Thibauld Feneuil\textsuperscript{1,2} and Matthieu Rivain\textsuperscript{1}

\textsuperscript{1} CryptoExperts, Paris, France
\textsuperscript{2} Sorbonne Universit\'e, CNRS, INRIA, Institut de Math\'ematiques
de Jussieu-Paris Rive Gauche, Ouragan, Paris, France
\{thibauld.feneuil,matthieu.rivain\}@cryptoexperts.com

\section*{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. In this work, we extend and improve the recently proposed framework of MPC-in-the-Head based on threshold secret sharing, here called \textit{Threshold Computation in the Head}. We first address some limitations of this framework, namely its overhead in the communication cost, its constraint on the number of parties and its degradation of the soundness. 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 as well as packed secret sharing. We show the benefits of our extended framework for several applications. First we provide post-quantum zero-knowledge arguments for arithmetic circuits which improve the state of the art in the "small to medium size" regime. Then we apply our extended framework to derive improved variants of the MPCitH candidates submitted to NIST. For most of them, we save between 5\% and 37\% of the signature size. We further propose a generic way to build efficient post-quantum ring signatures from any one-way function. When applying our TCitH framework to this design to concrete one-way functions, the obtained scheme outperforms all the previous proposals in the state of the art. For instance, our scheme instantiated with MQ achieves sizes below 6 KB and timings around 10 ms for a ring of 4000 users. Finally, we provide exact arguments for lattice problems. Our scheme is competitive with state-of-the-art zero-knowledge lattice techniques and achieves proofs around 15 KB for LWE instances with similar security as Kyber512. We conclude our work by exhibiting strong connections between the TCitH framework and other proof systems (namely VOLE-in-the-Head and Ligero) which thus unifies different MPCitH-like proof systems under the same umbrella.

\textbf{Keywords:} Zero-knowledge proofs · MPC-in-the-Head · Threshold secret sharing · Post-quantum signatures · Ring signatures

\section{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 that 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 [FR23], 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 \(\ell = 1\)). Moreover, it enjoys a very efficient verification, which is logarithmic in \(N\) (against linear in \(N\) for the traditional and hypercube settings).

However, the original TCitH framework suffers a few downsides. First, using Merkle trees in place of GGM trees for the commitments of shares implies a communication overhead (this overhead typically represents 2 KB for non-interactive arguments with 128-bit security). Then, 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 can be an issue when dealing with small fields. Finally, the soundness of TCitH is degraded by the ability of the prover to commit sharings that are not valid \((\ell + 1, N)\)-threshold sharings.

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

1. **TCitH with GGM trees.** We first address the two aforementioned limitations of TCitH compared to standard MPCitH (with seed trees and hypercube technique). We show how using techniques from [CDI05], we can generate and commit Shamir’s secret sharings 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-GGM variant, we further propose a way to mitigate the limitation on the number of parties by lifting the MPC protocol into a field extension. This lifting does not impact the proof size but slightly increases the number of party emulations to \(1 + \lceil \log N / \log |\mathbb{F}| \rceil\) (for a soundness error of \(\approx 1/N\)). In terms of computation, the lifted version of TCitH-GGM is equivalent to the hypercube technique [AGH+23] when the base field is as small as \(\mathbb{F}_2\) while it is strictly more efficient for larger fields. We show that our (lifted) TCitH-GGM framework can improve the computational performance of nearly all the recent NIST post-quantum signature candidates based on the MPCitH paradigm with savings ranging from 9% to 51% compared to their hypercube optimized version. 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. It hence essentially provides a new interesting trade-off.

2. **Degree-enforcing commitment scheme.** In the original TCitH framework, the Merkle tree commitment allows a malicious prover to commit inconsistent Shamir’s secret sharings, namely sharings which are not of the right degree. This results in a degradation of the soundness which is further amplified when using packing and/or high-degree MPC functions. To tackle this issue, we introduce a degree-enforcing commitment scheme for the Merkle tree variant of the TCitH framework. Such a scheme ensures that the committed sharings are of the right degree with a tunable soundness error which can be made negligible. Our approach is a tweak of the proximity test for Reed-Solomon codes proposed in Ligero [AHIV17] which is crucially integrated to the commitment (and not to the MPC protocol) to achieve stronger guarantees (i.e., exact degree of committed sharings) which allows us to reach a better soundness.

3. **Extended TCitH framework.** We extend the original TCitH framework in two ways. First, we consider MPC protocols locally computing quadratic (or higher degree) functions instead of being restricted to linear functions. This extension allows us to cover richer MPC protocols resulting in smaller proofs (or signatures). Second, we consider packed secret sharing which consists in packing several witness coordinates in a single sharing. This extension allows us to compress the size of the witness in the Merkle tree variant of the TCitH framework and to achieve better proof sizes for “small to medium” size statements. We thus obtain a versatile framework supporting packing and high-degree MPC and which comes with two variants depending on the used method to generate and commit the shares: GGM tree (smaller proofs for “small size” statements) vs. Merkle tree (faster verification, smaller proofs for “medium size” statements). We provide a tight soundness analysis for these two variants with respect to the packing and and degree parameters of the MPC protocol.
4. Applications. We demonstrate the efficiency and versatility of the improved TCitH framework with various applications:

(i) Short zero-knowledge arguments for arithmetic circuits. We provide generic proof systems resulting in short proofs for arithmetic circuits. For “small to medium” size circuits (with $\leq 2^{16}$ multiplication gates), our arguments are (up to twice) smaller compared to Ligero [AHIV17, AHIV23], the state of the art in terms of post-quantum zero-knowledge arguments in this regime.

