_i(P_{\alpha^j})_{|d} &= \varphi^j\big(\mathsf{coeff}_i(P_w), (\mathsf{coeff}_i(P_{\beta^k}))_{k\le j}\big)_{|d} \\
&= (A_0 \cdot \mathsf{coeff}_i(P_w))_{|d} + \sum_{k\le j} (A_k \cdot \mathsf{coeff}_i(P_{\beta^k}))_{|d} + \mathsf{coeff}_i(P_b) \\
&= A_0 \cdot (\mathsf{coeff}_i(P_w)_{|d}) + \sum_{k\le j} A_k \cdot (\mathsf{coeff}_i(P_{\beta^k})_{|d}) + \mathsf{coeff}_i(P_b) \\
&= \varphi^j\big(\mathsf{coeff}_i(P_w)_{|d}, (\mathsf{coeff}_i(P_{\beta^k})_{|d})_{k\le j}\big)
\end{aligned}$$

  which holds since the coefficients of $A_0, \ldots, A_j$ live in $\mathbb{F}$.

The same analysis also holds for the verifier: emulating the parties is η times more expensive. To sum up, when lifting in a field extension of degree η, we obtain the same communication cost, but emulating the MPC protocol is as expensive as emulating

- $1 + \ell \cdot \eta$ parties for the prover,
- $\ell \cdot \eta$ parties for the verifier.

We can observe that taking $\eta = 1$ corresponds to the zero-knowledge proof system of the previous section. Taking η larger does not change the communication cost but the emulation phase becomes more expensive. Despite this overhead, taking η larger than 1 can overcome the limitation of $N \le |\mathbb{F}|$. Indeed, we can now execute the proof system with at most $|\mathbb{K}| := |\mathbb{F}|^\eta$ parties. For instance, if the witness (and MPC computation) is defined over $\mathbb{F}_{16}$ and if one wants to take $N = 256$, one just needs to take $\eta = 2$. One then gets 3 party emulations for the prover (instead of 2) and 2 party emulations for the verifier (instead of 1) while squaring the affordable number of parties. Since the proof system is slower with larger η, the optimal strategy is to choose the minimal η satisfying $N \le |\mathbb{F}|^\eta$.

Let us remark that lifting in a field extension of degree $\eta > 1$ is more efficient to deal with small fields than the alternative solution with $\ell > 1$ (described in the previous section). Indeed, when targeting the same soundness error, both cases have similar communication costs, but the lifting tweak requires fewer emulations. This is illustrated in Figure 1 for the field $\mathbb{F}_{13}$.

We describe hereafter a way to further speed up the lifting tweak with $\ell = 1$ using a hypercube structure for the generation of shares.

Hypercube sharing generation. A straightforward execution of the routine GenerateSharingPolynomial to build the corresponding Shamir's polynomial $P_x(X) = c \cdot X + x$ with

$$c := \frac{1}{e_N} \Delta x + \sum_{i=1}^{N} \frac{1}{e_i} s_i$$

involves around N scalar multiplications between a value from $\mathbb{K}$ and a vector from $\mathbb{F}^{|x|}$, or equivalently $N \cdot \eta$ scalar multiplications between a value from $\mathbb{F}$ and a vector from $\mathbb{F}^{|x|}$. However, we can pack these

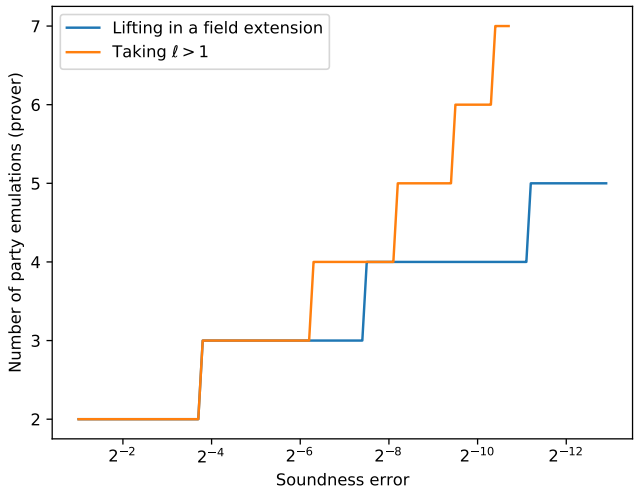


Fig. 1: Emulation cost when working on $\mathbb{F}_{13}$ using lifting and using $\ell > 1$.

multiplications by defining the parties' evaluation points $\{e_j\}_j$ as follows. First, we index these points over $[1 : N_1] \times \ldots \times [1 : N_\eta]$ where $N_1, \ldots, N_\eta$ are some parameters satisfying $N = N_1 \cdot \ldots \cdot N_\eta$. Then, for $\boldsymbol{i} \in [1 : N_1] \times \ldots \times [1 : N_d]$, we define $e_{\boldsymbol{i}}$ such that

$$\frac{1}{e_{\boldsymbol{i}}} = \frac{1}{e'_{i_1}} + \frac{1}{e'_{i_2}} \cdot \delta + \ldots + \frac{1}{e'_{i_\eta}} \cdot \delta^{\eta-1} \in \mathbb{K},$$

where $\{e'_j\}_j$ are distinct points over $\mathbb{F}^* \cup \{\infty\}$. With this definition, we get that c can be computed as

$$\begin{aligned}
c &= -\sum_{\boldsymbol{i} \in [1:N_1] \times \ldots \times [1:N_\eta]} \frac{1}{e_{\boldsymbol{i}}} s_{\boldsymbol{i}} \\
&= -\sum_{\boldsymbol{i} \in [1:N_1] \times \ldots \times [1:N_\eta]} \left(\frac{1}{e'_{i_1}} + \frac{1}{e'_{i_2}} \delta + \ldots + \frac{1}{e'_{i_\eta}} \delta^{\eta-1} \right) s_{\boldsymbol{i}} \\
&= -\sum_{k=1}^{\eta} \left(\sum_{\boldsymbol{i} \in [1:N_1] \times \ldots \times [1:N_\eta]} \frac{1}{e'_{i_k}} s_{\boldsymbol{i}} \right) \delta^{k-1} \\
&= -\sum_{k=1}^{\eta} \left(\sum_{j=1}^{N_k} \frac{1}{e'_j} \sum_{\boldsymbol{i}: i_k = j} s_{\boldsymbol{i}} \right) \delta^{k-1}
\end{aligned}$$

which involved only $N_1 + \ldots + N_\eta$ multiplications instead of $\eta \cdot N = \eta \cdot N_1 \cdot \ldots \cdot N_\eta$ and around $\eta \cdot N$ additions. The routine $\mathsf{GeneratePartyShare}_i$ is also impacted: on inputs $(s_j)_{j \neq i}$ and Δx, $\mathsf{GeneratePartyShare}_i$ outputs $[\![x]\!]_{\boldsymbol{i}}$ where

$$[\![x]\!]_{\boldsymbol{i}} := e_{\boldsymbol{i}} \cdot \sum_{k=1}^{\eta} \left(\sum_{v=1, v \neq i_k}^{N_k} \left(\frac{1}{e'_{i_k}} - \frac{1}{e'_v} \right) \sum_{\boldsymbol{j}: j_k = v} s_{\boldsymbol{j}} \right) \cdot \delta^{k-1} + \begin{cases} \Delta x \cdot (1 - \frac{e_{\boldsymbol{i}}}{e_{(N_1, \ldots, N_\eta)}}) & \text{if } i \neq N, \\ 0 & \text{otherwise.} \end{cases}$$

Remark 4. Let us remark that the extreme case of $\eta = \log_2 N$ and $N_1 = \ldots = N_\eta = 2$ corresponds to the standard additive-sharing MPCitH framework with hypercube optimization from [AGH+23] (since for

$(\ell, N) = (1, 2)$ a Shamir's secret sharing is equivalent to an additive sharing). Whenever the base field is $\mathbb{F}_2$, this gives the best we can hope for with our approach. Whenever the field is larger, our "pseudorandom sharing + lifting" TCitH framework always brings a better trade-off.

Remark 5. One may wonder what is the best choice for $N_1, \ldots, N_\eta$ given N and $|\mathbb{F}|$. For instance, working on $\mathbb{F}_{131}$ with $N = 512$, one could take $(N_1, N_2) = (32, 16)$ or $(N_1, N_2) = (128, 4)$. The only place where the choice of $(N_1, \ldots, N_\eta)$ has an impact is for the computation of the leading coefficient in the routine GenerateSharingPolynomial. As explained before, this step involves around $N_1 + \ldots + N_\eta$ multiplications, thus the best choice consists in minimizing $N_1 + \ldots + N_\eta$. The AM-GM inequality implies $(N_1 + \ldots + N_\eta)/\eta \geq \sqrt[\eta]{N_1 \cdot \ldots \cdot N_\eta}$ which, together with $N = N_1 \cdot \ldots \cdot N_\eta$, gives us:

$$N_1 + \ldots + N_\eta \geq \eta \cdot \sqrt[\eta]{N} \ .$$

We deduce that taking N_i as close as possible to $\sqrt[\eta]{N}$ leads to the optimal computational cost.

3.3 Global Comparison

In Table 1, we compare the following MPCitH/TCitH-based zero-knowledge proof systems:

- the traditional additive-sharing MPCitH framework [KKW18, BN20], where the prover emulates all the N parties in their head;
- the additive-sharing MPCitH framework with hypercube optimization [AGH+23], where the prover only emulates $1 + \log_2 N$ parties;
- the original TCitH framework [FR22] using Merkle tree commitments;
- the alternative TCitH framework using pseudo-random sharings and GGM trees, proposed in the previous section.

For all these proof systems, we give the number of party emulations for the prover and the verifier, while achieving a soundness error of $2^{-\lambda}$ with N parties. Moreover, we give the overall complexity of the prover and the verifier, which includes the emulation cost but also the cost of generation and commitment of the input sharings.

	Additive-sharing MPCitH		TCitH			
			Merkle tree variant		GGM tree variant ($\ell = 1$)	
	Traditional	Hypercube	$\ell = 1$	Any ℓ	Basic	With lifting
Number of Emulations (prover)	$\approx \lambda \frac{N}{\log_2 N}$	$\approx \lambda \frac{\log_2 N + 1}{\log_2 N}$	$\approx \lambda \frac{2}{\log_2 N}$	$\approx \lambda \frac{\ell+1}{\log_2 \binom{N}{\ell}}$	$\lambda \cdot \frac{2}{\log_2 N}$	$\lambda \cdot \frac{1+\eta}{\log_2 N}$
Time Complexity (prover)	$O(\lambda \frac{N}{\log_2 N})$	$O(\lambda N)$	$O(\lambda \frac{N}{\log_2 N})$	$O(\lambda \frac{N \cdot \ell}{\log_2 N})$	$O(\lambda \frac{N}{\log_2 N})$	$O(\lambda \frac{N \cdot \eta}{\log_2 N})$
Number of Emulations (verifier)	$\approx \lambda \frac{N-1}{\log_2 N}$	$\approx \lambda \frac{\log_2 N}{\log_2 N}$	$\approx \lambda \frac{1}{\log_2 N}$	$\approx \lambda \frac{\ell}{\log_2 \binom{N}{\ell}}$	$\lambda \cdot \frac{1}{\log_2 N}$	$\lambda \cdot \frac{\eta}{\log_2 N}$
Time Complexity (verifier)	$O(\lambda \frac{N}{\log_2 N})$	$O(\lambda N)$	$O(\lambda)$	$O(\lambda \cdot \ell)$	$O(\lambda \frac{N}{\log_2 N})$	$O(\lambda \frac{N \cdot \eta}{\log_2 N})$
Restriction	-	-	$N \leq \lvert\mathbb{F}\rvert$	$N \leq \lvert\mathbb{F}\rvert$	$N \leq \lvert\mathbb{F}\rvert$	$\sqrt[\eta]{N} \leq \lvert\mathbb{F}\rvert$

Table 1: Computational complexities for the existing MPCitH-based transformations, when achieving a soundness error of $2^{-\lambda}$ (assuming a negligible false positive probability p).

We can make the following observations:

- The smallest emulation costs are achieved by the TCitH framework. In fact, in TCitH, taking N larger reduces the emulation cost, while it was the opposite in the traditional MPCitH framework.

– All the protocols have a prover complexity around $O(\lambda \frac{N}{\log_2 N})$ even when the emulation cost is small because of the generation and the commitment of sharings. We deduce that the latter shall be the computational bottleneck for the TCitH framework (unless relying on heavy MPC protocols). When working on large fields, the hypercube approach has the largest overall asymptotic complexity of $O(\lambda N)$. We note that for small fields, TCitH has similar asymptotic complexity since one needs to take $\ell \approx \eta = \log N$.
– The TCitH framework with GGM tree is strictly better than the additive-sharing MPCitH framework with hypercube optimization as soon as the base field has more than two elements (both are equivalent on $\mathbb{F}_2$).
– The original TCitH framework is the only zero-knowledge proof system achieving fast verification (thanks to the Merkle trees). However, the original TCitH framework has a slightly larger communication cost than the other protocols (due to Merkle authentication paths vs. GGM sibling paths as explained in Section 2.5).

3.4 Application to NIST Post-Quantum Signature Candidates

In the recent NIST call for additional post-quantum signature schemes, 7 submissions fit the MPCitH framework:[8] AIMer [KHS+22, CCH+23], Biscuit [BKPV23], MIRA [ABC+23, ABB+23d], MiRitH [ARZV23, ABB+23b], MQOM [FR23], RYDE [BCF+23, ABB+23c] and SDitH [FJR22, AFG+23]. We fetched the source codes of all these submissions, applied our alternative TCitH framework, and compared with the former approaches. The resulting running times are given in Table 2.

Let us stress that we only fetched the source codes relative to the MPC protocols (and the arithmetic parts). For the rest of the implementation, we used a factorized source code implementing the MPCitH transformations. We are thus relying on the same source code for the symmetric components (pseudorandom generation, commitments, ...), leading to a fairer comparison. In addition, we can rely on exactly the same transformations for the three compared approaches. For example, in MiRitH, an implementation with the hypercube optimization is provided but it emulates $2\log_2 N$ parties while an optimal implementation only requires $1+\log_2 N$ party emulations. In our benchmark, the running times given for MiRitH with hypercube correspond to an emulation of $1 + \log_2 N$ parties. In our source code, the pseudo-randomness is generated using SHAKE and the hash function is instantiated with SHA3. We have benchmarked all the codes on a 3.8 GHz Intel Core i7 CPU with support of AVX2 and AES instructions. All the reported timings were measured on this CPU while disabling Intel Turbo Boost.

As expected, we can observe the TCitH framework does not lead to faster algorithms for AIMer and RYDE, since the latter have the binary field $\mathbb{F}_2$ as base field. When working on larger fields, the TCitH framework with GGM tree always leads to faster timings: the heavier the underlying MPC protocol, the larger the gain. For instance, for MIRA (which uses the heaviest MPC protocol among the submissions), the TCitH framework halves the running times.

Let us further stress that the timing improvements obtained thanks to the TCitH framework with GGM tree tend to flatten the MPC protocol contributions in the NIST candidate timings and hence significantly lessen the timing differences between the candidates. While the running times are in the range 4.5–344.3 ms for the traditional approach with $N = 256$, they are in the range 3.22–9.89 ms with the TCitH framework.

4 Extended TCitH Framework and Applications

This section presents our extended TCitH framework. We start by formalizing the MPC model for our extended framework (as an adaptation of the model from [FR22]), then we describe our extended TCitH proof system in two variants (Merkle tree vs. GGM tree), and finally give several applications and implementation results.

[8] PERK [ABB+23a] follows the shared-permutation framework [FJR23] which differs from the standard MPCitH framework.

			Additive MPCitH		TCitH (GGM tree)		
Scheme	N	Size	Traditional	Hypercube	η	Our scheme	Saving
AIMer	16	5 904	0.64	0.52	4	0.52	−0%
	256	4 176	4.53	3.22	8	3.22	−0%
Biscuit	16	6 726	2.81	1.71	1	1.44	−16%
	256	4 758	17.71	4.65	2	4.24	−9%
MIRA	32	7 376	74.95	15.02	2	8.04	−46%
	256	5 640	384.26	20.11	2	9.89	−51%
MiRitH-Ia	16	7 661	6.81	2.59	1	1.52	−41%
	256	5 665	54.15	6.60	2	5.42	−18%
MiRitH-Ib	16	8 800	11.22	4.04	1	2.11	−48%
	256	6 298	89.50	8.66	2	6.66	−23%
MQOM-gf31	32	7 621	12.88	4.64	1	3.31	−29%
	256	6 348	96.41	11.27	2	8.74	−22%
MQOM-gf251	32	7 809	8.56	3.05	1	2.16	−29%
	256	6 575	44.11	7.56	1	5.97	−21%
RYDE	32	7 446	2.31	1.14	5	1.14	−0%
	256	5 956	12.41	4.65	8	4.65	−0%
SDitH-gf256	32	11 515	16.85	4.90	1	3.07	−37%
	256	8 241	78.37	7.23	1	5.31	−27%
SDitH-gf251	32	11 515	4.17	2.17	1	1.79	−18%
	256	8 241	19.15	7.53	1	6.44	−14%

Table 2: Benchmark of all the NIST MPCitH-based signature schemes, for the three approaches. The sizes are in bytes and the timings are in milliseconds. The given timings correspond to the signing time, but the verification time is always very close to the signing time (since the verifier makes almost the same computation as the prover).

4.1 MPC Model

We consider an MPC protocol that performs its computation on a base finite field $\mathbb{F}$ so that all the manipulated variables (including the witness w) are tuples of elements from $\mathbb{F}$. In what follows, the sizes of the different tuples involved in the protocol are kept implicit for the sake of simplicity. For our extended Threshold-Computation-in-the-Head (TCitH) framework, we consider the following MPC model. As in the MPC model formalized in [FR22] and overviewed in Section 2.3, the parties jointly run the computation of a function

$$f(w, \boldsymbol{\varepsilon}, \boldsymbol{\beta}) = \begin{cases} \textsc{Accept} & \text{if } g(\boldsymbol{\alpha}) = 0, \\ \textsc{Reject} & \text{otherwise,} \end{cases} \tag{5}$$

with $\boldsymbol{\varepsilon} := (\varepsilon^1, \ldots, \varepsilon^t)$ the random values from a randomness oracle $\mathcal{O}_R$, $\boldsymbol{\beta} := (\beta^1, \ldots, \beta^t)$ the hints from a hint oracle $\mathcal{O}_H$, $\boldsymbol{\alpha} := (\alpha^1, \ldots, \alpha^t)$ the broadcasted and publicly recomputed values, and g some final check function. The main differences with the previous model are the following:

1. The considered protocols exclusively apply to Shamir's secret sharings of the form

   $$[\![v]\!] = ([\![v]\!]_1, \ldots, [\![v]\!]_N) := (P_v(e_1), \ldots, P_v(e_N))$$

   as defined in Section 2.1. We recall that such a sharing is a $(d+1, N)$-threshold secret sharing whenever the polynomial P_v is of degree $\deg(P_v) \leq d$, implying that $d+1$ shares are sufficient to reconstruct P_v and hence recover $v = P_v(0)$.

2. The round computation functions $\Phi^1, \ldots, \Phi^t$ (which are used to compute the broadcast values $\alpha^1, \ldots, \alpha^t$) are not restricted to be $\mathbb{F}$-linear but can be polynomial functions of higher degrees.

The latter difference implies that the sharings computed by the parties during the protocol can be of higher degrees than ℓ (the degree of the input witness sharing). In the following, we denote

$$\deg([\![v]\!]) = \deg(P_v)$$

the degree of the polynomial P_v underlying a Shamir's secret sharing $[\![v]\!]$, also called the degree of $[\![v]\!]$. While the input sharing of the protocol is a fresh degree-ℓ sharing, the computation of non-linear round functions φ might produce sharings $[\![\alpha]\!]$ higher degrees.

Protocol ingredients. The considered MPC protocol is as follows. At the start, the parties receive as input a fresh degree-ℓ Shamir's secret sharing $[\![w]\!]$ of the witness w (one share per party). Then the parties process one or several rounds of the following actions:

- **Receiving randomness:** The parties receive a random value (or random tuple) $\varepsilon \in \mathbb{F}^{|\varepsilon|}$ from a randomness oracle $\mathcal{O}_R$. When calling this oracle, all the parties get the same random value ε. Upon application of the TCitH transform, these random values are provided by the verifier as challenges.
- **Receiving hint:** Optionally, the parties receive a sharing $[\![\beta]\!]$ from a hint oracle $\mathcal{O}_H$. For some function ψ, the plain hint β is computed as $\beta := \psi(w, \boldsymbol{\varepsilon}; r)$ where $\boldsymbol{\varepsilon} = (\varepsilon^1, \varepsilon^2, \ldots)$ is made of the previous random values from $\mathcal{O}_R$ and where r is fresh randomness. A fresh $(d+1, N)$-Shamir's secret sharing of β is then generated and distributed to the parties (one share per party), for some hint degree d (which might be different from ℓ). Upon application of the TCitH transform, the hint $[\![\beta]\!]$ is generated and committed by the prover.
- **Computing & broadcasting:** The parties compute $[\![\alpha]\!] := \varphi([\![w]\!], [\![\boldsymbol{\beta}]\!], [\![\theta]\!])$, which means that they locally compute
$$[\![\alpha]\!]_i := \varphi([\![w]\!]_i, [\![\boldsymbol{\beta}]\!]_i, [\![\theta]\!]_i) \ , \ \forall i \in [1:N]$$
where $[\![\boldsymbol{\beta}]\!] = ([\![\boldsymbol{\beta^1}]\!], [\![\boldsymbol{\beta^2}]\!], \ldots)$ is made of the previous outputs of $\mathcal{O}_H$ and $[\![\theta]\!]$ is a fixed (publicly known) sharing. The parties then broadcast $[\![\alpha]\!]$ and publicly recompute α.

  The function φ is any multivariate polynomial function over $\mathbb{F}$ whose coefficients possibly depend on the previously broadcasted values and the previous random values from $\mathcal{O}_R$. (Similarly, the fixed shares of $[\![\theta]\!]$ possibly depend on these previously broadcasted or random values.) Let $d = \deg([\![\alpha]\!])$, the degree of the obtained sharing which depends on the degrees of the input sharings $[\![w]\!], [\![\boldsymbol{\beta}]\!], [\![\theta]\!]$ as well as on the (multivariate) degree of the function φ. We have that $[\![\alpha]\!]$ is a $(\ell, d+1, N)$-quasi-threshold secret sharing of α (which is a $(\ell+1, N)$-threshold secret sharing if and only if $d = \ell$). Upon application of the TCitH transform, the prover computes $[\![\alpha]\!]$ from the previously committed shares (as well as previous broadcasted values and random values) and sends $d+1$ shares of $\boldsymbol{\alpha}$ to the verifier (since $d+1$ shares are necessary to fully reconstruct the sharing $\boldsymbol{\alpha}$).

  Example 1. A broadcast value could be computed as
$$[\![\alpha]\!] := [\![w_1]\!] \cdot [\![w_2]\!]$$
with $w_1, w_2 \in \mathbb{F}$ two coordinates of w. Here the function φ is simply the product of the two first coordinates of the witness. This product is computed sharewisely: $[\![\alpha]\!]_i := [\![w_1]\!]_i \cdot [\![w_2]\!]_i$ for every i. The obtained sharing $[\![\alpha]\!]$ has underlying polynomial $P_\alpha := P_{w_1} \cdot P_{w_2}$, with P_{w_1}, P_{w_2} the polynomials underlying the sharings $[\![w_1]\!], [\![w_2]\!]$. We hence have $\deg([\![\alpha]\!]) = 2\ell$. Upon application of the TCitH framework, the prover must communicate $2\ell + 1$ shares of $[\![\alpha]\!]$ to allow the verifier to reconstruct the full sharing.

At the end of the protocols, the parties evaluate a function g of the publicly recomputed values $\alpha^1, \ldots, \alpha^t$. They output Accept if $g(\alpha^1, \ldots, \alpha^t) = 0$ and Reject otherwise.

General protocol description. We consider two notions of rounds in our MPC model. The MPC protocol is composed of one or several *outer rounds*. The first outer round starts with the parties receiving the input sharing and possibly a first sharing from the hint oracle. It is then composed of a call to the randomness oracle and one or several *inner rounds* of computing and broadcasting. Then the protocol either finishes with the computation of g, or the parties call the hint oracle. In the latter case, a new outer round begins with a

call to the randomness oracle followed by one or several inner rounds. In the MPCitH or TCitH paradigm, a new outer round begins each time the prover needs to commit a new sharing. Namely, an outer round in the MPC protocol translates to a pair of commit-challenge communication rounds in the zero-knowledge protocol.

Successive rounds of computing and broadcasting are called *inner rounds*. In outer round $j \in [1:t]$, the parties do $t_j^{(in)}$ successive rounds of locally computing and broadcasting $[\![\alpha^{j,k}]\!] = \varphi^{j,k}([\![w]\!], [\![\beta^1]\!], \ldots, [\![\beta^j]\!], [\![\theta^{j,k}]\!])$ for $k \in [1 : t_j^{(in)}]$. This enables each function $\varphi^{j,k}$ to depend on previously recomputed values $\alpha^{j,1}, \ldots, \alpha^{j,k-1}$. This notably gives a way to compute or verify non-linear (high degree) functions although the $\varphi^{j,k}$ functions might be linear (or low degree) – see for instance the product-check protocol of [BN20]. We shall denote by Φ^j the global iterative functions obtained from those $t_j^{(in)}$ inner rounds:

$$[\![\alpha^j]\!] = ([\![\alpha^{j,1}]\!], \ldots, [\![\alpha^{j,t_j^{(in)}}]\!]) = \Phi^j([\![w]\!], [\![\beta^1]\!], \ldots, [\![\beta^j]\!]) \ ,$$

where the fixed sharings $[\![\theta^{j,k}]\!]$ are "hardcoded" in the definition of Φ^j.

Following this structure, our general MPC protocol is depicted in Protocol 2.

1. The parties take as input an $(\ell+1, N)$-Shamir's secret sharing $[\![w]\!]$.
2. For $j = 1$ to t (outer rounds):
   (a) For some function ψ^j and some sharing degree d_j^β, the parties get a fresh $(d_j^\beta + 1, \ell)$-Shamir's secret sharing $[\![\beta^j]\!]$ from the hint oracle $\mathcal{O}_H$, such that
   $$\beta^j \leftarrow \psi^j(w, \varepsilon^1, \ldots, \varepsilon^{j-1}; r^j)$$
   for a uniform random tape r^j.
   (b) The parties get a common random ε^j from the oracle $\mathcal{O}_R$.
   (c) Inner rounds: The parties locally compute and broadcast
   $$[\![\alpha^j]\!] := \Phi^j([\![w]\!], [\![\beta^1]\!], \ldots, [\![\beta^j]\!])$$
   and publicly recompute α^j.
   This step is detailed in Protocol 3.
3. The parties finally accept if $g(\alpha^1, \ldots, \alpha^t) = 0$ and reject otherwise.

Protocol 2: General MPC protocol for extended TCitH framework.

False positive probability. The functionality computed by the protocol deterministically depends on the broadcasted values $\boldsymbol{\alpha}$ (through the function g), which in turn deterministically depend on the input witness w, the sampled random values $\boldsymbol{\varepsilon}$, and the hints $\boldsymbol{\beta}$. It is formally given by the function f from Equation (5), with $\boldsymbol{\alpha} = \Phi(w, \boldsymbol{\varepsilon}, \boldsymbol{\beta})$ where Φ is the deterministic function mapping $(w, \boldsymbol{\varepsilon}, \boldsymbol{\beta})$ to $\boldsymbol{\alpha}$ (defined by the coordinate functions $\Phi^1, \ldots, \Phi^t$). This function f aims at checking the validity of a witness w for a statement x with respect to some relation $\mathcal{R}$, namely checking that $(x, w) \in \mathcal{R}$. As in the MPC model of [FR22], we restrict our model to MPC protocols for which the function f satisfies the following properties:

– If w is a *good witness*, namely w is such that $(x, w) \in \mathcal{R}$, and if the hints $\boldsymbol{\beta}$ are genuinely sampled as $\beta^j \leftarrow \psi^j(w, \varepsilon^1, \ldots, \varepsilon^{j-1}; r^j)$ for every j, then the protocol always accepts. More formally:

$$\Pr_{\boldsymbol{\varepsilon}, \boldsymbol{r}}\left[f(w, \boldsymbol{\varepsilon}, \boldsymbol{\beta}) = \textsc{Accept} \;\middle|\; \begin{array}{c} (x, w) \in \mathcal{R} \\ \forall j, \beta^j \leftarrow \psi^j(w, \varepsilon^1, \ldots, \varepsilon^{j-1}; r^j) \end{array} \right] = 1.$$

(c) For $k = 1$ to $t_j^{(in)}$ (inner rounds):

- For some $\mathbb{F}$-polynomial function $\varphi^{j,k}$, the parties compute:

$$[\![\alpha^{j,k}]\!] := \varphi^{j,k}([\![w]\!], [\![\beta^1]\!], \ldots, [\![\beta^j]\!], [\![\theta^{j,k}]\!])$$

  where $[\![\theta^{j,k}]\!]$ is some fixed sharing.
- The parties broadcast $[\![\alpha^{j,k}]\!]$ and publicly reconstruct α^j.

NB: *The coefficients of the function $\varphi^{j,k}$ possibly depend on $\varepsilon^1, \ldots, \varepsilon^j$, $\alpha^1, \ldots, \alpha^{j-1}$ and $\alpha^{j,1}, \ldots, \alpha^{j,k-1}$.*

NB: *We denote $[\![\alpha^j]\!] = ([\![\alpha^{j,1}]\!], \ldots, [\![\alpha^{j,t_j^{(in)}}]\!])$ and $\Phi^j = ([\![\varphi^{j,1}]\!], \ldots, [\![\varphi^{j,t_j^{(in)}}]\!])$, with $[\![\theta^{j,k}]\!]$ "hardcoded" in Φ^j so that $[\![\alpha^j]\!] := \Phi^j([\![w]\!], [\![\beta^1]\!], \ldots, [\![\beta^j]\!])$.*

Protocol 3: General MPC protocol: inner rounds.

- If w is a *bad witness*, namely w is such that $(x, w) \notin \mathcal{R}$, then the protocol rejects with probability at least $1 - p$, for some constant probability p which is called the *false positive probability*. The latter holds even if the hints $\boldsymbol{\beta}$ are not genuinely computed. More formally, for any (adversarially chosen) deterministic functions $\chi^1, \ldots, \chi^t$, we have:

$$\Pr_{\boldsymbol{\varepsilon},\boldsymbol{r}}\left[f(w, \boldsymbol{\varepsilon}, \boldsymbol{\beta}) = \textsc{Accept} \;\middle|\; \begin{array}{c}(x, w) \notin \mathcal{R} \\ \forall j, \beta^j \leftarrow \chi^j(w, \varepsilon^1, \ldots, \varepsilon^{j-1}; r^j)\end{array}\right] \leq p.$$

We say that a *false positive* occurs whenever the MPC protocol outputs Accept on input a bad witness w, which occurs with probability at most p.

4.2 Extended TCitH Framework

We describe hereafter our extended framework of Threshold Computation in the Head (TCitH). The main difference with the original framework is the support of non-linear MPC round functions. We further propose a tweak of the original TCitH framework in the way to deal with the commitment of hints in protocols with multiple outer rounds. Our extended framework comes in two variants, namely TCitH with Merkle tree (TCitH-MT) as the original framework, and TCitH with GGM tree (TCitH-GGM) as presented in Section 3.

Tweaking hint commitments. The proof system described in the original TCitH framework [FR22] makes use of a different Merkle tree to commit the witness sharing $[\![w]\!]$ (together with first hint $[\![\beta^1]\!]$) and each hint sharing $[\![\beta^j]\!]$ in next outer rounds. In total, the resulting proof system thus uses t Merkle trees. We propose here the following tweak: while generating the sharings $[\![w]\!]$ and $[\![\beta^1]\!]$ in the first round, the prover also generates and commits the sharings $[\![\bar{\beta}^2]\!], \ldots, [\![\bar{\beta}^t]\!]$ of uniformly random values $\bar{\beta}^2 \in \mathbb{F}^{|\beta^2|}, \ldots, \bar{\beta}^t \in \mathbb{F}^{|\beta^t|}$ using *the same Merkle tree*. In the following rounds, to generate and commit a sharing of the j^{th} hint β^j, the prover just needs to compute a *hint correction* $\Delta\beta^j$ as $\Delta\beta^j := \beta^j - \bar{\beta}^j$ and they can then deduce a sharing $[\![\beta^j]\!]$ of β^j using

$$[\![\beta^j]\!] \leftarrow [\![\bar{\beta}^j]\!] + \Delta\beta^j.$$

This tweak presents three advantages:

- It only requires one Merkle tree instead of t, the communication cost induced by the authentication paths is thus decreased by a factor t. However, to reveal $[\![\beta^j]\!]_I$, the prover now needs to reveal $[\![\bar{\beta}^j]\!]_I$ and $\Delta\beta^j$ (instead of just $[\![\beta^j]\!]_I$). The global communication cost is smaller as soon as sending $\Delta\beta^j$ is cheaper than sending an authentication path, which is often the case.
- It allows to have symmetry between both variants, TCitH-MT and TCitH-GGM. In TCitH-GGM, by committing the seed tree in the first round, we are naturally committing random sharing $[\![\bar{\beta}^2]\!], \ldots, [\![\bar{\beta}^t]\!]$.

– To obtain sound proof in the MPCitH-MT variant, we will need what we call a *degree-enforcing commitment scheme* to make sure that the committed sharings are of the right degrees. This requires an additional challenge-response round for each sharing commitment. Using the above tweak, this additional round is performed a single time (after the initial Merkle tree commitment) instead of t.

Proof system blueprint. For both variants, the proof system arising from our extended framework runs as follows:

1. The prover generates and commits the witness sharing $[\![w]\!]$, a first hint $[\![\beta^1]\!]$ and $t-1$ random sharings $[\![\bar{\beta}^2]\!] \ldots, [\![\bar{\beta}^t]\!]$; In the TCitH-MT variant, an additional degree-enforcement round of challenge and response is performed (see details below);
2. The verifier generates the randomness ε^1 as challenge;
3. The prover runs the inner rounds of computing and broadcasting *in their head* and commits the broadcast shares $[\![\alpha^1]\!]$ to the verifier;
4. For each j from 2 to t:
   (a) The prover generates and commits the hint correction $\Delta\beta^j$;
   (b) The verifier generates the randomness ε^j as challenge;
   (c) The prover runs the inner rounds of computing and broadcasting *in their head* and commits the broadcast shares $[\![\alpha^j]\!]$ to the verifier;
5. The verifier generates a random subset $I \subseteq [1:N]$ of cardinality $|I| = \ell$ as challenge;
6. The prover sends to the verifier: the shares $[\![w]\!]_I$, $[\![\beta^1]\!]_I$, $[\![\bar{\beta}^2]\!]_I, \ldots, [\![\bar{\beta}^t]\!]_I$ (with hint corrections $\Delta\beta^2, \ldots, \Delta\beta^t$), the sharings $[\![\alpha^1]\!]$, $\ldots$, $[\![\alpha^t]\!]$.
7. The verifier checks:
   – the commitments of the open shares $[\![w]\!]_I, [\![\beta^1]\!]_I, \{[\![\bar{\beta}^j]\!]_I, \Delta\beta^j\}_{j\geq 2}$ and of the broadcast sharing $[\![\alpha^1]\!]$, $\ldots$, $[\![\alpha^t]\!]$;
   – the correct computation of the shares $[\![\alpha]\!]_I$ from $[\![w]\!]_I$ (and $[\![\beta^1]\!]_I$, $\ldots$, $[\![\beta^t]\!]_I$);
   – that $g(\alpha^1, \ldots, \alpha^t) = 0$ (*i.e.* that the protocol accepts).