(ii) Improved post-quantum signatures. We then apply our extended framework to propose improved variants of the recent NIST post-quantum signature candidates based on the MPCitH paradigm. We save between 5% and 37% of the signature size for all of these schemes. In particular, our framework applied to the non-structured multivariate quadratic (MQ) problem provides signature sizes of 4.2 KB which is to be compared 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) [FR23a, BKPV23]. For a small field instance of MQ (specifically on $\mathbb{F}_4$), we obtain signature sizes of 3.8 KB while still using 256 parties.

(iii) Short post-quantum ring signatures. 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 4000 users, these schemes have running times below 10 ms and 20 ms, while achieving sizes around 5 KB (for MQ) and 9 KB (for SD), which greatly improves the current state of the art. We further apply our scheme to an AES-based one-way function, which gives us a ring signature smaller that 8 KB that only relies on symmetric cryptography assumptions (for up to $2^{20}$ users).

(iv) Exact proofs for lattice problems. We finally show how our framework provides short and exact zero-knowledge arguments for lattice problems. On toy examples from the literature, our proofs are between 3 and 5 times smaller than the best previous MPCitH-based proofs. On the Kyber512 (resp. Dilithium2) LWE instance, we achieve 21 KB (resp. 40 KB), which is competitive with the recent state of the art [LNP22] achieving 19 KB for a lattice-based statement related to Kyber512 (but proving the $L_2$ norm whereas we prove the $L_\infty$ norm). For custom instances with same security as Kyber512, our proofs can be lowered to 15 KB.

5. Unifying MPCitH-like proof systems. We conclude our work by exhibiting strong connections between some variants and/or instantiations of the TCitH framework and other proof systems, namely VOLE-in-the-Head [BBdSG+23] and Ligero [AHIV17, AHIV23]. While the TCitH framework originates from an extension of standard MPC-in-the-Head proof systems with (packed) Shamir’s secret sharing, we show that (a specific instantiation of) the GGM variant can be interpreted as a VOLE-in-the-Head proof system. On the other hand, the Merkle tree variant of the framework with packed secret sharing can be seen as a generalization of Ligero which we improve thanks to the degree-enforcing commitment scheme. Our framework thus unifies different MPCitH-like proof systems under the same umbrella, which provides further insights on these techniques and might foster future improvements.

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 [FR23b] for their formal definitions with similar notations.

Notations for (vector) polynomials. Let $\mathbb{F}$ a finite field and $P \in \mathbb{F}[X]$ a polynomial with coefficients in $\mathbb{F}$. We shall denote $\text{coeff}_j(P)$ the function mapping a polynomial $P$ to its degree-$j$ coefficient, so that $P(X) = \sum_{j=0}^{\deg P} \text{coeff}_j(P)X^j$. We call $P = (P_1, \ldots, P_n) \in (\mathbb{F}[X])^n$ a vector polynomial. The $\text{coeff}_j$ function extends to vector polynomials: for any $P \in (\mathbb{F}[X])^n$, $\text{coeff}_j(P) \in \mathbb{F}^n$ is naturally defined as the tuple of the degree-$j$ coefficients $(\text{coeff}_j(P_1), \ldots, \text{coeff}_j(P_n))$. 


2.1 Secret Sharing

Along the paper, the sharing of a value \( v \) is denoted \( \llbracket v \rrbracket := (\llbracket v \rrbracket_1, \ldots, \llbracket v \rrbracket_N) \) with \( \llbracket v \rrbracket_i \) denoting the share of index \( i \) for every \( i \in [1 : N] \). For any subset of indices \( J \subseteq [N] \), we shall further denote \( \llbracket v \rrbracket_J := (\llbracket v \rrbracket_i)_{i \in J} \).

A \((t, N)\)-threshold secret sharing scheme is a method to share a value \( v \in \mathbb{F} \) into a sharing \( \llbracket v \rrbracket \) such that \( v \) 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 \( \llbracket v_1 \rrbracket \), \( \llbracket v_2 \rrbracket \) and any two values \( a_1, a_2 \in \mathbb{F} \), computing \( a_1 \cdot \llbracket v_1 \rrbracket + a_2 \cdot \llbracket v_2 \rrbracket \) yields a sharing of \( a_1 v_1 + a_2 v_2 \). The additive secret sharing is an \((N, N)\)-threshold LSSS for which \( v = \sum_{i=1}^N \llbracket v_i \rrbracket \).

The techniques proposed in the paper rely on Shamir’s secret sharing \(^\text{Sha79} \) and on its packed version \(^\text{FY92} \), which we formally define below.

**Definition 1 (Packed Shamir’s Secret Sharing).** Let \( \mathbb{F} \) be a finite field and let \( s, d, N \in \mathbb{N} \) such that \( s \leq d < N \) and \( s + N \leq |\mathbb{F}| + 1 \), where \( s \) is called the pack size, \( d \) the degree and \( N \) the sharing size. Let \( \{ e_1, \ldots, e_N \} \) and \( \{ \omega_1, \ldots, \omega_{d+1} \} \) be two public sets of distinct elements of \( \mathbb{F} \) such that \( \{ \omega_1, \ldots, \omega_s \} \) is disjoint from all \( e_i \)’s. A \((s, d, N)\)-packed Shamir’s secret sharing of a tuple \( v \in \mathbb{F}^s \) is a tuple

\[
\llbracket v \rrbracket = (\llbracket v \rrbracket_1, \ldots, \llbracket v \rrbracket_N) \in \mathbb{F}^N \quad \text{s.t.} \quad \llbracket v \rrbracket_i := P_v(e_i) \quad \forall i \in [1 : N],
\]

for some polynomial \( P_v \) of degree \( d \) satisfying \( (P_v(\omega_1), \ldots, P_v(\omega_s)) = v \).