The generation and commitment of shares in Step 1 (as well as their opening in Step 6) depend on the variant (MT vs. GGM – see details below). In Steps 3 and 4(c), the commitment of the sharing $[\![\alpha^j]\!]$ is done by hashing the $d_j^\alpha + 1$ first shares and sending the obtained hash $h_j = \mathrm{Hash}([\![\alpha^j]\!]_{[1:d_j^\alpha+1]})$ to the verifier, where $d_j^\alpha = \deg([\![\alpha^j]\!])$. Then, in Step 6, the opening of $[\![\alpha^j]\!]$ simply consists in revealing the shares $[\![\alpha^j]\!]_S$ for some set S such that $|S| = d_j^\alpha + 1 - \ell$ and $S \cap I = \emptyset$. In Step 7, the verifier recomputes the shares $[\![\alpha^j]\!]_I$ from $[\![w]\!]_I$ (and $[\![\beta^1]\!]_I, \ldots, [\![\beta^t]\!]_I$), then reconstructs the shares $[\![\alpha^j]\!]_{[1:d_j^\alpha+1]}$ from the shares $[\![\alpha^j]\!]_{I\cup S}$ to finally check the correctness of the hash h_j. This process checks at the same time the correct computation of the shares $[\![\alpha]\!]_I$ and the commitment of the sharing $[\![\alpha^j]\!]$.

Degree-enforcing commitment scheme (TCitH-MT). As explained in Section 3.1 (see the "Analysis" paragraph) one advantage of the TCitH-GGM framework is to enforce the commitment of a valid Shamir's secret sharing (of a possibly incorrect witness), while in the TCitH-MT framework a malicious prover might commit an invalid sharing (for which the shares do not correspond to the evaluations of a degree-ℓ polynomial). The latter issue results in a degradation of the soundness which would further amplify in the extended framework due to the use of higher degree sharings. To avoid this issue in our extended TCitH-MT framework, we tweak the sharing commitment scheme to make it *degree enforcing*, as described hereafter.

We describe hereafter a way to constrain the prover to commit a valid Shamir's secret sharing of the witness (*i.e.*, corresponding to a sharing of degree ℓ). Our technique uses the idea of the test of interleaved linear codes (a.k.a. proximity test) for Reed-Solomon codes proposed in Ligero [AHIV17]. However, we use

different parameters which crucially allows us to ensure a stronger soundness result. In Ligero (improved with subsequent analysis from [BCI+20]), the test ensures that committed codewords have a distance lower than $\delta/2$ from genuine codewords (where δ is the minimum distance of the underlying code) with possibly non-negligible soundness error. In our context, this translates to ensuring that a committed sharing supposed to be of degree ℓ has a distance lower than $(N-\ell)/2$ (i.e., less than $(N-\ell)/2$ non-equal evaluations) to some degree-ℓ sharing. Instead, we ensure that the committed sharings are exactly of the expected degree (which can be ℓ or larger), with a tunable soundness error (which can be made negligible).

We first explain the basic principle ignoring hint commitments for the sake of simplicity. At the beginning, the witness is extended with a random vector $u \in \mathbb{F}^\eta$ so that the extended witness (u, w) is shared and committed. Once $[\![u]\!], [\![w]\!]$ have been committed by the prover, the verifier samples a random matrix $\Gamma = (\gamma_{j,k})_{j,k}$ of dimensions $\eta \times |w|$. The prover then computes the sharing

$$[\![\xi]\!] = \Gamma \cdot [\![w]\!] + [\![u]\!] \ , \tag{6}$$

namely the sharing defined as $[\![\xi]\!]_i = \Gamma \cdot [\![w]\!]_i + [\![u]\!]_i$ for all $i \in [1:N]$, and commits it to the prover (by sending its hash value). The sharing $[\![\xi]\!]$ will be later revealed to the verifier which can then check that $[\![\xi]\!]$ is of degree ℓ and that the revealed shares well satisfy (6). This ensures that the committed sharings $[\![u]\!], [\![w]\!]$ were of degree ℓ with high probability. The sharing $[\![u]\!]$ is used to ensure the zero-knowledge property by masking $[\![\xi]\!]$ so that revealing $[\![\xi]\!]$ does not leak any information on w.

When hints are used, we must further ensure that the committed sharings $[\![\beta^1]\!]$, $[\![\bar{\beta}^2]\!]$, …, $[\![\bar{\beta}^t]\!]$ are of the right degrees, which might be different for the different hints. Let d_j^β denote the degree of the hint $[\![\beta^j]\!]$ and let $d_{max}^\beta = \max(\ell, d_1^\beta, \ldots, d_t^\beta)$. Our goal is to enforce that the polynomials P_w, P_{β_1}, …, P_{β_t} underlying the committed sharings are of right degrees $\deg(P_w) = \ell$, $\deg(P_{\beta_j}) = d_j^\beta$ for all $j \in [1:t]$. Let us stress that, in its basic form explained above, the degree enforcement consists in sending a polynomial $P_\xi := \Gamma \cdot P_w + P_u$ to the verifier. To extend this to further polynomials with different degrees, we shall align all the polynomials to the degree d_{max}^β by multiplying each polynomial P by the monomial $X^{d_{max}^\beta - \deg(P)}$. Namely, we define the global vector polynomial $Q \in (\mathbb{F}[X])^{|Q|}$ as

$$Q(X) := \left(X^{d_{max}^\beta - \ell} \cdot P_w(X) \mid X^{d_{max}^\beta - d_1^\beta} \cdot P_{\beta_1}(X) \mid \cdots \mid X^{d_{max}^\beta - d_t^\beta} \cdot P_{\beta_t}(X)\right) \tag{7}$$

which is of length $|Q| = |w| + |\beta^1| + \cdots + |\beta^t|$. In this general setting, the mask sharing $[\![u]\!]$ randomly generated and committed by the prover is of degree d_{max}^β and the random matrix Γ generated by the verifier is of dimensions $\eta \times |Q|$. The revealed degree-enforcing polynomial $P_\xi \in (\mathbb{F}[X])^\eta$ is then defined as

$$P_\xi := \Gamma \cdot Q + P_u \ . \tag{8}$$

In the above equation, the dot product $\Gamma \cdot Q$ is to be interpreted coefficient-wise: $\mathsf{coeff}_i(P_\xi) = \Gamma \cdot \mathsf{coeff}_i(Q) + \mathsf{coeff}_i(P_u)$ for all $i \in [1 : d_{max}^\beta]$, where $\mathsf{coeff}_i(P_\xi) \in \mathbb{F}^\eta$ (resp. $\mathsf{coeff}_i(P_u) \in \mathbb{F}^\eta$, $\mathsf{coeff}_i(Q) \in \mathbb{F}^{|Q|}$) is the vector composed of the ith coefficient of each coordinate polynomial of P_ξ (resp. P_u, Q).

To wrap-up, our degree-enforcement commitment scheme works as follows:

1. The prover generates the sharing of the witness $[\![w]\!]$, the sharing of the random mask $[\![u]\!]$, the sharing of the first hint $[\![\beta^1]\!]$ and sharings $[\![\bar{\beta}^2]\!]$, …, $[\![\bar{\beta}^t]\!]$ of uniform random vectors $\bar{\beta}^j \in \mathbb{F}^{|\beta^j|}$ for all $j \in [2:t]$.
2. The prover commits these sharings using a Merkle tree. Specifically, they compute the leaf commitments $\mathsf{com}_i := \mathrm{Com}([\![w]\!]_i, [\![u]\!]_i, [\![\beta^1]\!]_i, [\![\bar{\beta}^2]\!]_i, \ldots, [\![\bar{\beta}^t]\!]_i; \rho_i)$ and the root $h_1 := \mathsf{MerkleRoot}(\mathsf{com}_1, \ldots, \mathsf{com}_N)$ and send h_1 to the verifier.
3. The verifier samples a random matrix Γ of dimensions $\eta \times |Q|$ where $|Q| = |w| + |\beta^1| + \cdots + |\beta^t|$ and sends it to the prover.
4. The prover computes the degree-enforcing polynomial $P_\xi \in (\mathbb{F}[X])^\eta$ using Equation (8) and sends $h'_1 := \mathrm{Hash}(P_\xi)$ to the verifier.

The rest of the protocol runs as overviewed above with the following tweaks. During the opening phase, the prover further reveal $P_\xi(e_i)$ for all $i \in S$ with S some set of cardinality $|S| = d^\beta_{max} + 1 - \ell$ and disjoint of I (the set of opened shares). During the final checks, the verifier computes $P_\xi(e_i) = \Gamma \cdot Q(e_i) + P_u(e_i)$ for all $i \in I$ from opened shares $[\![w]\!]_i, [\![u]\!]_i, [\![\beta^1]\!]_i, [\![\bar{\beta}^2]\!]_i, \ldots, [\![\bar{\beta}^t]\!]_i$ (by definition $Q(e_i)$ and $P_u(e_i)$ are linear functions of these shares). From $\{P_\xi(e_i)\}_{i \in S \cup I}$ the verifier reconstructs P_ξ by interpolation and check the hash $h'_1 = \mathrm{Hash}(P_\xi)$.

Pseudorandom generation of high-degree sharings (TCitH-GGM). In Section 3, we explain how Shamir's secret sharings of degree ℓ can be pseudorandomly generated and committed (in a ℓ-private way) using a GGM tree with $\binom{N}{\ell}$ leaves. For the extended TCitH-GGM framework, we need to generate and commit Shamir's secret sharings of degrees possibly greater than ℓ for the hints. We explain hereafter how to adapt this ℓ-private pseudorandom generation to the case of higher degree sharings.

To generate a pseudorandom degree-d sharing $[\![x]\!]$, the expanded randomness s_T is of length $(d-\ell+1)\cdot|x|$. Then the underlying polynomial P_x is defined as

$$P_x(X) = \Delta x \cdot P_{T_0,1}(X) + \sum_{T \in S^N_\ell} \sum_{k \in [1:d-\ell+1]} s^{(k)}_T \cdot P_{T,k}(X) \ \in \big(\mathbb{F}[X]\big)^{|x|},$$

while the recovery of the ith party share $[\![x]\!]_i$ from $(\{s_T\}_{T:i \notin T}, \Delta x)$ is defined as:

$$[\![x]\!]_i := \sum_{T \in S^N_\ell, i \notin T} \sum_{k \in [1:d-\ell+1]} s^{(k)}_T \cdot P_{T,k}(e_i) + \begin{cases} \Delta x \cdot P_{T_0,1}(e_i) & \text{if } i \notin T_0, \\ 0 & \text{otherwise.} \end{cases},$$

where

- for all T, $s_T := (s^{(1)}_T, \ldots, s^{(d-\ell+1)}_T)$;
- for all (T,k), $P_{T,k}$ is the degree-d polynomial satisfying

$$\begin{cases} P_{T,k}(e'_k) = 1 & \\ P_{T,k}(e'_j) = 0 & \text{for all } j \in [1 : d-\ell+1] \backslash \{k\} \\ P_{T,k}(e_j) = 0 & \text{for all } j \in T \end{cases}$$

  with $\{e_j\}_j$ and $\{e'_j\}_j$ two disjoint sets of distinct field elements with $e'_1 = 0$.

Protocol description. The zero-knowledge protocol obtained by applying our extended TCitH framework to the general MPC protocol (Protocol 2) is formally described in Protocol 4. The way the shares are generated and committed (as well as opened and decommitted) depends on the variant (TCitH-GGM vs. TCitH-MT). The formal description hence makes use of four variant-dependent routines:

- GenerateAndCommitShares: This routine takes as input the witness w, the first hint β^1, and a root seed rseed, and it generates the sharings $[\![w]\!]$, $[\![\beta^1]\!]$ of w and β^1, the random sharings $[\![u]\!], [\![\bar{\beta}^2]\!], \ldots, [\![\bar{\beta}^t]\!]$, and a commitment h_1 of these sharings.
- OpenShares: This routine takes as input the witness w, the root seed rseed, the hint β^1, the hint corrections $\{\Delta\beta^j\}_{j \geq 2}$ and a set $I \subseteq [1:N]$ such that $|I| = \ell$, and it returns views_I an opening of the shares in I as well as decom_I the necessary data to decommit views_I (namely to check the consistency of views_I with the commitment h_1). In the TCitH-GGM framework, views_I is defined as the sibling path of the leaf with index I, concatenated with $(\Delta w, \Delta\beta^1, \ldots, \Delta\beta^t)$ when $I \neq T_0$, and decom_I is defined as the commitment com_I (the leaf which cannot be recomputed from the sibling path). In the TCitH-MT framework, views_I is defined as all the shares $[\![w]\!]_I, [\![\beta^1]\!]_I, [\![\bar{\beta}^2]\!]_I, \ldots, [\![\bar{\beta}^t]\!]_I$ and the hint corrections $\Delta\beta^2, \ldots, \Delta\beta^t$, and decom_I is defined as the authentication paths of the open commitments $\{\mathsf{com}_i\}_{i \in I}$ in the Merkle tree.

1. The prover samples a root seed $\mathsf{rseed} \in \{0,1\}^\lambda$, compute the plain hint $\beta^1 = \psi(w; r^1)$ with $r^1 \leftarrow \mathrm{PRG}(\mathsf{rseed})$, and computes:

$$([\![w]\!], [\![u]\!], [\![\beta^1]\!], [\![\bar{\beta}^2]\!], \ldots, [\![\bar{\beta}^t]\!], h_1) \leftarrow \mathsf{GenerateAndCommitShares}(w, \beta^1, \mathsf{rseed}) \ .$$

   The prover sends h_1 to the verifier.

   In the TCitH-MT variant, the prover and verifier additionally perform the following steps:
   (a) The verifier samples a random matrix Γ from $\mathbb{F}^{n \times |w|}$ and sends it to the prover;
   (b) The prover computes $P_\xi := \Gamma \cdot Q + P_u$ where Q is computed from the polynomials of the sharings $([\![w]\!], [\![\beta^1]\!], [\![\bar{\beta}^2]\!], \ldots, [\![\bar{\beta}^t]\!])$ using (7) and P_u is the polynomial of $[\![u]\!]$. The prover sends $h'_1 := \mathrm{Hash}(P_\xi)$ to the verifier.

2. The verifier samples at random a challenge ε^1 and sends it to the prover;

3. The prover runs the MPC computation in their head. Specifically, the prover computes the shares

$$[\![\alpha^1]\!]_i = \Phi^1([\![w]\!]_i, [\![\beta^1]\!]_i) \ \ \forall i \in [1 : d_1^\alpha + 1] \ .$$

4. For $j = 2$ to t:
   (a) The prover computes the plain hint $\beta^j = \psi^j(w, \varepsilon^1, \ldots, \varepsilon^{j-1}; r^j)$ with $r^j \leftarrow \mathrm{PRG}(\mathsf{rseed})$ and deduce the hint correction $\Delta\beta^j = \beta^j - \bar{\beta}^j$ and the hint sharing $[\![\beta^j]\!] = [\![\bar{\beta}^j]\!] + \Delta\beta^j$. The prover then computes the hash $h_j = \mathrm{Hash}([\![\alpha^{j-1}]\!]_{[1:d^\alpha_{j-1}+1]}, \Delta\beta^j)$ and sends it to the verifier.
   (b) The verifier samples at random a challenge ε^j and sends it to the prover;
   (c) The prover runs the MPC computation in their head. Specifically, the prover computes the shares

$$[\![\alpha^j]\!]_i = \Phi^j([\![w]\!]_i, [\![\beta^1]\!]_i, \ldots, [\![\beta^j]\!]_i) \ \ \forall i \in [1 : d_j^\alpha + 1] \ .$$

5. The prover computes $h_{t+1} = \mathrm{Hash}([\![\alpha^t]\!]_{[1:d_t^\alpha+1]})$ and sends it to the verifier.

6. The verifier samples at random a subset $I \subset [1 : N]$ of ℓ parties (*i.e.* $|I| = \ell$) and sends it to the prover.

7. The prover reveals the views of all the parties in I. Specifically, the prover computes

$$(\mathsf{views}_I, \mathsf{decom}_I) \leftarrow \mathsf{OpenShares}(w, \mathsf{rseed}, \beta^1, \{\Delta\beta^j\}_{j \geq 2}, I) \ .$$

   The prover sends views_I, decom_I, $\{[\![\alpha^j]\!]_{S^j}\}_{i \in [1:t]}$ to the verifier, where $S^j \subseteq [1 : N]$ is of cardinality $|S^j| = d_j^\alpha + 1 - \ell$ and such that $S^j \cap I = \emptyset$, the prover computes $[\![\alpha]\!]_{S^j}$ from $[\![\alpha]\!]_{[1:d_j^\alpha+1]}$.

   In the TCitH-MT variant, the prover further sends $\{P_\xi(e_i)\}_{i \in S}$ for a set $S \subseteq [1 : N]$ of cardinality $|S| = d^\beta_{max} + 1 - \ell$ and such that $S \cap I = \emptyset$.

8. The verifier performs the following checks:
   – *(Shares' commitment)* First, the verifier checks the opened views vs. the commitment h_1. Namely it computes:

$$\hat{h}_1 \leftarrow \mathsf{VerifyDecommitment}(\mathsf{views}_I, \mathsf{decom}_I, I) \ .$$

     If $\hat{h}_1 \neq h_1$ the verifier stops and outputs Reject.
   – *(Parties' computation)* Then, the verifier computes

$$([\![w]\!]_I, \{[\![\beta^j]\!]_I\}_j) \leftarrow \mathsf{RetrieveShares}(\mathsf{views}_I, I)$$

     and

$$[\![\alpha^j]\!]_i = \Phi^j([\![w]\!]_i, [\![\beta^1]\!]_i, \ldots, [\![\beta^j]\!]_i) \quad \forall i \in I \ \ \forall j \in [1 : t] \ .$$

     For all $j \in [1 : t]$, the verifier recovers the shares $[\![\alpha]\!]_{[1:d_j^\alpha+1]}$ from $[\![\alpha]\!]_{I \cup S^j}$ and checks that $h_{j+1} = \mathrm{Hash}([\![\alpha^j]\!]_{[1:d^\alpha_{j-1}+1]}, \Delta\beta^{j+1})$ (if $j < t$) or $h_{j+1} = \mathrm{Hash}([\![\alpha^j]\!]_{[1:d^\alpha_{j-1}+1]})$ (if $j = t$). If the check fails, the verifier stops and outputs Reject.

     In the TCitH-MT variant, the verifier further computes $P_\xi(e_i) = \Gamma \cdot Q(e_i) + P_u(e_i)$ for all $i \in I$ from opened shares. From $\{P_\xi(e_i)\}_{i \in S \cup I}$ the verifier reconstructs P_ξ by interpolation and check that $h'_1 = \mathrm{Hash}(P_\xi)$. If the check fails, the verifier stops and outputs Reject.
   – *(Protocol outcome)* The verifier recovers the plain broadcast value α from $[\![\alpha]\!]_{I \cup S}$ and checks that $g(\alpha) = 0$. If one of the checks fails, the verifier stops and outputs Reject.

   If none of the above checks failed, the verifier outputs Accept.

Protocol 4: Zero-knowledge protocol: application of the extended TCitH framework to the general MPC protocol (Protocol 2) with a single outer round (no hints).

– VerifyDecommitment: This routine takes as input an opening views_I, some associated decommitment data decom_I and the set I and it recomputes the commitment h_1.
– RetrieveShares: This routine takes as input an opening views_I and the associated set I and returns the witness shares $[\![w]\!]_I$ and the hint shares $\{[\![\beta^j]\!]_I\}_j$.

The formal description of these routines is given in Figure 2. In the formal description of OpenShares, some values must be retrieved from $(w, \beta^1, \mathsf{rseed})$ which have been already computed in GenerateAndCommitShares. We denote this by $(w, \beta^1, \mathsf{rseed}) \mapsto (\ldots)$. Of course, in practice, this computation does not need to be performed twice. Moreover, the routines in Figure 2 rely on GGM trees and Merkle trees. To handle the GGM trees, we denote

– TreePRG the subroutine that expands the seed tree from the root seed,
– $\mathsf{GetSiblingPath}_I$ the subroutine which computes the sibling paths of the leaves indexed by I,
– RetriveLeavesFromPath the subroutine which recomputes all the leaves except those indexed by I from the corresponding sibling paths.

To handle the Merkle tree, we denote

– MerkleRoot the subroutine which computes the root of the Merkle tree for the given leaves,
– GetAuthPath the subroutine that extracts the authentication paths for the leaves indexed by I,
– RetrieveRootFromPath the subroutine which recomputes the root of the Merkle tree from some leaves with their authentication paths.

Security. The completeness, soundness, and zero-knowledge properties of the obtained protocol are stated in the following theorem. The input MPC protocol has the following parameters: the size of the sharings N, the privacy threshold ℓ (the input sharing is $(\ell+1, N)$-threshold and the MPC protocol is ℓ-private), the maximal degree of the broadcast sharings d^{α}_{max}, the maximal degree for the hint sharings d^{β}_{max} (wlog. $d^{\beta}_{max} \leq d^{\alpha}_{max}$), the false positive probability p. The theorem proof is provided in Appendix B.

Theorem 1. *Let Π be an MPC protocol of parameters $(N, \ell, d^{\alpha}_{max}, d^{\beta}_{max}, p)$ complying to the format of Protocol 2. In particular, Π is ℓ-private in the semi-honest model and of false positive probability p. Then, Protocol 4 built from Π satisfies the three following properties:*

– ***Completeness.*** *A prover $\mathcal{P}$ who knows a witness w such that $(x, w) \in \mathcal{R}$ and who follows the steps of the protocol always succeeds in convincing the verifier $\mathcal{V}$.*
– ***Soundness.*** *Suppose that there is an efficient prover $\tilde{\mathcal{P}}$ that, on input x, convinces the honest verifier $\mathcal{V}$ to accept with probability*

$$\tilde{\epsilon} := \Pr[\langle \tilde{\mathcal{P}}, \mathcal{V} \rangle(x) \to 1] \;>\; \epsilon$$

where the soundness error ϵ is defined as

$$\epsilon := p + (1-p) \cdot \frac{\binom{d^{\alpha}_{max}}{\ell}}{\binom{N}{\ell}} \qquad \text{for the TCitH-GGM variant,}$$

and

$$\epsilon := p + (1-p) \cdot \frac{\binom{d^{\alpha}_{max}}{\ell}}{\binom{N}{\ell}} + \frac{\binom{N}{d^{\beta}_{max}+1}^2}{|\mathbb{F}|^{\eta}} \qquad \text{for the TCitH-MT variant.}$$

Then, there exists a probabilistic extraction algorithm $\mathcal{E}$ with time complexity in $\mathsf{poly}(\lambda, (\tilde{\epsilon} - \epsilon)^{-1})$ that, given rewindable black-box access to $\tilde{\mathcal{P}}$, outputs either a witness w satisfying $(x, w) \in \mathcal{R}$, a hash collision, or a commitment collision.

– ***Honest-Verifier Zero-Knowledge.*** *Let the pseudorandom generator* PRG *used in Protocol 1 be $(t, \epsilon_{\mathrm{PRG}})$-secure and the commitment scheme* Com *be $(t, \epsilon_{\mathrm{Com}})$-hiding. There exists an efficient simulator $\mathcal{S}$ which, given random challenge I outputs a transcript which is $(t, \epsilon_{\mathrm{PRG}} + \epsilon_{\mathrm{Com}})$-indistinguishable from a real transcript of Protocol 1.*

TCitH-GGM

TCitH-MT

GenerateAndCommitShares$(w, \beta^1, \mathsf{rseed})$:
- $\{\mathsf{seed}_T\}_{T \in S_\ell^N} \leftarrow \mathsf{TreePRG}(\mathsf{rseed})$
- For all T,
  - $s_T^0 \leftarrow \mathrm{PRG}(\mathsf{seed}_T, 0)$
  - $s_T^1 \leftarrow \mathrm{PRG}(\mathsf{seed}_T, 1)$
- $\Delta w \leftarrow w - \sum_T s_T^0$
- $\Delta\beta^1 \leftarrow \beta^1 - \sum_T s_T^1$
- $[\![w]\!] \leftarrow \mathsf{GenerateSharing}(\Delta w, \{s_T^0\}_T, \ell)$
- $[\![\beta^1]\!] \leftarrow \mathsf{GenerateSharing}(\Delta\beta^1, \{s_T^1\}_T, d_1^\beta)$
- For all $j \in [2:t]$,
  - $s_T^j \leftarrow \mathrm{PRG}(\mathsf{seed}_T, j)$
  - $[\![\bar\beta^j]\!] \leftarrow \mathsf{GenerateSharing}(0, \{s_T^j\}_T, d_j^\beta)$
- For all $T \in S_\ell^N$:
  - If $T \neq T_0$,
    - $\mathsf{com}_T \leftarrow \mathrm{Com}(\mathsf{seed}_T; \rho_T^j)$
  - Else
    - $\mathsf{com}_T \leftarrow \mathrm{Com}(\mathsf{seed}_T, \Delta w, \Delta\beta^1; \rho_T^j)$
- $h_1 \leftarrow \mathrm{Hash}(\{\mathsf{com}_T\}_T)$
- Return $([\![w]\!], \emptyset, [\![\beta^1]\!], [\![\bar\beta^2]\!], \ldots, [\![\bar\beta^t]\!], h_1)$

GenerateAndCommitShares$(w, \beta^1, \mathsf{rseed})$:
- $\{r_k^0\}_{k\in[1:\ell]} \leftarrow \mathrm{PRG}(\mathsf{rseed}, 0)$
- $\{r_k^1\}_{k\in[1:d_1^\beta]} \leftarrow \mathrm{PRG}(\mathsf{rseed}, 1)$
- $u, \{r_k^u\}_{k\in[1:d_{max}^\beta]} \leftarrow \mathrm{PRG}(\mathsf{rseed}, -1)$
- For all $i \in [1:N]$:
  - $P_w(X) = w + \sum_{k=1}^{\ell} r_k^0 \cdot X^k$
  - $P_{\beta^1}(X) = \beta^1 + \sum_{k=1}^{d_1^\beta} r_k^1 \cdot X^k$
  - $P_u(X) = u + \sum_{k=1}^{d_{max}^\beta} r_k^u \cdot X^k$
- For all $j \in [2:t]$,
  - $\bar\beta^j, \{r_k^j\}_{k\in[1:d_j^\beta]} \leftarrow \mathrm{PRG}(\mathsf{rseed}, j)$
  - For all $i \in [1:N]$:
    - $P_{\bar\beta^j}(X) := \bar\beta^j + \sum_{k=1}^{d_j^\beta} r_k^j \cdot X^k$
- For all $i \in [1:N]$
  - $[\![w]\!]_i \leftarrow P_w(e_i)$
  - $[\![\beta^1]\!]_i \leftarrow P_{\beta^1}(e_i)$
  - $[\![\bar\beta^j]\!]_i \leftarrow P_{\bar\beta^j}(e_i)$ for all $j \geq 2$
  - $\mathsf{com}_i := \mathrm{Com}([\![w]\!]_i \,\|\, [\![\beta^1]\!]_i \,\|\, [\![\bar\beta^2]\!]_i \,\|\, \ldots \,\|\, [\![\bar\beta^t]\!]_i, \rho_i)$
- $h_1 := \mathsf{MerkleRoot}(\mathsf{com}_1, \ldots, \mathsf{com}_N)$
- Return $([\![w]\!], [\![u]\!], [\![\beta^1]\!], [\![\bar\beta^2]\!], \ldots, [\![\bar\beta^t]\!], h_1)$

OpenShares$(w, \mathsf{rseed}, \beta^1, \{\Delta\beta^j\}_{j\geq 2}, I)$:
- $(w, \mathsf{rseed}, \beta^1) \mapsto (\Delta w, \Delta\beta^1, \mathsf{com}_I)$
- $\mathsf{path}_I \leftarrow \mathsf{GetSiblingPath}_I(\mathsf{rseed})$
- $\mathsf{views}_I \leftarrow (\mathsf{path}_I, \Delta w, \Delta\beta^1, \ldots, \Delta\beta^t)$
- $\mathsf{decom}_I \leftarrow \mathsf{com}_I$
- Return $(\mathsf{views}_I, \mathsf{decom}_I)$

OpenShares$(w, \mathsf{rseed}, \beta^1, \{\Delta\beta^j\}_{j\geq 2}, I)$:
- $(w, \mathsf{rseed}, \beta^1) \mapsto ([\![w]\!], [\![\beta^1]\!], [\![\bar\beta^2]\!], \ldots, [\![\bar\beta^t]\!], \{\mathsf{com}_i\}_i)$
- $\mathsf{views}_I \leftarrow ([\![w]\!]_I, [\![\beta^1]\!]_I, [\![\bar\beta^2]\!]_I, \ldots, [\![\bar\beta^t]\!]_I, \Delta\beta^2, \ldots, \Delta\beta^t)$
- $\mathsf{decom}_I \leftarrow \mathsf{GetAuthPath}(\{\mathsf{com}_i\}_i, I)$
- Return $(\mathsf{views}_I, \mathsf{decom}_I)$

VerifyDecommitment$(\mathsf{views}_I, \mathsf{decom}_I, I)$:
- $(\mathsf{path}_I, \Delta w, \Delta\beta^1, \ldots, \Delta\beta^t) \leftarrow \mathsf{views}_I$
- $\{\mathsf{seed}_T\}_{T\neq I} \leftarrow \mathsf{RetriveLeavesFromPath}(\mathsf{path}_I)$
- $\mathsf{com}_I \leftarrow \mathsf{decom}_I$
- For all $T \neq I$:
  - If $T \neq T_0$
    - $\mathsf{com}_T \leftarrow \mathrm{Com}(\mathsf{seed}_T; \rho_T^j)$
  - Else
    - $\mathsf{com}_T \leftarrow \mathrm{Com}(\mathsf{seed}_T, \Delta w, \Delta\beta^1; \rho_T^j)$
- $h_1 \leftarrow \mathrm{Hash}(\{\mathsf{com}_T^j\}_T)$
- Return h_1

VerifyDecommitment$(\mathsf{views}_I, \mathsf{decom}_I, I)$:
- $([\![w]\!]_I, [\![\beta^1]\!]_I, [\![\bar\beta^2]\!]_I, \ldots, [\![\bar\beta^t]\!]_I, \Delta\beta^2, \ldots, \Delta\beta^t) \leftarrow \mathsf{views}_I$
- For all $i \in I$:
  - $\mathsf{com}_i := \mathrm{Com}([\![w]\!]_i \,\|\, [\![\beta^1]\!]_i \,\|\, [\![\bar\beta^2]\!]_i \,\|\, \ldots \,\|\, [\![\bar\beta^t]\!]_i, \rho_i)$
- $h_1 \leftarrow \mathsf{RetrieveRootFromPath}(\mathsf{decom}_I, \{\mathsf{com}_i\}_{i\in I})$
- Return h_1

RetrieveShares(views_I, I):
- $\mathsf{path}_I \,\|\, (\Delta w, \Delta\beta^1, \ldots, \Delta\beta^t) \leftarrow \mathsf{views}_I$
- $\{\mathsf{seed}_T\}_{T\neq I} \leftarrow \mathsf{RetriveLeavesFromPath}(\mathsf{path}_I)$
- For all $T \neq I$,
  - $s_T^j \leftarrow \mathrm{PRG}(\mathsf{seed}_T, j)$ for $j \in \{0, \ldots, t\}$
- For all $i \in I$,
  - $[\![w]\!]_i \leftarrow \mathsf{GeneratePartyShare}_i((s_T^0)_{T:i\notin T}, \Delta w, \ell)$
  - For all $j \in [1:t]$:
    - $[\![\beta^j]\!]_i \leftarrow \mathsf{GeneratePartyShare}_i((s_T^j)_{T:i\notin T}, \Delta\beta^j, d_j^\beta)$
- Return $([\![w]\!]_I, [\![\beta^1]\!]_I, \ldots, [\![\beta^t]\!]_I)$

RetrieveShares(views_I, I):
- $\mathsf{shares}_I \,\|\, (\Delta\beta^2, \ldots, \Delta\beta^t) \leftarrow \mathsf{views}_I$
- $([\![w]\!]_i, [\![\beta^1]\!]_i, [\![\bar\beta^2]\!]_i, \ldots, [\![\bar\beta^t]\!]_i)_{i\in I} \leftarrow \mathsf{shares}_I$
- For all $i \in I$,
  - If $e_i \neq \infty$,
    - $[\![\beta^j]\!]_i \leftarrow \Delta\beta^j + [\![\bar\beta^j]\!]_i$ for all $j \geq 2$
  - Else,
    - $[\![\beta^j]\!]_i \leftarrow [\![\bar\beta^j]\!]_i$ for all $j \geq 2$
- Return $([\![w]\!]_I, [\![\beta^1]\!]_I, \ldots, [\![\beta^t]\!]_I)$

Fig. 2: Sharing generation and commitment routines.

4.3 General Proof System for Polynomial Constraints

We instantiate the TCitH framework with an MPC protocol verifying for general polynomial constraints. The communication of the obtained proof system only depends on the size of the input of the circuit and the circuit degree (and not of its number of multiplications as many previous MPCitH proof systems).

For a witness $w \in \mathbb{F}^{|w|}$, the considered MPC protocol checks that w satisfies some polynomial relations:

$$\forall j \in [1:m], f_j(w) = 0$$

where $f_1, \ldots, f_m$ are polynomials from $\mathbb{F}[X_1, \ldots, X_{|w|}]$ of total degree at most d. Given some field extension $\mathbb{K}$ of $\mathbb{F}$, the protocol runs as follows:

1. The parties receive a sharing $[\![w]\!]$, with $\deg[\![w]\!] = \ell$.
2. The parties receive a hint $[\![v]\!]$ from $\mathcal{O}_H$, where v is uniformly sampled in $\mathbb{K}$ and $\deg[\![v]\!] = d \cdot \ell - 1$.
3. The parties receive random values $\gamma_1, \ldots, \gamma_m \in \mathbb{K}$ from $\mathcal{O}_R$.
4. The parties locally compute
$$[\![\alpha]\!] \leftarrow [\![v]\!] \cdot [\![0]\!] + \sum_{j=1}^{m} \gamma_j \cdot f_j([\![w]\!])$$
where $[\![0]\!]$ is a publicly-known sharing of 0 such that $\deg[\![0]\!] = 1$.
5. The parties open $[\![\alpha]\!]$ to get α.
6. The parties output Accept if and only if $\alpha = 0$.