**Fresh sharing.** A fresh sharing of \( v \) with parameter \((s, d, N)\) is generated as follows:

- sample \( r_1, \ldots, r_\ell \) uniformly in \( \mathbb{F} \) where \( \ell := d + 1 - s \),
- build the polynomial \( P_v \) by interpolation as the polynomial of degree \( \leq d \) satisfying \( P_v(\omega_i) = v_i \forall i \in [1 : s] \) and \( P_v(\omega_{i+s}) = r_i \forall i \in [1 : \ell] \),
- build the shares \( \llbracket v \rrbracket_i \), as evaluations \( P_v(e_i) \) of \( P_v \) for each \( i \in [1 : N] \).

**Privacy and reconstructability.** The parameter \( \ell := d + 1 - s \) is the privacy threshold of the sharing: any set of \( \ell \) shares of a fresh sharing \( \llbracket v \rrbracket \) do not reveal any information on \( v \). On the other hand, \( d + 1 \) shares are necessary to fully recover \( v \). For any subset \( J \subseteq [N] \), s.t. \( |J| = d + 1 \), the recovery of \( v \) from \( \llbracket v \rrbracket_J \) works by interpolating the polynomial \( P_v \) from the \( d + 1 \) evaluation points \( \llbracket v \rrbracket_J = (P_v(e_i))_{i \in J} \) and outputting the evaluations \( (P_v(\omega_1), \ldots, P_v(\omega_s)) = v \). The packed Shamir’s secret sharing scheme is hence a \((\ell, d + 1, N)\)-quasi-threshold secret sharing scheme since, whenever \( s > 1 \), we have a gap between the privacy threshold \( (\ell) \) and the number of shares \( (d + 1 = \ell + s) \) allowing full recovery of \( v \).

**Vector sharings.** For a tuple \( v \in \mathbb{F}^{|v|} \) of length \( |v| > s \), a \((s, d, N)\)-packed sharing of \( v \) is defined as

\[
\llbracket v \rrbracket = \left( \begin{array}{c}
\llbracket (v_1, \ldots, v_s) \rrbracket \\
\llbracket (v_{s+1}, \ldots, v_{2s}) \rrbracket \\
\vdots
\end{array} \right) \in (\mathbb{F}^s)^{|v|},
\]

where \( |v|_s = \lceil |v|/s \rceil \) is the number of \( s \)-tuples composing \( v \) (where \( v \) is padded with 0’s or garbage if \( |v| \) is not a multiple of \( s \)) and the number of sharings in the vector sharing. (For such a vector sharing the row vs. column notations of tuples is used interchangeably without effect.) A vector sharing gives rise to a vector polynomial \( P_v \in (\mathbb{F}[X])^{|v|}_s \) defined as \( P_v(X) = (P_v^{(1)}(X), P_v^{(2)}(X), \ldots) \) where \( P_v^{(i)} \in \mathbb{F}[X] \) is the polynomial arising in the Shamir’s secret sharing of the \( i \)-th packed tuple \( (v_{i(s-1)+1}, \ldots, v_{is}) \).

\(^3\) We also consider the “evaluation point” \( e_i = \infty \). For this special evaluation point, the evaluation \( P_v(\infty) \) of a polynomial \( P_v \in \mathbb{F}[X] \) is defined as the leading degree coefficient of \( P_v \).
Linear and multiplicative homomorphisms. Consider the packed sharings \([v_1], [v_2]\) of tuples \(v_1, v_2 \in F^s\). Then for any scalars \(a_1, a_2 \in F\) we have that \(a_1 \cdot [v_1] + a_2 \cdot [v_2]\) is a sharing of \(a_1 \cdot v_1 + a_2 \cdot v_2\) (where the scalars \(a_i\) are multiplied to each coordinate of the tuple \(v_i\)). We also have that \([v_3] = [v_1] \cdot [v_2]\) (where the product is sharewise, namely \([v_3]_i = [v_1]_i \cdot [v_2]_i\), \(\forall i \in [1 : N]\)) is a sharing of \(v_3 = v_1 \circ v_2\), where \(\circ\) denotes the coordinate-wise product. Finally, for two vector sharings \([v_1], [v_2] \in (F^N)^t\) encoding packed tuples \(v_1, v_2 \in F^{s \cdot t}\), we denote by \([v_3] = ([v_1], [v_2])\) the “inner product” sharing which is defined as \([v_3] = \sum_{j=1}^t [v_{1,j}] \cdot [v_{2,j}]\) where \([v_{1,j}]\) (resp. \([v_{2,j}]\)) denotes the \(j^{th}\) packed sharing in the vector sharing \([v_1]\) (resp. \([v_2]\)). The resulting sharing \([v_3]\) encodes a tuple \(v_3 \in F_s\) for which the \(j^{th}\) coordinate is the inner product between the “column tuple” made of the \(i^{th}\) coordinates of the packs of \(v_1\) and the “column tuple” made of the \(i^{th}\) coordinates of the packs of \(v_2\).

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 \((x, w) \in R\) for some relation \(R\). To proceed, we shall use an MPC protocol in which \(N\) parties \(P_1, \ldots, P_N\) securely and correctly evaluate a function \(f\) on a secret witness \(w\) with the following properties:

- each party \(P_i\) takes a share \([w]_i\) as input, where \([w]\) is a sharing of \(w\);
- the function \(f\) outputs \text{ACCEPT} when \((x, w) \in R\) and \text{REJECT} otherwise;
- the protocol is \(\ell\)-private in the semi-honest model, meaning that the views of any \(\ell\) 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 \((x, w) \in R\). 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 \text{ACCEPT}.