It is easy to check that the MPC protocol outputs Accept when w vanishes the polynomials $f_1, \ldots, f_m$ since

$$\alpha = v \cdot 0 + \sum_{j=1}^{m} \gamma_j \cdot f_j(w) = 0 \ .$$

When w does not vanish the polynomials, there exists j' such that $f_{j'}(w) \neq 0$. In that case, α is uniformly random in $\mathbb{K}$ since $\gamma_{j'}$ has been chosen uniformly at random in $\mathbb{K}$. We get that the MPC protocol will output Accept with probability $\frac{1}{|\mathbb{F}|}$, which corresponds to its *false positive probability*. Finally, the ℓ-privacy of the protocol holds for the following reason. In our MPC model (assuming that hints are of degree at least ℓ), the protocol is ℓ-private as long as the broadcast sharings do not leak information on the witness. In the present case, we have that $[\![v]\!] \cdot [\![0]\!]$ is a uniformly-random degree-$(d \cdot \ell)$ sharing of 0, which implies that $[\![\alpha]\!]$ is a uniformly-random degree-$(d \cdot \ell)$ sharing of $\sum_{j=1}^{m} \gamma_j \cdot f_j(w) = 0$. The above MPC protocol is thus ℓ-private.

When applying the TCitH framework to this MPC protocol, by Theorem 1 we obtain a complete, sound and honest-verifier zero-knowledge proof system of soundness error

$$\epsilon := \begin{cases} \frac{1}{|\mathbb{K}|} + \left(1 - \frac{1}{|\mathbb{K}|}\right) \cdot \frac{\binom{d\cdot\ell}{\ell}}{\binom{N}{\ell}} & \text{for the TCitH-GGM variant,} \\ \frac{1}{|\mathbb{K}|} + \left(1 - \frac{1}{|\mathbb{K}|}\right) \cdot \frac{\binom{d\cdot\ell}{\ell}}{\binom{N}{\ell}} + \frac{\binom{N}{\ell+1}^2}{|\mathbb{F}|^\eta} & \text{for the TCitH-MT variant.} \end{cases}$$

As usual, to achieve a targeted soundness error $2^{-\lambda}$, we can perform τ parallel repetitions of the protocol such that $\epsilon^\tau \leq 2^{-\lambda}$. In practice, we shall choose the parameter η of the degree enforcing commitment (TCitH-MT) and field extension $\mathbb{K}$ such that $\binom{N}{\ell+1}^2/(2|\mathbb{F}|^\eta)$ and $1/|\mathbb{K}|$ are both negligible, in order to get $\epsilon \approx \binom{d\cdot\ell}{\ell}/\binom{N}{\ell}$. In particular, we choose η such that $\binom{N}{\ell+1}^2/(2|\mathbb{F}|^\eta) \leq 2^{-\lambda}$. This implies that we can use the *same* degree-enforcing verifier challenge Γ across all the parallel executions (as suggested in Limbo [DOT21]). It results that the value $P_\xi(0)$ sent by the prover for the degree-enforcement check (in TCitH-MT) is the same across all executions and only needs to be included once (instead of τ) in the proof transcript, thus saving communication. We obtain the following communication cost (in bits):

$$\textsc{Size}_{\text{GGM}} = 4\lambda + \tau \cdot \left(\underbrace{\ell \cdot n \cdot \log_2 |\mathbb{F}|}_{[\![w]\!]_I} + \underbrace{(d-1) \cdot \ell \cdot \log_2 |\mathbb{K}|}_{[\![\alpha]\!]} + \underbrace{\lambda \cdot \log_2 N}_{\text{GGM tree}} + 2\lambda \right)$$

for TCitH-GGM, and

$$\textsc{Size}_{\text{MT}} = 6\lambda + \underbrace{(\ell+1)\cdot\eta\cdot\log_2|\mathbb{F}|}_{\text{Sharing-degree testing}} + \tau\cdot\left(\underbrace{\ell\cdot(n\cdot\log_2|\mathbb{F}|+\log_2|\mathbb{K}|)}_{[\![w]\!]_I,[\![v]\!]_I} + \underbrace{(d-1)\cdot\ell\cdot\log_2|\mathbb{K}|}_{[\![\alpha]\!]} + \underbrace{2\lambda\cdot\ell\cdot\log_2\frac{N}{\ell}}_{\text{Merkle tree}}\right)$$

for TCitH-MT.

We can transform this proof system into non-interactive arguments and signatures using the Fiat-Shamir transform. However, since the proof system has 5 rounds when TCitH-GGM and 7 rounds with TCitH-MT, we might take into account potential KZ-like forgery attacks [KZ20a]. In practice, to achieve a security level of λ bits, we propose to select

- the parameter η of the sharing-degree testing such that $\frac{\binom{N}{\ell+1}^2}{2|\mathbb{F}|^\eta} \leq 2^{-\lambda}$, when using TCitH-MT;
- the field extension $\mathbb{K}$ of the proof system such that $\frac{1}{|\mathbb{K}|} \leq 2^{-\lambda}$;
- the number τ of repetitions such that $\left(\frac{\binom{d\cdot\ell}{\ell}}{\binom{N}{\ell}}\right)^\tau \leq 2^{-\lambda}$, where N and ℓ are flexible parameters.

Using those parameters, the resulting scheme achieves λ bits of security in the non-interactive setting (and is safe against KZ-like forgery attacks).

4.4 Arguments of Low-Degree Arithmetic Circuits

The above proof system is particularly efficient when applied to an arithmetic circuit C of low degree d. We obtain a zero-knowledge argument of knowledge of a witness w satisfying $C(w) = y$ for a public output y. We get $m = |y|$ polynomial constraints $f_j(w) = C_j(w) - y_j$, where $y_j \in \mathbb{F}$ denotes jth coordinate of the output y and C_j denotes the subcircuit computing this output coordinate. From the analysis of the above section, the size of the obtained argument is independent of the number of gates in the circuit (and in particular on the number of multiplications) and mainly depends on its degree d.

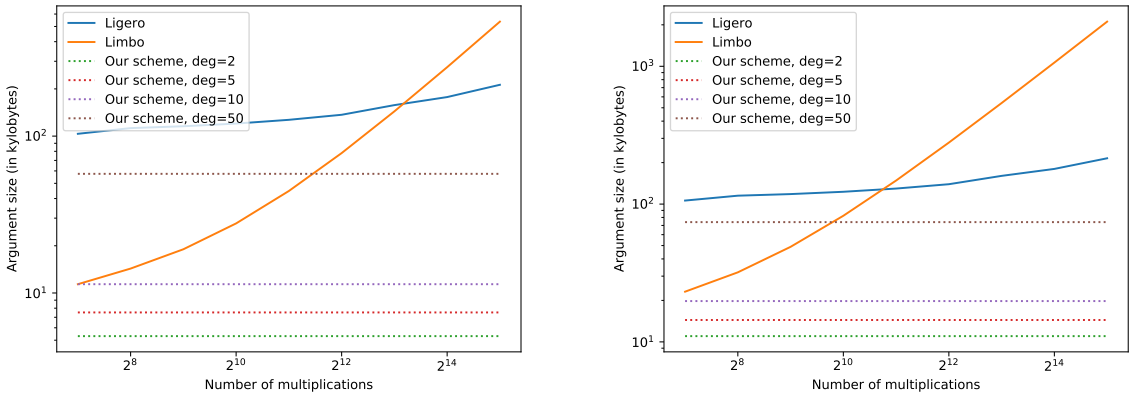
We compare our scheme with other MPCitH-based proof systems performing well on small/average size arithmetic circuits, namely Ligero [AHIV17] and Limbo [DOT21]. Figures 3a and 3b show that our scheme quickly achieves competitive cost when the circuit degree is small/medium (below 50).

4.5 Improved Post-Quantum Signature Schemes

A standard way to build a post-quantum signature scheme is as follows. First, select an (allegedly) post-quantum secure one-way function F. For a random input w of F, the private key is defined as w and the public key is defined as $y = F(w)$. Then use an (allegedly) post-quantum secure zero-knowledge argument to prove knowledge of w satisfying $y = F(w)$ (in a non-interactive message-dependent way). We follow this approach hereafter with our general proof system described in Section 4.3 for classical post-quantum one-way functions, namely the multivariate quadratic problem, as used in MQOM [FR23] and on the syndrome decoding problem, as used in SDitH [FJR22, AFG+23]. In both cases, we explain how to transform the relation $y = F(w)$ into low-degree polynomial constraints.

Multivariate Quadratic (MQ) problem over $\mathbb{F}_q$. Given matrices $A_1, \ldots, A_m \in \mathbb{F}_q^{n\times n}$, vectors $b_1, \ldots, b_m \in \mathbb{F}_q^n$ and scalars $y_1, \ldots, y_m \in \mathbb{F}_q$, the MQ problem consists in finding a vector x such that, for all j, $x^T A_j x + b_j^T x = y_j$. Applying the proof system to this problem is straightforward since it is naturally expressed as degree-2 polynomials. We just need to define the polynomials $f_1, \ldots, f_m$ as

$$\forall j \in [1:m],\ f_j(x) = x^T A_j x + b_j^T x - y_j.$$



(a) Over the field $\mathbb{F}_{256}$

(b) Over the field $\mathbb{F}_{2^{32}}$

Fig. 3: Comparison of our scheme with Ligero [AHIV17] and Limbo [DOT21] for arithmetic circuits (with input of 100 field elements). The curves for Ligero correspond to a lower bound, since we omit the addition gates (which are not for free in Ligero) while estimating the communication cost.

Syndrome Decoding (SD) problem over $\mathbb{F}_q$. Given a matrix $H = (H'|I_{n-k}) \in \mathbb{F}_q^{(n-k)\times n}$ and a vector $y \in \mathbb{F}_q^{n-k}$, the SD problem consists in finding a vector x such that $y = Hx$ and such that x has at most ω non-zero coordinates. Using the arithmetization of the SDitH scheme [FJR22], the SD problem consists in finding a vector $x_A \in \mathbb{F}_q^k$ and a monic degree-ω polynomial $Q(X) := X^\omega + \sum_{i=0}^{\omega-1} Q_i X^i \in \mathbb{F}_q[X]$ such that $x_j \cdot Q(e_j) = 0$ for all j where $x = (x_A \parallel y - H'x_A)$ and $\{e_j\}$ are distinct *public* points of $\mathbb{F}_q$ (requiring that $q \geq n$). We can use the proof system by defining the polynomials $f_1, \ldots, f_m$ as

$$\forall j \in [1:m],\ f_j(x_A, Q) = \left(e_j^\omega + \sum_{i=0}^{\omega-1} Q_i \cdot e_j^i\right) \cdot \begin{cases} (x_A)_j & \text{if } j \leq k, \\ (y - Hx_A)_{j-k} & \text{if } j > k. \end{cases}$$

Performances. Table 3 summarizes the obtained signature sizes and running times (benchmarked on the same platform as before) for the proposed MQ-based and SD-based signature schemes (when taking $\ell = 1$) and compares them to MQOM and SDitH. Our extended TCitH-GGM framework saves 35% and 11% of size for MQOM and SDitH respectively. In particular, our MQ-based scheme achieves signatures of size 4.2 KB for similar (non-structured) MQ instances as the MQOM scheme. This also outperforms Biscuit [BKPV23] which is yet based on a structured MQ problem.

		TCitH-GGM			TCitH-MT		
		Size	Signing	Verif	Size	Signing	Verif
MQ over $\mathbb{F}_{251}$ ($m = n = 43$)	MQOM	6 575 B	5.97 ms	5.57 ms	$\approx$ 13000 B	-	-
	Our scheme	4 257 B	5.23 ms	4.77 ms	6 817 B	3.55 ms	0.63 ms
SD over $\mathbb{F}_{251}$ ($n = 230, k = 126, k = 79$)	SDitH	8 241 B	6.44 ms	6.11 ms	10 117 B	1.55 ms	0.17 ms
	Our scheme	7 335 B	6.73 ms	6.45 ms	9 895 B	4.85 ms	0.30 ms

Table 3: Benchmark for the signature schemes based on MQ and SD problems over $\mathbb{F}_{251}$. The timings of MQOM and SDitH when using TCitH-GGM correspond to the running times of these schemes when integrating the optimization of Section 3 (see Section 3.4 for details). The timings of SDitH using TCitH-MT correspond to the running times of the official (optimized) implementation on the same platform. The authors of MQOM did not propose a variant of their scheme using TCitH-MT, but we give in Table 3 the signature size they would obtain.

Application to other NIST post-quantum signature candidates. As explained in Section 3.4, several MPCitH-based schemes relying on different hardness assumptions have been proposed in the new NIST call for additional post-quantum signatures. We already deal with the case of MQOM and SDitH above (the $\mathbb{F}_{251}$ instance in both cases) but our proof system can also be applied to the other schemes. We provide in Table 4 a list of NIST candidates for which an application of our general proof system of Section 4.3 (TCitH-GGM variant) provides an improvement of the signature size, namely all the MPCitH-based candidates except PERK [ABB+23a] (which is based on the shared-permutation technique [FJR23] and does not fit our framework), AIMer [CCH+23], RYDE [ABB+23c], and MIRA [ABB+23d]. The three latter schemes are defined over very small fields for which our extended framework suffers communication overhead: while the field lifting tweak described in Section 3.2 is free in communication in the presence of linear MPC round functions $\varphi^{j,k}$, using higher degree round functions (as in the proof system of Section 4.3) increases the cost of lifting.[9] We stress that, although the schemes in Table 4 (except SDitH and MQOM on $\mathbb{F}_{251}$) are defined over small fields (of size between 16 and 31) which are not amenable to our framework, we can still obtain some size improvement for these schemes.

As it was the case for the non-structured MQ problem, adapting the NIST candidate Biscuit [BKPV23] is straightforward since it relies on a structured variant of the MQ problem, called the PowAff2 problem, which is directly expressed as degree-2 polynomial constraints on the witness. MiRitH [ABB+23b] relies on the MinRank problem which consists, given $k+1$ matrices $M_0, M_1, \ldots, M_k$, in finding x such that the rank of $E := M_0 + \sum_{j=1}^{k} x_j \cdot M_j \in \mathbb{F}_q^{m \times n}$ is smaller than a public bound r. The idea of MiRitH is to show that the $n-r$ last columns E_R of $E := (E_L \mid E_R)$ can be expressed as a linear combination of the r first columns E_L, namely there exists a matrix $A \in \mathbb{F}_q^{r \times (n-r)}$ such that $E_R = E_L \cdot A$. The latter relation provides us the degree-2 polynomial constraints we can use, assuming A is part of the witness (together with x). MIRA [ABB+23d] is another NIST candidate that relies on the MinRank problem, but using another verification circuit (based on q-polynomials). One could adapt the scheme by finding the underlying degree-2 polynomial constraints (as for MiRitH). However, the size of our proof system only depends on the size of the witness (and not on the number of multiplications involved in the constraints). In MiRitH, the witness is composed of $x \in \mathbb{F}_q^k$ and of the matrix $A \in \mathbb{F}_q^{r \times (n-r)}$. In MIRA, the witness is composed of $x \in \mathbb{F}_q^k$ and of a vector of $\mathbb{F}_{q^n}^r$ (which represents a monic degree-q^r q-polynomial of $\mathbb{F}_{q^n}[X]$). We thus have that adapting MIRA will lead to larger signature sizes than adapting MiRitH (for the same MinRank parameters).

	Original size	Our variant	Saving
Biscuit [BKPV23]	4 758 B	4 352 B	−9%
MiRitH [ABB+23b]	5 665 B	4 998 B	−12%
MQOM (over $\mathbb{F}_{251}$) [FR23]	6 575 B	4 257 B	−35%
MQOM (over $\mathbb{F}_{31}$) [FR23]	6 348 B	4 238 B	−33%
SDitH (over $\mathbb{F}_{251}$ & $\mathbb{F}_{256}$) [FJR22, AFG+23]	8 241 B	7 335 B	−11%

Table 4: Signature sizes for NIST MPCitH-based candidates.

Although no candidate based on the subset sum problem (SSP) has been submitted to the recent NIST call, let us stress that our general proof system of Section 4.3 (TCitH-GGM variant) can also improve the SSP signature scheme proposed in [FMRV22] based on the principle of MPCitH with rejection. While the original scheme is of size 19 100 bytes, our variant achieves a size of 11 981 bytes.

[9] This is because, for the degree-1 case, one only needs to include the plain values of the broadcast α to the proof transcript (which lies on the original field) while in the higher-degree case, one has to further include some shares $[\![\alpha]\!]_i$ which lives in the extended field.