Then the verifier randomly chooses \(\ell\) 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 \(\ell\) 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 \(1 - (\frac{\ell-2}{\ell})^N\), 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 \([KZ20b, BD20, BN20, BDK+21b, FJR22]\).

\(^4\) The following formalism is heavily borrowed from \([FR22]\).
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 $F$ so that all the manipulated variables (including the witness $w$) are tuples of elements from $F$. 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 $\langle w \rangle$ of the witness $w$ (one share per party), which is defined as

$$\langle w \rangle = (\langle w \rangle_1, \ldots, \langle w \rangle_N) \text{ s.t. } w = \sum_{i=1}^{N} \langle w \rangle_i .$$

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) $\varepsilon$ from a randomness oracle $O_R$. When calling this oracle, all the parties get the same random value $\varepsilon$.

**Receiving hint:** The parties receive a fresh sharing $\langle \beta \rangle$ (one share per party) from a hint oracle $O_H$. The hint $\beta$ can depend on the witness $w$ and the previous random values sampled from $O_R$.

**Computing & broadcasting:** The parties locally compute $\langle \alpha \rangle := \langle \varphi(v) \rangle$ from a sharing $\langle v \rangle$ where $\varphi$ is an $F$-linear function, then broadcast all the shares $\langle \alpha \rangle_1, \ldots, \langle \alpha \rangle_N$ to publicly reconstruct $\alpha := \varphi(v)$.

After $t$ rounds of the above actions, the parties finally output $\text{Accept}$ 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 $\text{Accept}$ with some probability $p$ over the random choice of the values $\varepsilon^1, \ldots, \varepsilon^t$ from the randomness oracle $O_R$. We refer to [FR22] for a formal definition or to Section 5 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 1: 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 1 is

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

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 1 includes:

- the input shares $(\langle w \rangle_i, \langle \beta^1 \rangle_i, \ldots, \langle \beta^t \rangle_i)$ of the opened parties. In practice, a seed $\text{seed}_i \in \{0, 1\}^\lambda$ 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 $\langle v \rangle$ is built as

$$\langle v \rangle_i \leftarrow \text{PRG(\text{seed}_i)} \text{ for } i \neq N$$

$$\langle v \rangle_N = v - \sum_{i=1}^{N-1} \langle v \rangle_i .$$
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
   \[
   \text{com}_j := \begin{cases} \text{Com}(\{ w \}, \{ \beta_j \}, \rho'_j) & \text{if } j = 1 \\ \text{Com}(\{ \beta_j \}, \rho'_j) & \text{if } j > 1 \end{cases}
   \]
   for all \( i \in \{1, \ldots, N\} \), for some commitment randomness \( \rho'_j \);
   (c) the prover sends
   \[
   h_j := \begin{cases} \text{Hash}(\text{com}_1, \ldots, \text{com}_i) & \text{if } j = 1 \\ \text{Hash}(\text{com}_j, \ldots, \text{com}_i, [\alpha^{i-1}]) & \text{if } j > 1 \end{cases}
   \]
   to the verifier;
   (d) the verifier picks at random a challenge \( \varepsilon_j \) and sends it to the prover;
   (e) the prover computes
   \[
   \alpha_j := \phi_j(\varepsilon_i)_{i \leq j}, (\alpha_i)_{i < j}, \text{com}_1, \ldots, \text{com}_j
   \]
   and recomposes \( \alpha^j \).

   The prover further computes \( h_{t+1} := \text{Hash}(\text{com}_1) \) 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 \( (\{ w \}, (\{ \beta_j \}, \rho'_j)_{i \neq i^*})_j, \text{com}_{i^*}, [\alpha^{i^*}], \ldots, [\alpha^t], \) to the verifier.

5. The verifier recomputes the commitments \( \text{com}_{i^*} \) and the broadcast values \( [\alpha^i], \) for \( i \in [N] \setminus \{ i^* \} \) and \( j \in [t] \) from \( (\{ w \}, (\{ \beta_j \}, \rho'_j)_{i \neq i^*})_j, \text{com}_{i^*}, [\alpha^{i^*}], \ldots, [\alpha^t] \) 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 \( t + 1 \),
   \[
   h_j \overset{\dagger}{=} \begin{cases} \text{Hash}(\text{com}_1, \ldots, \text{com}_j) & \text{if } j = 1 \\ \text{Hash}(\text{com}_j, \ldots, \text{com}_i, [\alpha^{i-1}]) & \text{if } j > 1 \\ \text{Hash}([\alpha^i]) & \text{if } j = t + 1 \end{cases}
   \]
   (b) the output of the MPC protocol is Accept, i.e.
   \[
   g(\alpha^1, \ldots, \alpha^t) \overset{\dagger}{=} 0.
   \]

Protocol 1: Zero-knowledge protocol - Application of the MPCitH principle to the general MPC protocol of [FR22].
Thus, instead of committing \( ([w]_i, [\beta^j]_i) \), the initial commitments simply include the seeds for \( i \neq N \), and \( \text{com}^i_1 \) becomes useless for \( j \geq 2 \) and \( i \neq N \). Formally, we have:

\[
\text{com}^i_j = \begin{cases} 
\text{Com}(\text{seed}; \rho^j_1) & \text{for } j = 1 \text{ and } i \neq N \\
\text{Com}([w]_N, [\beta^j]_N; \rho^j_N) & \text{for } j = 1 \text{ and } i = N \\
\emptyset & \text{for } j > 1 \text{ and } i \neq N \\
\text{Com}([\beta^j]_N; \rho^j_N) & \text{for } j > 1 \text{ and } i = N 
\end{cases}
\]