4.6 Short Post-Quantum Ring Signatures

As a last application, we propose a new ring signature scheme. Such a scheme allows a person to sign a message on behalf of a group of people (called a ring) without revealing which member of the ring signed the message. We denote r the size of the ring and consider r public keys $y_1, \dots y_r$ (one per ring member). The important security property of the scheme (beyond the unforgeability) is the anonymity of the signer. Let us denote j^* the secret index of the signer within the ring. To build a ring signature scheme with our framework, we need to build an MPC protocol that checks that the input sharing $[\![w]\!]$ is the private key which corresponds to one public key of the ring, namely y_{j^*}, while keeping j^* private. A way to proceed is to input to the protocol (in addition to the witness sharing $[\![w]\!]$) a sharing $[\![s]\!]$ of a one-hot encoding $s \in \{0,1\}^r$ of j^*. That is $\forall j \in [1:r]$, $s_j = 1$ if $j = j^*$ and $s_j = 0$ otherwise. Then the MPC protocol starts by securely computing a sharing of the right public key as

$$[\![y_{j^*}]\!] = \sum_{j=1}^{r} [\![s_j]\!] \cdot y_j$$

and next checks that $[\![w]\!]$ corresponds to $[\![y_{j^*}]\!]$ using some existing protocol (depending on the underlying one-way function). The main drawback of this strategy is that the signature includes some shares of $[\![s]\!]$ and so its size scales linearly with the size r of the ring. To handle this issue, we propose to use a "multidimensional one-hot encoding" which is composed of d one-hot vectors $s^{(1)}, \dots, s^{(d)}$ of size $\sqrt[d]{r}$ such that

$$s^{(1)}_{j^*_1} = s^{(2)}_{j^*_2} = \cdots = s^{(d)}_{j^*_d} = 1$$

while the other coordinates of the $s^{(j)}$'s equal zero, where $(j^*_1, \dots, j^*_d)$ is the base-$\sqrt[d]{r}$ decomposition of j^* "shifted by one" (which means that $(j^*_1 - 1, \dots, j^*_d - 1)$ is the standard base-$\sqrt[d]{r}$ decomposition of $j^* - 1$; we use this translation because vector indices start from 1). Here d is a parameter of the scheme that we can tune to optimize the signature size. Then we can compute a sharing of the standard one-hot encoding $[\![s]\!]$ from the sharings $[\![s^{(1)}]\!], \dots, [\![s^{(d)}]\!]$ by

$$\forall j,\ [\![s_j]\!] = [\![s^{(1)}_{j_1}]\!] \times \ldots \times [\![s^{(d)}_{j_d}]\!]\ , \tag{9}$$

with $(j_1, \dots, j_d)$ the base-$\sqrt[d]{r}$ decomposition of j "shifted by one". Computing $[\![s]\!]$ with the above method involves a large number of multiplications, but fortunately, using of our proof system of Section 4.3, the obtained signature size is independent of this number of multiplications but solely depends on d. The MPC protocol should further check that the $[\![s^{(j)}]\!]$ corresponds to the sharing of a one-hot vector with $\sqrt[d]{r} - 1$ zeros and 1 one. To this purpose, we further extend the input of the protocol with secret values $\{p_j\}_j$ encoding the position of the non-zero coordinates in the vectors $\{s^{(j)}\}_j$:

$$\forall j \in [1:d],\ \forall k \in [1:\sqrt[d]{r}\,],\quad s^{(j)}_k \neq 0 \ \Leftrightarrow\ p_j = e_k$$

where $\{e_k\}_k$ are public distinct points of $\mathbb{F}$. Then the protocol first checks

$$\forall j \in [1:d],\ \forall k \in [1:\sqrt[d]{r}],\ s^{(j)}_k \cdot (\gamma_k - p_j) = 0, \tag{10}$$

which guarantees that each of the $s^{(j)}$'s has a single non-zero coordinate and hence that s has a single non-zero coordinate, then the protocol checks

$$\sum_{j=1}^{r} s_j = 1\ , \tag{11}$$

to verify that this non-zero coordinate of s equals 1.

To sum up, the MPC protocol takes as input the sharing $[\![w]\!]$ of the witness, the sharings $[\![s^{(1)}]\!], \dots, [\![s^{(d)}]\!]$ of the one-hot vectors, and the sharings $[\![p_1]\!], \dots, [\![p_d]\!]$ of the non-zero position encodings, then runs the following steps:

1. Compute $[\![s]\!]$ from $[\![s^{(1)}]\!], \ldots, [\![s^{(d)}]\!]$ using Equation (9);
2. Compute $[\![y_{j^*}]\!]$ as $\sum_{j=1}^{r} [\![s]\!] \cdot y_j$;
3. Check that $[\![w]\!]$ is a valid witness for $[\![y_{j^*}]\!]$;
4. Check that $[\![s]\!]$ is in the right form by checking Equations (10) and (11).

Using this method, we can build a ring signature from any one-way function. We propose hereafter concrete schemes relying on the MQ and SD problems by adapting the polynomial constraints of the schemes described in Section 4.5 to handle the one-hot vector s. In what follows, we shall denote g_j the degree-d function defined as

$$g_j(s^{(1)}, \ldots, s^{(d)}) = \prod_{i=1}^{d} s_{j_i}^{(i)} \ .$$

Ring signatures from MQ. We shall use the same quadratic equations $\{A_j, b_j\}_j$ for all the users of the ring so that only the solution y of the system shall differ between the different public keys. In that case, we can define the functions as polynomial constraints:

$$\begin{aligned}
&f_j(x, s^{(1)}, \ldots, s^{(d)}, p) = x^T A_j x + b_j^T - \sum\nolimits_{h=1}^{r} g_h(s^{(1)}, \ldots, s^{(d)}) \cdot y_h \quad \forall j \in [1:m] \ , \\
&f'_{j,k}(x, s^{(1)}, \ldots, s^{(d)}, p) = s_k^{(j)} \cdot (\gamma_k - p_j) \quad \forall j \in [1:d] \ \ \forall k \in [1:\sqrt[d]{r}] \ , \\
&f''(x, s^{(1)}, \ldots, s^{(d)}, p) = \sum\nolimits_{h=1}^{r} g_h(s^{(1)}, \ldots, s^{(d)}) - 1 \ .
\end{aligned}$$

All these polynomials have degrees at most $\max\{d, 2\}$, and we can apply the proof system of Section 4.3 (with the Fiat-Shamir transformation) to get the desired signature scheme.

Ring signatures from SD. We shall use the same matrix H for all the users of the ring so that only the syndrome $y = Hx$ shall differ between the different public keys. In that case, we can define the functions as polynomial constraints:

$$\begin{aligned}
&f_j(x_A, Q, s^{(1)}, \ldots, s^{(d)}, p) = \left(z_j^{\omega} + \sum\nolimits_{i=0}^{\omega-1} Q_i \cdot z_j^i \right) \\
&\qquad \times \begin{cases} (x_A)_j & \text{if } j \leq k, \\ (\sum_{h=1}^{r} g_h(s^{(1)}, \ldots, s^{(d)}) \cdot y_h - Hx_A)_{j-k} & \text{if } j > k. \end{cases} \qquad \forall j \in [1:m] \ , \\
&f'_{j,k}(x, s^{(1)}, \ldots, s^{(d)}, p) = s_k^{(j)} \cdot (\gamma_k - p_j) \quad \forall j \in [1:d] \ \ \forall k \in [1:\sqrt[d]{r}] \ , \\
&f''(x, s^{(1)}, \ldots, s^{(d)}, p) = \sum\nolimits_{h=1}^{r} g_h(s^{(1)}, \ldots, s^{(d)}) - 1 \ .
\end{aligned}$$

All these polynomials have degrees at most $d + 1$, and we can apply the proof system of Section 4.3 (with the Fiat-Shamir transformation) to get the desired signature scheme.

Benchmark. We give the obtained signature sizes and running times in Figure 4. We use the same MQ and SD parameters as in the previous section. However, the sizes depend on the field structure:

- when using $\mathbb{F}_{251}$ (no subfield), we need to share the coordinates of $s^{(1)}, \ldots, s^{(d)}$ over $\mathbb{F}_{251}$. Sending such a share costs $\log_2(251)$ bits per coordinate.
- when using $\mathbb{F}_{256}$ (for which the smallest subfield is $\mathbb{F}_2$), we can share the coordinates of $s^{(1)}, \ldots, s^{(d)}$ over $\mathbb{F}_2$, so sending such a share only costs 1 bits per coordinate.

For this reason, we obtain smaller sizes when working on a field extension, but we obtain very competitive sizes in both cases. We benchmarked both schemes over $\mathbb{F}_{251}$ on the same platform as the previous sections. Signing and verification for a ring of 1000 users takes less than 10 ms, which is very competitive compared to the state of the art of post-quantum ring signatures. Running times with $\mathbb{F}_{256}$ would be similar, up to the fact that a multiplication over $\mathbb{F}_{256}$ is slower than a multiplication over $\mathbb{F}_{251}$ on the considered platform

(the field multiplication time is the bottleneck of the scheme). Let us stress that we only implemented and benchmarked the TCitH-GGM variant here. The TCitH-MT variant would give close results in terms of timings (since the bottleneck is the emulation of the MPC protocol which is similar in both variants) but would suffer an increased signature size (of roughly 2KB for a 128-bit security) due to the Merkle tree vs. GGM tree trade-off.

We compare our schemes with the previous post-quantum ring signatures from the state of the art in Table 5. We can remark that we achieve the smallest signature sizes in all the cases with our MQ-based scheme.

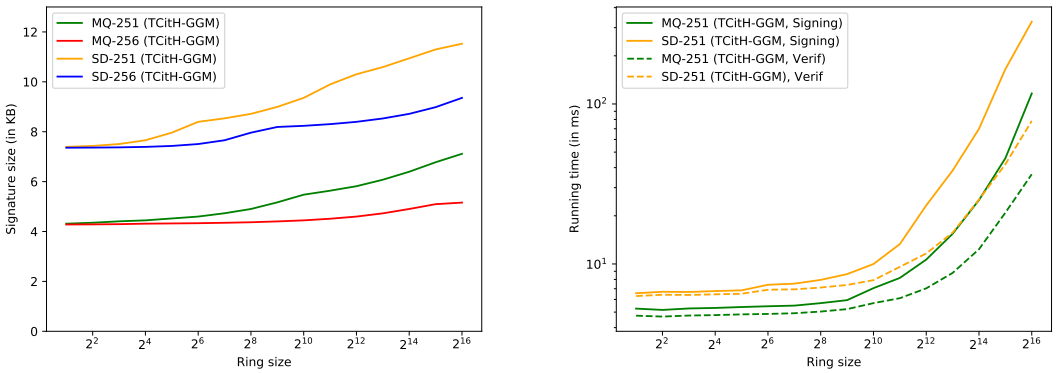


Fig. 4: Benchmark of the proposed ring signature schemes

Table 5: Signature size (KB) for different post-quantum ring signature schemes.

#users		2^3	2^6	2^8	2^{10}	2^{12}	2^{20}	Assumption	Security
Our scheme	2023	4.41	4.60	4.90	5.48	5.82	8.19	MQ over $\mathbb{F}_{251}$	NIST I
Our scheme	2023	4.30	4.33	4.37	4.45	4.60	5.62	MQ over $\mathbb{F}_{256}$	NIST I
Our scheme	2023	7.51	8.40	8.72	9.36	10.30	12.81	SD over $\mathbb{F}_{251}$	NIST I
Our scheme	2023	7.37	7.51	7.96	8.24	8.40	10.09	SD over $\mathbb{F}_{256}$	NIST I
KKW [KKW18]	2018	-	250	-	-	456	-	LowMC	NIST V
GGHK [GGHAK22]	2021	-	-	-	56	-	-	LowMC	NIST V
Raptor [LAZ19]	2019	10	81	333	1290	5161	-	MSIS / MLWE	100 bit
EZSLL [EZS+19]	2019	19	31	-	-	148	-	MSIS / MLWE	NIST II
Falafl [BKP20]	2020	30	32	-	-	35	-	MSIS / MLWE	NIST I
Calamari [BKP20]	2020	5	8	-	-	14	-	CSIDH	128 bit
LESS [BBN+22]	2022	11	14	-	-	20	-	Code Equiv.	128 bit
MRr-DSS [BESV22]	2022	27	36	64	145	422	-	MinRank	NIST I

5 Comparison to VOLE-in-the-Head

During the early stages of our work, a concurrent work has been released which introduced the VOLE-in-the-Head (VOLEitH) framework [BBdSG+23]. This work proposes a method to compile a zero-knowledge

protocol in the VOLE-hybrid model into a publicly verifiable protocol. As mentioned by the authors "like MPCitH, VOLE-in-the-head proofs are based on standard symmetric cryptographic primitives and are publicly verifiable." An interesting question is how much VOLEitH is related to MPCitH? We show hereafter that VOLEitH can be interpreted as an MPCitH construction, more precisely, as a particular case of our TCitH framework. Our analysis thus draws out strong links between the TCitH framework (which arguably belongs to the MPCitH family) and the VOLEitH framework.

Let us recall that a VOLE (for vector oblivious linear evaluation) is a multiparty gadget, where one party called prover, learns a pair $u \in \mathbb{F}_p^\ell, v \in \mathbb{F}_{p^k}^\ell$, while the verifier learns a random $\Delta \in \mathbb{F}_{p^k}$ and $q = u \cdot \Delta + v \in \mathbb{F}_{p^k}^\ell$. The key idea of the VOLEitH construction consists in proposing a method to commit a random vector u through a VOLE for which the verifier will be enabled to get a VOLE correlation $(\Delta, q = u \cdot \Delta + v)$. The used technique consists in transforming a vector commitment with all-but-one opening into a VOLEitH correlation. We can observe that a VOLE gadget can be seen as a $(2, N)$-threshold Shamir's secret sharing of u, for which the secret is stored at $P_u(\infty)$ instead of $P_u(0)$. Namely, to share u,

- one samples a random degree-1 polynomial P such that $P_u(\infty) = u$, *i.e.* one samples a random $v \in \mathbb{F}$ and defines $P_u(X) := uX + v$,
- the i^{th} party share is defined as
$$[\![u]\!]_i := P_u(e_i) = u \cdot e_i + v,$$
where $\{e_i\}_i$ are public evaluation points.

In this setting, each share corresponds to a VOLE correlation. Then, the technique of [BBdSG+23] to generate a VOLE correlation through a GGM tree is equivalent to the pseudo-random generation of a Shamir's secret sharing (with $\ell = 1$) derived from [CDI05] (and presented in Section 3.1). Let us stress that the techniques developed in our article do not specifically require storing the secret in $P(0)$, this is just the most common choice. In Equation (3), if we want to store the plain value at the infinity point, we just need to adapt the definition of P_T: P_T is here the unique degree-ℓ polynomial such that

$$\begin{cases} P_T(\infty) = 1 & \text{(instead of } P_T(0) = 1) \\ P_T(e_j) = 0 & \text{for all } j \in T. \end{cases}$$

In that case, if we restrict Equation (3) with $\ell = 1$, we obtain (up to the auxiliary value)

$$\begin{aligned} [\![w]\!]_i &= \sum_{j \neq i} s_{\{j\}} \cdot P_{\{j\}}(e_i) \\ &= \sum_{j \neq i} s_{\{j\}} \cdot (e_i - e_j) \end{aligned}$$

which exactly corresponds to the formulae from [BBdSG+23] (see Equation 2 in [BBdSG+23] for example). Once we made this observation (committing a VOLE gadget is equivalent to committing a $(2, N)$-threshold Shamir's secret sharing), we may wonder what would be the equivalent MPC protocol used in [BBdSG+23]. In fact, [BBdSG+23] relies on a variant of the QuickSilver VOLE-based protocol [YSWW21], and it can be equivalent to the MPC protocol of Section 4.3 adapted to the case where the plain values are stored at the infinity point in the sharing. To sum up, the VOLE-in-the-Head construction can be interpreted as an instantiation of the TCitH-GGM framework with $\ell = 1$ using the MPC protocol of Section 4.3.

However, the VOLE-in-the-Head article proposes an interesting optimization. In what follows, we explain it when interpreted using sharings. Let us assume that we have τ sharings $[\![u]\!]^{(1)}, \ldots, [\![u]\!]^{(\tau)}$ of the same value $u \in \mathbb{F}$. We denote $P_u^{(1)}(X) := u \cdot X + v^{(1)}, \ldots, P_u^{(\tau)}(X) := u \cdot X + v^{(\tau)}$ the corresponding Shamir's polynomials (while assuming the shared value is stored to the infinity point). Let us consider an isomorphism ϕ between $\mathbb{F}_q^\tau$ and $\mathbb{F}_{q^\tau}$. Then, the N^τ-sharing $[\![u]\!]^{(\phi)}$ defined as

$$\forall (i_1, \ldots, i_\tau) \in [1 : N]^\tau, \ [\![u]\!]^{(\phi)}_{1 + i_1 \cdot N + \ldots + i_\tau \cdot N^{\tau - 1}} = \phi([\![u]\!]^{(1)}_{i_1}, \ldots, [\![u]\!]^{(\tau)}_{i_\tau})$$

is a $(2, N^\tau)$-threshold Shamir's secret sharing of $u \in \mathbb{F}$, with polynomial

$$P_u^{(\phi)}(X) = u \cdot X + \phi(v^{(1)}, \ldots, v^{(\tau)}).$$

Indeed, we have

$$\forall (i_1, \ldots, i_\tau) \in [1:N]^\tau,\ [\![u]\!]^{(\phi)}_{1+i_1 \cdot N + \ldots + i_\tau \cdot N^{\tau-1}} = P_u^{(\phi)}(\phi(e_{i_1}, \ldots, e_{i_\tau})).$$

So, τ N-sharings of the same value can be seen as a single N^τ-sharing of this value (living in the field extension $\mathbb{F}_{q^\tau}$). Then, instead of executing τ times the MPC protocol on these τ sharings, we can merge them and execute the MPC protocol only once in the field extension. The main advantage is that the resulting soundness error will be around $\frac{d}{N^\tau}$ instead of $(\frac{d}{N})^\tau$, where d is the maximal degree of the broadcasted sharings. However, it implies executing the MPC protocol in $\mathbb{F}_{q^\tau}$ instead of executing it τ times in $\mathbb{F}_q$, which represents an extra computational cost. So this tweak actually provides new trade-offs between the communication cost and the signature size (shorter size, but larger running times). Let us further remark that to use this tweak the prover must additionally prove that these τ sharings encode the same value u, which can be easily handled by adding an additional verifier challenge (increasing the number of rounds) and short fixed-size communication cost.

The TCitH framework can be seen as a generalization of the VOLEitH construction which, as analyzed above, corresponds to the GGM variant with $\ell = 1$ (and a further tweak). Our work thus strengthens the bridge initiated by [BBdSG+23] between VOLE-based protocols and the MPCitH framework. The proposed TCitH framework also gathers several works under a common theory (the hypercube technique from [AGH+23], the threshold approach from [FR22], VOLE-in-the-Head from [BBdSG+23]) and initiates a bridge between additive-based MPCitH proof systems and Ligero (the latter was the only one of its kind during many years).