Some coordinates of the \( \beta^j \) might be uniformly distributed over \( \mathbb{F} \) (remember that the \( \beta^j \) are tuples of \( \mathbb{F} \) elements). We denote \( \beta^\text{unif} \) the sub-tuple composed of those uniform coordinates. In this context, the last share \( [\beta^\text{unif}]_N \) can be built as \( [\beta^\text{unif}]_N \leftarrow \text{PRG}(\text{seed}_N) \) so that a seed \( \text{seed}_N \) can be committed in \( \text{com}^i_N \) (instead of committing \( [\beta^\text{unif}]_N \)). This way the prover can save communication by revealing \( \text{seed}_N \) instead of \( [\beta^\text{unif}]_N \) whenever the latter is larger;

- the messages \( [\alpha^1], \ldots, [\alpha^t] \) broadcasted by the unopened party. Let us stress that one can sometimes save communication by sending only some elements of \( [\alpha^1], \ldots, [\alpha^t] \) 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 \( \text{com}_1^1, \ldots, \text{com}_t^1 \) (as explained above, we have \( \text{com}^i_j = \emptyset \) 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 \( \text{mseed} \) 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 \( \text{mseed} \). Then, one inductively labels the children of each node with the output of a standard PRG applied to the node’s label. The subseeds \( (\text{seed}_i)_{i \in [1:N]} \) 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 \( \text{seed}_{i^*} \), which one wants to keep secret (i.e., all the labels on the siblings of the path from \( \text{seed}_{i^*} \) to the root). Those labels allow the verifier to reconstruct the \( N-1 \) seeds \( (\text{seed}_i)_{i \in [1:N] \setminus \{i^*\}} \). Using GGM trees, the prover hence only needs to communicate \( \log_2(N) \lambda \)-bit seeds to the verifier.

### 2.5 Threshold Computation in the Head: Original Framework

In [FR23b], 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 \( \ell \)-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 \( i \in \{1, \ldots, N\} \), \( \text{com}_i \leftarrow \text{Com}([w]_i, \rho_i) \) where \( \rho_i \) is some commitment randomness. They send a hash digest \( h_1 \) of all \( \{\text{com}_i\}_{i \in [1:N]} \) to the verifier.
2. The prover emulates a subset \( S \subset \{1, \ldots, N\} \) 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 \( I \subset \{1, \ldots, N\} \) of \( \ell \) 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 \( i^* \in S \setminus I \).
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 $\text{com}_i$. Then, while opening the commitments, the prover further sends the authentication paths of the opened leaves $\{\text{com}_i\}_{i \in I}$ to allow the verifier to recompute and check $h_1$. In [FR23b], the soundness error of the obtained proof system is shown to be

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

where $p$ is the false positive probability of the underlying multiparty protocol. Since $\ell$ 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 [FR23b] 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 $2\lambda$ bits while a GGM sibling path is made of $\log N$ seeds of $\lambda$ bits.
- There is a soundness loss since a malicious prover can commit an invalid sharing (see Equation (3) vs. Equation (2)).

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 GGM Trees

3.1 General Technique

In this section, we only work on the case of the non-packed Shamir’s secret sharing for the sake of simplicity, namely Shamir’s secret sharing of parameters $(s, \ell, N)$ with $s = 1$. The general case of is described later in Section 5.2. So in the context of this section, whenever a tuple $w \in F^{|w|}$ is shared, the sharing $[w]$ is to be interpreted as a tuple of the sharings $([w_i])_{1 \leq i \leq |w|}$ of the coordinates $w_i \in F$. Without loss of generality, we shall assume that the evaluation point $\omega_1$ revealing the secret coordinate is fixed to $0$, that is by $P_{w_i}(0) = w_i$ (as in the original Shamir’s scheme).

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 $(N)$ 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 $\ell$-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 $\{s_T\}_{T \ni i}$ and converts them into one Shamir’s share $[w_i]$.

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

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

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

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

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

Let us denote $P_T \in \mathbb{F}[X]$ the unique degree-$\ell$ polynomial 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 $\lfloor w \rfloor_i \in \mathbb{F}^{|w|}$ as

$$\lfloor w \rfloor_i := \sum_{T \in S_i^N, i \not\in T} s_T \cdot P_T(e_i) + \begin{cases} \Delta w \cdot P_{T_0}(e_i) & \text{if } i \not\in T_0, \\ 0 & \text{otherwise}. \end{cases}$$

(4)

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

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

(5)

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

$$P(0) = \Delta w \cdot P_{T_0}(0) + \sum_{T \in S_i^N} s_T \cdot P_T(0)$$

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

We thus deduce that the shares $\{\lfloor w \rfloor_i\}_i$ forms a valid Shamir’s secret sharing of $w$, since they are evaluations of a degree-$\ell$ polynomial with $w$ as constant term.