6 Conclusion

In this work, we have introduced a generic practical (MPCitH-based) framework, named *Threshold Computation in the head* (TCitH), which could be used as a theoretical foundation for many applications. We notably showed that this framework is instrumental to build post-quantum (ring) signatures with short sizes and/or fast verification. In particular, applying our techniques we were able to improve the timings and signature sizes of most of the MPCitH-based signature candidates submitted to the recent NIST call. We could also introduce the post-quantum ring signature scheme with the shortest signatures in the current literature.

We provided some comparative analysis of our TCitH framework and the recently proposed VOLE-in-the-Head (VOLEitH) framework. While the GGM case of our framework with $\ell = 1$ is similar to the VOLEitH construction, our framework brings further flexibility and trade-offs. In particular, the threshold parameter ℓ can be taken greater than 1 which provides further interesting trade-offs in some applications (see, e.g. the SDitH NIST candidate [AFG+23]). The Merkle tree variant of our framework enjoys a particularly efficient verification, which scales in $O(\lambda)$ against $O(\lambda N/\log(N))$ for other MPCitH / VOLEitH proof systems based on GGM trees. Finally, the TCitH framework (MT variant) can support batching (as shown in [FR22]) and could further be extended to sublinear arguments using techniques à la Ligero [AHIV17], which would deserve further investigations.

References

ABB+23a. Najwa Aaraj, Slim Bettaieb, Loïc Bidoux, Alessandro Budroni, Victor Dyseryn, Andre Esser, Philippe Gaborit, Mukul Kulkarni, Victor Mateu, Marco Palumbi, Lucas Perin, and Jean-Pierre Tillich. PERK, 2023. `https://pqc-perk.org/assets/downloads/PERK_specifications.pdf`.

ABB+23b. Gora Adj, Stefano Barbero, Emanuele Bellini, Andre Esser, Luis Rivera-Zamarripa, Carlo Sanna, Javier Verbel, and Floyd Zweydinger. MiRitH (MinRank in the Head). 29st May 2023, 2023. `https://pqc-mirith.org/assets/downloads/mirith_specifications_v1.0.0.pdf`.

ABB+23c. Nicolas Aragon, Magali Bardet, Loïc Bidoux, Jesús-Javier Chi-Domínguez, Victor Dyseryn, Thibauld Feneuil, Philippe Gaborit, Antoine Joux, Matthieu Rivain, Jean-Pierre Tillich, and Adrien Vinçotte. RYDE Specifications, 2023. https://pqc-ryde.org/assets/downloads/ryde_spec.pdf.

ABB+23d. Nicolas Aragon, Magali Bardet, Loïc Bidoux, Jesús-Javier Chi-Domínguez, Victor Dyseryn, Thibauld Feneuil, Philippe Gaborit, Romaric Neveu, Matthieu Rivain, and Jean-Pierre Tillich. MIRA Specifications, 2023. https://pqc-mira.org/assets/downloads/mira_spec.pdf.

ABC+23. Nicolas Aragon, Loïc Bidoux, Jesús-Javier Chi-Domínguez, Thibauld Feneuil, Philippe Gaborit, Romaric Neveu, and Matthieu Rivain. Mira: a digital signature scheme based on the minrank problem and the mpc-in-the-head paradigm, 2023.

AFG+23. Carlos Aguilar Melchor, Thibauld Feneuil, Nicolas Gama, Shay Gueron, James Howe, David Joseph, Antoine Joux, Edoardo Persichetti, Tovohery H. Randrianarisoa, Matthieu Rivain, and Dongze Yue. The Syndrome Decoding in the Head (SD-in-the-Head) Signature Scheme – Algorithm Specifications and Supporting Documentation. Version 1.0 – 31st May 2023, 2023. https://sdith.org/docs/sdith-v1.0.pdf.

AGH+23. Carlos Aguilar Melchor, Nicolas Gama, James Howe, Andreas Hülsing, David Joseph, and Dongze Yue. The return of the SDitH. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 564–596. Springer, Heidelberg, April 2023.

AHIV17. Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 2087–2104. ACM Press, October / November 2017.

ARZV23. Gora Adj, Luis Rivera-Zamarripa, and Javier Verbel. Minrank in the head. In Nadia El Mrabet, Luca De Feo, and Sylvain Duquesne, editors, *Progress in Cryptology - AFRICACRYPT 2023*, pages 3–27, Cham, 2023. Springer Nature Switzerland.

BBdSG+23. Carsten Baum, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß, Emmanuela Orsini, Lawrence Roy, and Peter Scholl. Publicly verifiable zero-knowledge and post-quantum signatures from vole-in-the-head. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023*, pages 581–615, Cham, 2023. Springer Nature Switzerland.

BBN+22. Alessandro Barenghi, Jean-François Biasse, Tran Ngo, Edoardo Persichetti, and Paolo Santini. Advanced signature functionalities from the code equivalence problem. *International Journal of Computer Mathematics: Computer Systems Theory*, 7(2):112–128, 2022.

BCCT12. Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In Shafi Goldwasser, editor, *ITCS 2012*, pages 326–349. ACM, January 2012.

BCF+23. Loïc Bidoux, Jesús-Javier Chi-Domínguez, Thibauld Feneuil, Philippe Gaborit, Antoine Joux, Matthieu Rivain, and Adrien Vinçotte. Ryde: A digital signature scheme based on rank-syndrome-decoding problem with mpcith paradigm, 2023.

BCI+20. Eli Ben-Sasson, Dan Carmon, Yuval Ishai, Swastik Kopparty, and Shubhangi Saraf. Proximity gaps for reed-solomon codes. In *61st FOCS*, pages 900–909. IEEE Computer Society Press, November 2020.

BD20. Ward Beullens and Cyprien Delpech de Saint Guilhem. LegRoast: Efficient post-quantum signatures from the Legendre PRF. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, pages 130–150. Springer, Heidelberg, 2020.

BDK+21. Carsten Baum, Cyprien Delpech de Saint Guilhem, Daniel Kales, Emmanuela Orsini, Peter Scholl, and Greg Zaverucha. Banquet: Short and fast signatures from AES. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 266–297. Springer, Heidelberg, May 2021.

BESV22. Emanuele Bellini, Andre Esser, Carlo Sanna, and Javier Verbel. Mr-dss – smaller minrank-based (ring-)signatures. In Jung Hee Cheon and Thomas Johansson, editors, *Post-Quantum Cryptography*, pages 144–169, Cham, 2022. Springer International Publishing.

BKP20. Ward Beullens, Shuichi Katsumata, and Federico Pintore. Calamari and Falafl: Logarithmic (linkable) ring signatures from isogenies and lattices. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 464–492. Springer, Heidelberg, December 2020.

BKPV23. Luk Bettale, Delaram Kahrobaei, Ludovic Perret, and Javier Verbel. Biscuit: Shorter MPC-based Signature from PoSSo, 2023. https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-1/spec-files/Biscuit-spec-web.pdf.

BN20. Carsten Baum and Ariel Nof. Concretely-efficient zero-knowledge arguments for arithmetic circuits and their application to lattice-based cryptography. In Aggelos Kiayias, Markulf Kohlweiss, Petros

Wallden, and Vassilis Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 495–526. Springer, Heidelberg, May 2020.

CCH+23. Jihoon Cho, Mingyu Cho, Jincheol Ha, Seongkwang Kim, Jihoon Kwon, Byeonghak Lee, Joohee Lee, Jooyoung Lee, Sangyub Lee, Dukjae Moon, Mincheol Son, and Hyojin Yoon. The AIMer Signature Scheme – Submission to the NIST PQC project. Version 1.0 – 1st June 2023, 2023. `https://aimer-signature.org/docs/AIMer-NIST-Document.pdf`.

CDI05. Ronald Cramer, Ivan Damgård, and Yuval Ishai. Share conversion, pseudorandom secret-sharing and applications to secure computation. In Joe Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 342–362. Springer, Heidelberg, February 2005.

DOT21. Cyprien Delpech de Saint Guilhem, Emmanuela Orsini, and Titouan Tanguy. Limbo: Efficient zero-knowledge MPCitH-based arguments. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 3022–3036. ACM Press, November 2021.

EZS+19. Muhammed F. Esgin, Raymond K. Zhao, Ron Steinfeld, Joseph K. Liu, and Dongxi Liu. MatRiCT: Efficient, scalable and post-quantum blockchain confidential transactions protocol. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 567–584. ACM Press, November 2019.

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, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 541–572. Springer, Heidelberg, August 2022.

FJR23. Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. Shared permutation for syndrome decoding: new zero-knowledge protocol and code-based signature. *Des. Codes Cryptogr.*, 91(2):563–608, 2023.

FMRV22. Thibauld Feneuil, Jules Maire, Matthieu Rivain, and Damien Vergnaud. Zero-knowledge protocols for the subset sum problem from MPC-in-the-head with rejection. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part II*, volume 13792 of *LNCS*, pages 371–402. Springer, Heidelberg, December 2022.

FR22. Thibauld Feneuil and Matthieu Rivain. Threshold linear secret sharing to the rescue of MPC-in-the-head. Cryptology ePrint Archive, Report 2022/1407, 2022. `https://eprint.iacr.org/2022/1407`.

FR23. Thibauld Feneuil and Matthieu Rivain. MQOM: MQ on my Mind – Algorithm Specifications and Supporting Documentation. Version 1.0 – 31st May 2023, 2023. `https://mqom.org/docs/mqom-v1.0.pdf`.

GGHAK22. Aarushi Goel, Matthew Green, Mathias Hall-Andersen, and Gabriel Kaptchuk. Efficient set membership proofs using mpc-in-the-head. *Proceedings on Privacy Enhancing Technologies*, 2022:304–324, 04 2022.

GGM84. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th FOCS*, pages 464–479. IEEE Computer Society Press, October 1984.

Gro16. Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016.

IKOS07. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *39th ACM STOC*, pages 21–30. ACM Press, June 2007.

ISN89. Mitsuru Ito, Akira Saito, and Takao Nishizeki. Secret sharing scheme realizing general access structure. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, 72(9):56–64, 1989.

KHS+22. Seongkwang Kim, Jincheol Ha, Mincheol Son, Byeonghak Lee, Dukjae Moon, Joohee Lee, Sangyup Lee, Jihoon Kwon, Jihoon Cho, Hyojin Yoon, and Jooyoung Lee. AIM: Symmetric primitive for shorter signatures with stronger security. Cryptology ePrint Archive, Report 2022/1387, 2022. `https://eprint.iacr.org/2022/1387`.

KKW18. Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 525–537. ACM Press, October 2018.

KZ20a. Daniel Kales and Greg Zaverucha. An attack on some signature schemes constructed from five-pass identification schemes. In Stephan Krenn, Haya Shulman, and Serge Vaudenay, editors, *CANS 20*, volume 12579 of *LNCS*, pages 3–22. Springer, Heidelberg, December 2020.

KZ20b. Daniel Kales and Greg Zaverucha. Improving the performance of the Picnic signature scheme. *IACR TCHES*, 2020(4):154–188, 2020. `https://tches.iacr.org/index.php/TCHES/article/view/8680`.

LAZ19. Xingye Lu, Man Ho Au, and Zhenfei Zhang. Raptor: A practical lattice-based (linkable) ring signature. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *ACNS 19*, volume 11464 of *LNCS*, pages 110–130. Springer, Heidelberg, June 2019.

NIS22. NIST. Call for Additional Digital Signature Schemes for the Post-Quantum Cryptography Standardization Process, 2022. https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022.pdf.

YSWW21. Kang Yang, Pratik Sarkar, Chenkai Weng, and Xiao Wang. QuickSilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 2986–3001. ACM Press, November 2021.

ZCD+17. Greg Zaverucha, Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, and Daniel Slamanig. Picnic. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions.

A Zero-Knowledge Proof based on Standard MPCitH with Additive Sharing

1. The prover shares the witness w into a sharing $[\![w]\!]$.
2. The prover emulates "in her head" the N parties of the MPC protocol.
   For $j = 1$ to t:
   (a) the prover computes
   $$\beta^j = \psi^j(w, (\varepsilon^i)_{i<j}),$$
   shares it into a sharing $[\![\beta^j]\!]$;
   (b) the prover computes the commitments
   $$\mathsf{com}_i^j := \begin{cases} \mathrm{Com}([\![w]\!]_i, [\![\beta^1]\!]_i; \rho_i^1) & \text{if } j = 1 \\ \mathrm{Com}([\![\beta^j]\!]_i; \rho_i^j) & \text{if } j > 1 \end{cases}$$
   for all $i \in \{1, \ldots, N\}$, for some commitment randomness ρ_i^j;
   (c) the prover sends
   $$h_j := \begin{cases} \mathrm{Hash}(\mathsf{com}_1^1, \ldots, \mathsf{com}_N^1) & \text{if } j = 1 \\ \mathrm{Hash}(\mathsf{com}_1^j, \ldots, \mathsf{com}_N^j, [\![\alpha^{j-1}]\!]) & \text{if } j > 1 \end{cases}$$
   to the verifier;
   (d) the verifier picks at random a challenge ε^j and sends it to the prover;
   (e) the prover computes
   $$[\![\alpha^j]\!] := \varphi^j_{(\varepsilon^i)_{i\le j}, (\alpha^i)_{i<j}}\big([\![w]\!], ([\![\beta^i]\!])_{i\le j}\big)$$
   and recomposes α^j.

   The prover further computes $h_{t+1} := \mathrm{Hash}([\![\alpha^t]\!])$ and sends it to the verifier.
3. The verifier picks at random a party index $i^* \in [N]$ and sends it to the prover.
4. The prover opens the commitments of all the parties except party i^* and further reveals the commitments and broadcast messages of the unopened party i^*. Namely, the prover sends $([\![w]\!]_i, ([\![\beta^j]\!]_i, \rho_i^j)_{j\in[t]})_{i\neq i^*}$, $\mathsf{com}_{i^*}^1, \ldots, \mathsf{com}_{i^*}^t$, $[\![\alpha^1]\!]_{i^*}, \ldots, [\![\alpha^t]\!]_{i^*}$ to the verifier.
5. The verifier recomputes the commitments com_i^j and the broadcast values $[\![\alpha^j]\!]_i$ for $i \in [N] \setminus \{i^*\}$ and $j \in [t]$ from $([\![w]\!]_i, ([\![\beta^j]\!]_i, \rho_i^j)_{j\in[t]})_{i\neq i^*}$ in the same way as the prover.
6. The verifier accepts if and only if:
   (a) the views of the opened parties are consistent with each other, with the committed input shares and with the hash digest of the broadcast messages, *i.e.* for $j = 1$ to $t + 1$,
   $$h_j \stackrel{?}{=} \begin{cases} \mathrm{Hash}(\mathsf{com}_1^1, \ldots, \mathsf{com}_N^1) & \text{if } j = 1 \\ \mathrm{Hash}(\mathsf{com}_1^j, \ldots, \mathsf{com}_N^j, [\![\alpha^{j-1}]\!]) & \text{if } j > 1 \\ \mathrm{Hash}([\![\alpha^t]\!]) & \text{if } j = t + 1 \end{cases}$$
   (b) the output of the MPC protocol is Accept, *i.e.*
   $$g(\alpha^1, \ldots, \alpha^t) \stackrel{?}{=} 0.$$

Protocol 5: Zero-knowledge protocol - Application of the MPCitH principle to the general MPC protocol of [FR22].

B Proof of Theorem 1

We first recall the theorem statement, and then give the proof.

Theorem 1. *Let Π be an MPC protocol of parameters $(N, \ell, d^{\alpha}_{max}, d^{\beta}_{max}, p)$ complying to the format of Protocol 2. In particular, Π is ℓ-private in the semi-honest model and of false positive probability p. Then, Protocol 4 built from Π satisfies the three following properties:*

- ***Completeness.*** *A prover $\mathcal{P}$ who knows a witness w such that $(x, w) \in \mathcal{R}$ and who follows the steps of the protocol always succeeds in convincing the verifier $\mathcal{V}$.*
- ***Soundness.*** *Suppose that there is an efficient prover $\tilde{\mathcal{P}}$ that, on input x, convinces the honest verifier $\mathcal{V}$ to accept with probability*

$$\tilde{\epsilon} := \Pr[\langle \tilde{\mathcal{P}}, \mathcal{V} \rangle(x) \to 1] \ > \ \epsilon$$

where the soundness error ϵ is defined as

$$\epsilon := p + (1-p) \cdot \frac{\binom{d^{\alpha}_{max}}{\ell}}{\binom{N}{\ell}} \quad \textit{for the TCitH-GGM variant,}$$

and

$$\epsilon := p + (1-p) \cdot \frac{\binom{d^{\alpha}_{max}}{\ell}}{\binom{N}{\ell}} + \frac{\binom{N}{d^{\beta}_{max}+1}^2}{|\mathbb{F}|^{\eta}} \quad \textit{for the TCitH-MT variant.}$$

Then, there exists a probabilistic extraction algorithm $\mathcal{E}$ with time complexity in $\mathsf{poly}(\lambda, (\tilde{\epsilon} - \epsilon)^{-1})$ that, given rewindable black-box access to $\tilde{\mathcal{P}}$, outputs either a witness w satisfying $(x, w) \in \mathcal{R}$, a hash collision, or a commitment collision.