**Remark 1.** Steps 1–4 consist in generating the $\ell$-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 $T \in S_i^N$, 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 $\lfloor w \rfloor_i$ as

$$\lfloor w \rfloor_i := \Delta w \cdot v^{(T_0)}_i + \sum_{T \in S_i^N, i \not\in T} s_T \cdot v^{(T)}_i.$$
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 \( \{\text{seed}_T\}_{T \in S^N} \) and \( \Delta w \). To open a party \( i \), the prover then needs to reveal all the seeds \( \{\text{seed}_T\}_{T \in S^N \setminus i \notin T} \) and \( \Delta w \) (if \( i \notin T_0 \)), from which the verifier can recompute the share \([w_i]\), thanks to Equation (4). In practice, the prover should reveal a subset \( I \) of \( \ell \) parties, so they will reveal

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

In other words, it means that the prover should reveal all the seeds except \( \text{seed}_I \). To proceed, they just need to reveal the sibling path of the hidden leaf \( \text{seed}_I \) in the GGM tree.

Protocol 2 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 \( \psi^j \) are the functions defining the hints.

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 (10)) on the plain input witness. Such “emulation” is often cheaper than a party emulation, since, in the MPCitH context, the prover already knows some expected values \( \alpha^j \) (which satisfy \( g(\alpha^1, \ldots, \alpha^\ell) = \text{ACCEPt} \) ) and can thus save some computation.

Protocol routines. Protocol 2 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 \( \text{GenerateSharingPolynomial}(\Delta x, (s_T)_{T \in S^N}) \) outputs the polynomial \( P_x \) defined as

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

- **GeneratePartyShare** (for some \( i \in [1 : N] \)) builds the share of the \( i \)th party, using Equation (4). It takes as inputs the randomness \( (s_T)_{T : i \notin T} \), together with the auxiliary value when necessary. Formally, the call \( \text{GeneratePartyShare}_i((s_T)_{T : i \notin T}, \Delta x) \) outputs

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

- **RecomputeSharing** builds the Shamir’s polynomial \( P_x \) from a plain value \( x \) and \( \ell \) 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 \( \text{RecomputeSharing}(x, ([x]_i)_{i \in I}) \) outputs the polynomial \( P_x \in (\mathbb{F}[X])^{[x]} \) defined as

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

\(^a\) A formal description of the general MPC protocol is also provided in Section 5.3 (see Protocol 3). The zero-knowledge protocol described here (Protocol 2) is an application of our TCitH framework with GGM tree to this general MPC protocol with the restriction that the \( \Phi \) round functions are made of inner functions \( \varphi^j \) which are linear.
1. The prover samples a root seed \( \text{seed} \in \{0,1\}^s \) and expands it through a GGM tree to obtain \( \{T\} \) leaf seeds \( \{\text{seed}_T\}_{T \in \mathcal{T}} \).

2. The prover expands each leaf seed, \( s_T' \leftarrow \text{PRG}(\text{seed}_T) \) for all \( T \), builds the auxiliary value \( \Delta w = w - \sum_T s_T' \), and computes
   \[
   P_v \leftarrow \text{GenerateSharingPolynomial}(\Delta w, \{s_T'\}_{T \in \mathcal{T}}).
   \]

3. The prover emulates "in her head" the MPC protocol.
   For \( j = 1 \) to \( t \):
   (a) the prover computes
   \[
   \beta^j = \psi^j(w, \{s_T'\}_{T \leq j}, r^j)
   \]
   using fresh uniform random tape \( r^j \).
   (b) the prover expands more randomness \( \{s_T'\}_{T \in \mathcal{T}} \) from the leaf seeds (such that \( |s_T'| = |\beta| \) for all \( T \)), builds the auxiliary value \( \Delta \beta^j = \beta^j - \sum_T s_T' \), and computes
   \[
   \com^j_T = \text{GenerateSharingPolynomial}(\Delta \beta^j, \{s_T'\}_{T \in \mathcal{T}}).
   \]
   (c) the prover expands some commitment randomness \( \rho^j_T \) from the leaf seeds (such that \( |\rho^j_T| = \lambda \) for all \( T \)) and computes the commitments
   \[
   P_v^j = \text{GenerateSharingPolynomial}(\Delta \beta^j, \{\rho^j_T\}_{T \in \mathcal{T}}).
   \]
   for all \( T \in \mathcal{T} \).
   (d) the prover sends
   \[
   h_j := \begin{cases}
   \text{Hash}(\{\com^j_T\}_T) & \text{if } j = 1 \\
   \text{Hash}(\{\com^j_T\}_T, P_v^{j-1}) & \text{if } j > 1
   \end{cases}
   \]
   to the verifier;
   (e) the verifier picks at random a challenge \( \psi^j \) and sends it to the prover;
   (f) the prover computes the plain broadcast
   \[
   \alpha^j = \text{coeff}(P_v^j) = \Phi^j(w, (\beta^j)_{k \leq j}).
   \]
   (g) the prover computes, for \( i \in [1 : t] \),
   \[
   \text{coeff}(P_v^j) = \Phi^j(\text{coeff}(P_v^j), (\text{coeff}(P_v^j))_{k \leq j}).
   \]
   The prover further computes \( h_{j+1} := \text{Hash}(P_v^j) \) and sends it to the verifier.

4. The verifier picks at random a subset \( I \subseteq [1 : N] \) of \( t \) parties (i.e. \( |I| = t \)) 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 \( \text{seed}_T \) in the GGM tree to the verifier, together with \( \Delta w, (\Delta \beta^j)_{k \in [1 : t]} \) when \( I \neq \mathcal{T} \).
   The prover further sends the digests of the unopened commitments \( \{\com^j_T\}_{T \in [1 : t]} \) and the plain broadcast values \( \{(\alpha^j)_{k \in [1 : t]}\} \).

6. The verifier recomputes the commitments \( \com^j_T \) for \( T \neq \mathcal{T} \) 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 \( \{s_T', s_{T+1}', \ldots, s_T''\}_{T \in \mathcal{T}} \) from seeds and build the share of the open parties: for all \( i \in I \),
   \[
   \{v_i\} = \text{GeneratePartyShare}(\{s_T', s_{T+1}', \ldots, s_T''\}_{T \in \mathcal{T}}, \Delta w)
   \]
   \[
   \{\beta_i\} = \text{GeneratePartyShare}(\{s_T', s_{T+1}', \ldots, s_T''\}_{T \in \mathcal{T}}, \Delta \beta^j)\]
   for all \( j \in [1 : t] \)
   where \( \Delta w \) and \( \{\Delta \beta^j\} \) 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 \( j \in [1 : t] \),
   \[
   [v_i^j] = \Phi^j([w_i], ([\beta_i^j]_{k \in [1 : t]})).
   \]
   Finally, the verifier can recompute the Shamir’s polynomials \( P_v^j \) of all \( ([\alpha^j])_{k \in [1 : t]} \), for all \( j \in [1 : t] \).
   \[
   P_v^j = \text{RecomputeSharing}_I([\alpha^j], ([\beta^j])_{k \in [1 : t]}).
   \]

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 \( t+1 \),
   \[
   h_j := \begin{cases}
   \text{Hash}(\{\com^j_T\}_T) & \text{if } j = 1 \\
   \text{Hash}(\{\com^j_T\}_T, P_v^{j-1}) & \text{if } 2 \leq j \leq t \\
   \text{Hash}(P_v^j) & \text{if } j = t+1
   \end{cases}
   \]
   (b) the output of the opened parties are \textbf{ACCEPT}, i.e.
   \[
   g(\alpha^1, \ldots, \alpha^t) = 0.
   \]

Protocol 2: Zero-knowledge protocol: application of the TCitH framework with GGM tree to the general MPC protocol of [FR22].
where \( i_\infty \) is the index such that \( e_{i_\infty} = \infty \), and
\[
\begin{align*}
I' := I \setminus \{ i_\infty \} & \quad \text{and} \quad c_\infty := \lceil x \rceil_{i_\infty} \quad \text{if} \quad i_\infty \in I, \\
I' := I & \quad \text{and} \quad c_\infty := 0 \quad \text{if} \quad i_\infty \notin I.
\end{align*}
\]

**Soundness and zero-knowledge analysis.** Let us analyze the soundness of Protocol 2. 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-\( \ell \) 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 (3)) 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 \(( N ) \) seeds \( \{ \text{seed}_T \} \) with an auxiliary value \( \Delta w \). 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 \( s_T \leftarrow \text{PRG}(\text{seed}_T) \) 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 \( \lceil w \rceil_i \)’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 \( \epsilon \) is
\[
\epsilon := \frac{1}{N} + p \cdot \left( 1 - \frac{1}{N} \right),
\]
which for \( \ell = 1 \) matches the soundness error of the MPCitH framework with additive sharing (see Equation (2)). 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 2 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 \( \text{seed}_I \) remains hidden and the plain witness \( w \) is masked by the hidden value \( s_I := \text{PRG}(\text{seed}_I) \). This ensures that the proof system leaks no information about the witness (provided that the underlying MPC protocol is \( \ell \)-private in the semi-honest model).

**Performances.** Let us analyze the communication cost of Protocol 2. 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 \( \text{seed}_I \) 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 \( \{ \text{com}_j \}_{j \in [1:t]} \), which cost \( t \cdot 2\lambda \) bits when \( I = T_0 \) and \( 2\lambda \) bits otherwise (since \( \text{com}_j^I = \emptyset \) for \( j > 0 \) and \( I \neq T_0 \)).

We obtain a total communication cost of
- when \( I \neq T_0 \),
\[
\text{Cost} = (t + 1) \cdot 2\lambda + \underbrace{\text{inputs}}_{h_1, h_2, \ldots, h_{t+1}} + \underbrace{\text{comm}}_{\text{seed}_I \text{ for } T \neq I} + \underbrace{\text{com}_j}_{\alpha^1, \ldots, \alpha^t} \cdot \underbrace{\log_2 \binom{N}{\ell}}_{\text{seed}_I \text{ for } T \neq I} + \underbrace{\log_2 \binom{N}{\ell}}_{\text{com}_j} + \underbrace{t \cdot 2\lambda}_{\text{com}_j^I, \ldots, \text{com}_j^I}.
\]
- when \( I = T_0 \),
\[
\text{Cost} = (t + 1) \cdot 2\lambda + \underbrace{\text{comm}}_{h_1, h_2, \ldots, h_{t+1}} + \underbrace{\text{com}_j}_{\alpha^1, \ldots, \alpha^t} \cdot \underbrace{\log_2 \binom{N}{\ell}}_{\text{seed}_I \text{ for } T \neq I} + \underbrace{t \cdot 2\lambda}_{\text{com}_j^I, \ldots, \text{com}_j^I}.
\]
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 \(\tau\) times such that \(\epsilon^\tau < 2^{-\lambda}\). The resulting averaged cost (in bits) is the following:

\[
\text{Cost} = (t + 1) \cdot 2\lambda + \tau \cdot \left( \binom{N}{\ell} - 1 \right) \cdot \text{_inputs} + \text{comm} + \lambda \cdot \log_2 \left( \binom{N}{\ell} - 1 + t \right) \cdot \left( \frac{N}{\ell} \right) \cdot 2\lambda.
\]

Comparison. Let us first consider the case \(\ell = 1\). We can check that Protocol 2 (with \(\ell = 1\)) achieves exactly the same communication cost and soundness as the MPCitH transformation with additive sharing and GGM tree (see, e.g., [FR23b, Section 3.2]). Moreover, in Protocol 2

- 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 \(\tau\) times to achieve a negligible soundness error, we obtain a proof system that emulates \(2\tau\) 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 \(\epsilon^1, \ldots, \epsilon^t\) across the \(\tau\) 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

  \[
  \binom{N}{\ell} - 1 \approx 1 \quad \text{and} \quad \binom{N}{\ell} - 1 + t \approx 1
  \]

  for all \(\ell\). Moreover, we can observe that the seed trees have \(2^{\lambda_0}\) leaves in both cases. We thus get that the communication cost is the same for any \(\ell\) (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 \(\ell\).