- ***Honest-Verifier Zero-Knowledge.*** *Let the pseudorandom generator* PRG *used in Protocol 1 be $(t, \epsilon_{\text{PRG}})$-secure and the commitment scheme* Com *be $(t, \epsilon_{\text{Com}})$-hiding. There exists an efficient simulator $\mathcal{S}$ which, given random challenge I outputs a transcript which is $(t, \epsilon_{\text{PRG}} + \epsilon_{\text{Com}})$-indistinguishable from a real transcript of Protocol 1.*

Proof. The completeness and the zero-knowledge properties directly hold from the correctness and the privacy of the MPC protocol Π. We sketch the soundness proof hereafter for both variants of the TCitH framework. Along the proof, we shall denote Succ the event that the malicious prover $\tilde{\mathcal{P}}$ succeeds in convincing the verifier $\mathcal{V}$, that is:

$$\mathsf{Succ} \ \equiv \ \text{“}\langle \tilde{\mathcal{P}}, \mathcal{V} \rangle(x) \to 1\text{”}.$$

In the following, we assume that for any set of successful transcripts received by the extractor, no commitment collisions or hash collisions occur. Otherwise, the extractor trivially outputs such a collision. We will thus assume that for any hash or commitment value output produced by $\tilde{\mathcal{P}}$, a single preimage of this commitment/hash can be extracted from $\tilde{\mathcal{P}}$. In particular, the commitment h_1 corresponds to a single extractable value of the sharing $[\![w]\!]$.

Proof of soundness for the TCitH-GGM framework. Since $\tilde{\mathcal{P}}$ is malicious, their first message h_1 commits a sharing $[\![w]\!]$ of an invalid witness w, *i.e.* which does not satisfy $(x, w) \in \mathcal{R}$. While going through the protocol, two cases can occur:

- A false positive event occurs: the MPC randomness $\varepsilon^1, \ldots, \varepsilon^t$ sampled by the verifier is such that an honest execution of the MPC protocol produces the output Accept. By definition of the considered MPC protocol Π, this case occurs with probability p.
- No false positive event occurs. In that case, we show below that the last challenge-response round of the zero-knowledge protocol can be seen as a $\left(\binom{d^{\alpha}_{max}}{\ell} + 1\right)$-special-sound protocol. The probability that the verifier is convinced after the last round is thus $\binom{d^{\alpha}_{max}}{\ell} / \binom{N}{\ell}$, since $\binom{N}{\ell}$ is the size of the challenge space.

Denoting FP the false positive event, we get that the soundness error satisfies:

$$\epsilon = \Pr[\mathsf{FP}] + \Pr[\neg\mathsf{FP}] \cdot \Pr[\mathsf{Succ} \mid \neg\mathsf{FP}] = p + (1-p) \cdot \frac{\binom{d^{\alpha}_{max}}{\ell}}{\binom{N}{\ell}} \ .$$

Let us assume that the malicious prover ran all the rounds of the protocol except the last one such that no false positive occurred. We will show that the remaining challenge (the party opening challenge) and associated response form a $\left(\binom{d^{\alpha}_{max}}{\ell}+1\right)$-special-sound protocol. Namely, we will show that if $\binom{d^{\alpha}_{max}}{\ell}+1$ different accepting transcripts can be obtained from the malicious prover which only differ in their last round of challenge-response (opening of the parties), then a valid witness w can be extracted from these transcripts, leading to a contradiction. This shall imply that at most $\binom{d^{\alpha}_{max}}{\ell}$ valid transcripts can be obtained from the malicious prover if no false positive event occurs, implying $\Pr[\mathsf{Succ} \mid \neg\mathsf{FP}] \leq \binom{d^{\alpha}_{max}}{\ell} / \binom{N}{\ell}$.

Let us consider $\binom{d^{\alpha}_{max}}{\ell}+1$ accepting transcripts $\{\mathcal{T}_k\}_{k\in[1:\binom{d^{\alpha}_{max}}{\ell}+1]}$ where

$$\mathcal{T}_k := (\underbrace{h_1, \varepsilon^1, \ldots, h_t, \varepsilon^t, h_{t+1}}_{\text{Same for all transcripts}}, I_k, \underbrace{\sigma_k := (\mathsf{seedpath}_k, \{\mathsf{com}^j_{I_k}\}_j, \{\alpha^j\}_j, (\Delta x, \{\Delta\beta^j\}_j))}_{\text{Prover's last response, } (\Delta x, \{\Delta\beta^j\}_j) \text{ is omitted if } I_k = T_0})$$

with $I_k \neq I_{k'}$ for all $k \neq k'$. By assumption, the seed paths, the plain broadcast values $\{\alpha^j\}_j$, and the auxiliary values in $\{\sigma_k\}_k$ are consistent since otherwise we would get a commitment collision or a hash collision. Since we have at least two transcripts, we have at least two different party challenges, which means that *all the leaves* of the seed tree are *known*. Moreover, we have that at least one of the transcripts contains the auxiliary values. We can then build the Shamir's polynomials $P_w, P_{\beta^1}, \ldots, P_{\beta^t}$ using the Equation (4). Since $\{\mathcal{T}_k\}_k$ are valid transcripts, it means that, for all $i \in \bigcup_k I_k$ and $j \in [1:t]$, we have

$$P_{\alpha^j}(e_i) = \Phi^j\big(P_w(e_i), (P_{\beta^k}(e_i))_{k\leq j}\big)$$

since $[\![\alpha^j]\!]_i = P_{\alpha^j}(e_i)$, $[\![w]\!]_i = P_w(e_i)$ and $[\![\beta^k]\!]_i = P_{\beta^k}(e_i)$. Since the polynomials in the above relation are of degree at most d^{α}_{max} and since the relations hold for at least $d^{\alpha}_{max}+1$ evaluation points $\{e_i\}_{\bigcup_k I_k}$, we have that the relations hold directly on the polynomials: $P_{\alpha^j} = \Phi^j\big(P_w, (P_{\beta^k})_{k\leq j}\big)$ for all j. In particular, the relations are true when evaluating the polynomials in zero and we get

$$\alpha^j = \Phi^j\big(w, (\beta^k)_{k\leq j}\big)$$

since $P_{\alpha^j}(0) = \alpha^j$, $P_w(0) = w$ and $P_{\beta^k}(0) = \beta^k$. Moreover, since $\{\mathcal{T}_k\}_k$ are valid, we have

$$g(\alpha^1, \ldots, \alpha^t) = 0.$$

It means that we have an honest execution of the MPC protocol which outputs Accept on the witness w. Since we assumed that there were no false positive events, we get that w must be a valid witness, which concludes the proof.

Proof of soundness for the TCitH-MT framework. In the TCitH-MT framework, the first message h_1 sent by the prover is a Merkle tree commitment of the sharings $[\![w]\!]$, $[\![u]\!]$, $[\![\beta^1]\!]$, $[\![\bar{\beta}^2]\!]$, …, $[\![\bar{\beta}^t]\!]$ where each leaf of the tree is a commitment of the ith shares of all the sharings, for $i \in [1:N]$. Since only ℓ leaves are eventually open, the malicious prover may commit some sharings that do not form valid Shamir's secret sharings of the expected degrees.

Let $J \subseteq [1:N]$ such that $|J| = d_\beta + 1$, where $d_\beta = \max(\ell, d_1, \ldots, d_t)$. We shall denote $Q^{(J)}$, resp. $P_u^{(J)}$, the polynomial obtained by interpolation of $\{Q(e_i)\}_{i\in J}$, resp. $\{P_u(e_i)\}_{i\in J}$, for Q defined in Equation (7) (where both $Q(e_i)$ and $P_u(e_i)$ are defined with respect to the shares $[\![w]\!]_i, [\![u]\!]_i, [\![\beta^1]\!]_i, [\![\bar{\beta}^2]\!]_i, \ldots, [\![\bar{\beta}^t]\!]_i$). We stress that if the committed sharings are valid Shamir's secret sharings of their respective degrees (meaning that the polynomials P_w, P_u, P_{β_1}, …, P_{β_t} are of degrees ℓ, d^{β}_{max}, d_1, …, d_t respectively), all the polynomials $Q^{(J)}$

are equal to Q. But by committing invalid sharings (sharings of higher degrees), one ends up with different polynomials $Q^{(J)}$. We shall further denote $P_\xi^{(J)} = \Gamma \cdot Q^{(J)} + P_u^{(J)}$.

In the following, we shall assume that there exists a set $J \subseteq [1:N]$ (with $|J| = d_\beta + 1$) such that $P_\xi^{(J)} = P_\xi$, where P_ξ is the committed polynomial. Indeed, if no such set exists we have that at most d_{max}^β parties are such that $P_\xi(e_i) = \Gamma \cdot Q(e_i) + P_u(e_i)$ in the final checks of the verifier. The malicious prover can then only convince the verifier if the open parties are among these d_{max}^β parties. This means that we get $\Pr[\mathsf{Succ}] \leq \binom{d_{max}^\beta}{\ell} / \binom{N}{\ell} \leq \binom{d_{max}^\alpha}{\ell} / \binom{N}{\ell}$ so the final result directly holds. For this reason, we always consider that at least one set $J \subseteq [1:N]$ (with $|J| = d_\beta + 1$) is such that $P_\xi^{(J)} = P_\xi$.

The idea behind the degree-enforcing commitment scheme is the following: even though the malicious prover might commit invalid sharings giving rise to several pairs of polynomials $(Q^{(J)}, P_u^{(J)})$, the commitment of P_ξ should constrain the malicious prover to choose a single of these pairs, namely to choose a single set of sharings $[\![w]\!]$, $[\![u]\!]$, $[\![\beta^1]\!]$, $[\![\bar{\beta}^2]\!]$, $\ldots$, $[\![\bar{\beta}^t]\!]$ of the right degrees. More formally, for any two sets $J, J' \subseteq [1:N]$ with $|J| = |J'| = d_\beta + 1$, such that $(Q^{(J)}, P_u^{(J)}) \neq (Q^{(J')}, P_u^{(J')})$, we have:

$$\Pr\left[P_\xi^{(J)} = P_\xi^{(J')} \mid (Q^{(J)}, P_u^{(J)}) \neq (Q^{(J')}, P_u^{(J')})\right] \leq \frac{1}{|\mathbb{F}|^\eta} \ ,$$

where the above probability is over the randomness of Γ. We can deduce that for any initial commitment, and, more generally, for any set of polynomials $\{(Q^{(J)}, P_u^{(J)})\}_{|J|=d_\beta+1}$, we have

$$\Pr[\mathsf{Coll}] \leq \frac{\binom{N}{d_\beta+1}^2}{2|\mathbb{F}|^\eta} \ ,$$

for Coll the event defined as

$$\mathsf{Coll} \ \equiv \ \text{“}\exists\, J, J' \subseteq [1:N] \text{ with } |J| = |J'| = d_\beta + 1 \text{ s.t. } (Q^{(J)}, P_u^{(J)}) \neq (Q^{(J')}, P_u^{(J')}) \text{ and } P_\xi^{(J)} = P_\xi^{(J')}\text{”}$$

The soundness error of the protocol then satisfies:

$$\epsilon \ = \ \Pr[\mathsf{Coll}] + \Pr[\neg\mathsf{Coll}] \cdot \Pr[\mathsf{Succ} \mid \neg\mathsf{Coll}] \ \leq \ \Pr[\mathsf{Succ} \mid \neg\mathsf{Coll}] + \frac{\binom{N}{d_\beta+1}^2}{2|\mathbb{F}|^\eta} \ .$$

Let us now focus on $\Pr[\mathsf{Succ} \mid \neg\mathsf{Coll}]$, namely we consider a valid transcript for which $\neg\mathsf{Coll}$ occurs. As argued above, there exists at least one set $J \subseteq [1:N]$ with $|J| = d_\beta + 1$ such that $P_\xi^{(J)} = P_\xi$. Let us denote

$$\mathcal{H} = \bigcup_{P_\xi^{(J)} = P_\xi} J \ .$$

For any $i \notin \mathcal{H}$, we have $P_\xi(e_i) \neq \Gamma \cdot Q(e_i) + P_u(e_i)$ (where $Q(e_i)$ and $P_u(e_i)$ are recomputed from the shares $[\![w]\!]_i, [\![u]\!]_i, [\![\beta^1]\!]_i, [\![\bar{\beta}^2]\!]_i, \ldots, [\![\bar{\beta}^t]\!]_i$), so the verification always fails if such a party is open. Moreover, since $\neg\mathsf{Coll}$ occurs, $\{Q^{(J)}\}_{J \subseteq \mathcal{H}}$ is a singleton, meaning that the shares $\{[\![w]\!]_i, [\![u]\!]_i, [\![\beta^1]\!]_i, [\![\bar{\beta}^2]\!]_i, \ldots, [\![\bar{\beta}^t]\!]_i\}_{i \in \mathcal{H}}$ are valid Shamir's secret shares of unique values w, u, β^1, $\bar{\beta}^2$, $\ldots$, $\bar{\beta}^t$. The rest of the proof is similar as the TCitH-GGM case. We also consider two cases:

- A false positive event occurs: the MPC randomness $\varepsilon^1, \ldots, \varepsilon^t$ sampled by the verifier is such that an honest execution of the MPC protocol produces the output Accept. By definition of the considered MPC protocol Π, this case occurs with probability p.
- No false positive event occurs. In that case, we show below that the last challenge-response round of the zero-knowledge protocol can be seen as a $\left(\binom{d_{max}^\alpha}{\ell} + 1\right)$-special-sound protocol. The probability that the verifier is convinced after the last round is thus $\binom{d_{max}^\alpha}{\ell} / \binom{N}{\ell}$, since $\binom{N}{\ell}$ is the size of the challenge space.

We get that the soundness error satisfies:

$$\begin{aligned}\epsilon \;&\leq\; \Pr[\mathsf{FP}] + \Pr[\neg\mathsf{FP}] \cdot \Pr[\mathsf{Succ} \mid (\neg\mathsf{FP}) \wedge (\neg\mathsf{Coll})] + \Pr[\mathsf{Coll}] \\ &\leq\; p + (1-p) \cdot \frac{\binom{d^{\alpha}_{max}}{\ell}}{\binom{N}{\ell}} + \frac{\binom{N}{d_\beta+1}^2}{2|\mathbb{F}|^\eta} \;. \end{aligned} \tag{12}$$

Let us assume that the malicious prover ran all the rounds of the protocol except the last one such that no false positive occurred. We will show that the remaining challenge (the party opening challenge) and associated response form a $\left(\binom{d^{\alpha}_{max}}{\ell}+1\right)$-special-sound protocol. Namely, we will show that if $\binom{d^{\alpha}_{max}}{\ell}+1$ different accepting transcripts can be obtained from the malicious prover which only differ in their last round of challenge-response (opening of the parties), then a valid witness w can be extracted from these transcripts, leading to a contradiction. This shall imply that at most $\binom{d^{\alpha}_{max}}{\ell}$ valid transcripts can be obtained from the malicious prover if no false positive event occurs, implying $\Pr[\mathsf{Succ} \mid \neg\mathsf{FP}] \leq \binom{d^{\alpha}_{max}}{\ell} / \binom{N}{\ell}$.

Let us consider $\binom{d^{\alpha}_{max}}{\ell}+1$ accepting transcripts $\{\mathcal{T}_k\}_{k\in[1:\binom{d^{\alpha}_{max}}{\ell}+1]}$ where

$$\mathcal{T}_k := \big(\underbrace{h_1, \Gamma, h'_1, \varepsilon^1, \ldots, h_t, \varepsilon^t, h_{t+1}}_{\text{Same for all transcripts}}, I_k, \underbrace{\sigma_k := \big(\{[\![w]\!]_i, [\![\beta^1]\!]_i, [\![\bar{\beta}^2]\!]_i, \ldots, [\![\bar{\beta}^t]\!]_i\}_{i\in I_k}, \{\Delta\beta^j\}_j, \mathsf{path}_{I_k}, [\![\alpha]\!]_{S_k}\big)}_{\text{Prover's last response}}\big)$$

with $I_k \neq I_{k'}$ for all $k \neq k'$ (and where S_k denote some set of cardinality $|S_k| = d_\beta + 1 - \ell$ disjoint of I_k).

From these transcript, since $I_k \neq I_{k'}$ for all $k \neq k'$, we can retrieve at least $d^{\alpha}_{max} + 1 \geq d^{\beta}_{max} + 1$ shares of the committed sharings $[\![w]\!]$, $[\![u]\!]$, $[\![\beta^1]\!]$, $[\![\bar{\beta}^2]\!]$, …, $[\![\bar{\beta}^t]\!]$ (which pass the degree-enforcement test, *i.e.* which are consistent with P_ξ). From these shares, we can build the Shamir's polynomials $P_w, P_u, P_{\beta^1}, \ldots, P_{\beta^t}$ by interpolation. The rest of the proof is similar to the GGM case. Since $\{\mathcal{T}_k\}_k$ are valid transcripts, it means that, for all $i \in \bigcup_k I_k$ and $j \in [1:t]$, we have

$$P_{\alpha^j}(e_i) = \Phi^j\big(P_w(e_i), (P_{\beta^k}(e_i))_{k\leq j}\big)$$

since $[\![\alpha^j]\!]_i = P_{\alpha^j}(e_i)$, $[\![w]\!]_i = P_w(e_i)$ and $[\![\beta^k]\!]_i = P_{\beta^k}(e_i)$. Since the polynomials in the above relation are of degree at most d^{α}_{max} and since the relations hold for at least $d^{\alpha}_{max} + 1$ evaluation points $\{e_i\}_{\bigcup_k I_k}$, we have that the relations hold directly on the polynomials: $P_{\alpha^j} = \Phi^j\big(P_w, (P_{\beta^k})_{k\leq j}\big)$ for all j. In particular, the relations are true when evaluating the polynomials in zero and we get

$$\alpha^j = \Phi^j\big(w, (\beta^k)_{k\leq j}\big)$$

since $P_{\alpha^j}(0) = \alpha^j$, $P_w(0) = w$ and $P_{\beta^k}(0) = \beta^k$. Moreover, since $\{\mathcal{T}_k\}_k$ are valid, we have

$$g(\alpha^1, \ldots, \alpha^t) = 0.$$

It means that we have an honest execution of the MPC protocol which outputs Accept on the witness w. Since we assumed that there were no false positive events ($\neg\mathsf{FP}$), we get that w must be a valid witness, which concludes the proof. □