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 \((N \leq |F|)\) which, for a small field, might prevent reaching the target per-repetition soundness error \(2^{-\lambda_0}\). While increasing \(\ell\), 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: \( N \leq |F| \) (or \( N \leq |F| + 1 \) in some cases) as long as \( \ell < N - 1 \) (see Lemma 1 in [FR23b]). 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 \( \eta \) such that \( N \leq |F|^\eta \) and consider the field extension \( \mathbb{K} = \mathbb{F}[\delta]/f(\delta) \) where \( f \) is a public irreducible degree-\( \eta \) 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 \( \Delta w \) as

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

but we compute the shares \( \langle w \rangle_i \) using Equation (4) with parties’ evaluation points \( \{e_j\} \) living in the field extension \( \mathbb{K} \) (instead of living in \( \mathbb{F} \)). As consequence, the shares \( \{\langle w \rangle_i\} \) live in \( \mathbb{K}[\omega] \) instead of \( \mathbb{F}[\omega] \). Let us stress that the security properties still hold using this tweak:

- Zero-knowledge: the seed \( \text{seed}_I \) remains hidden as previously, and so the plain value \( w \) is masked by the hidden value \( s_I := \text{PRG}(\text{seed}_I) \).
- Soundness: the extractor of the proof of Theorem 2 (see Section 5) outputs the witness even if the polynomials live in a field extension.

**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 2) remains unchanged, since the plain values still live in \( \mathbb{F} \);
- The cost of running the MPC protocol on the \( \ell \) other coefficients of the Shamir’s polynomials (Step 3(g) of Protocol 2) is bigger. It is exactly \( \eta \) times bigger as we can decompose this computation into \( \eta \) 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 \( \varphi^j \), 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 \( x_{id} \) the \( \mathbb{F} \)-coordinate of \( x \in \mathbb{K} \) corresponding to the coefficient of the term \( \delta^d \) when decomposing \( x \in \mathbb{K} \) in the \( \mathbb{F} \)-basis \( (1, \delta, \ldots, \delta^{\eta-1}) \), we have:

\[
\text{coeff}_i(P_{aw})|_d = \varphi^j(\text{coeff}_i(P_w), (\text{coeff}_i(P_{\beta k}))_{k \leq j})|_d
\]

\[
= (A_0 \cdot \text{coeff}_i(P_w))|_d + \sum_{k \leq j} (A_k \cdot \text{coeff}_i(P_{\beta k}))|_d + \text{coeff}_i(P_b)
\]

\[
= A_0 \cdot (\text{coeff}_i(P_w))|_d + \sum_{k \leq j} A_k \cdot (\text{coeff}_i(P_{\beta k}))|_d + \text{coeff}_i(P_b)
\]

\[
= \varphi^j(\text{coeff}_i(P_w)|_d, (\text{coeff}_i(P_{\beta k})|_d)_{k \leq j})
\]

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 \( \eta \) times more expensive. To sum up, when lifting in a field extension of degree \( \eta \), we obtain the same communication cost, but emulating the MPC protocol is as expensive as emulating

\[1 + \ell \cdot \eta\] parties for the prover,

\(^7\) 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 \( \ell \).
\(-\ell \cdot \eta \) parties for the verifier.

We can observe that taking \(\eta = 1\) corresponds to the zero-knowledge proof system of the previous section. Taking \(\eta\) larger does not change the communication cost but the emulation phase becomes more expensive. Despite this overhead, taking \(\eta\) larger than 1 can overcome the limitation of \(N \leq |F|\). Indeed, we can now execute the proof system with at most \(|K| := |F|^\eta\) parties. For instance, if the witness (and MPC computation) is defined over \(F_{16}\) 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 \(\eta\), the optimal strategy is to choose the minimal \(\eta\) satisfying \(N \leq |F|^\eta\).

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 \(F_{13}\).

\[
\text{Fig. 1: Emulation cost when working on } F_{13} \text{ using lifting and using } \ell > 1.
\]

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 \texttt{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 \(K\) and a vector from \(F^{|x|}\), or equivalently \(N \cdot \eta\) scalar multiplications between a value from \(F\) and a vector from \(F^{|x|}\). However, we can pack these 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
One may wonder what is the best choice for Remark 5. \( \text{sharing + lifting} \) TCitH framework always brings a better trade-off.

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

3.3 Global Comparison

We deduce that taking \( N_i \) as close as possible to \( \sqrt{N} \) leads to the optimal computational cost.
the original TCitH framework \cite{FR23b} 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.

<table>
<thead>
<tr>
<th>Additive-sharing MPCitH</th>
<th>TCitH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Merkle tree variant</td>
</tr>
<tr>
<td>Traditional</td>
<td>$\ell = 1$</td>
</tr>
<tr>
<td>Hypercube</td>
<td>Basic</td>
</tr>
<tr>
<td>Number of Emulations</td>
<td>$\approx \lambda \frac{N}{\log_2 N}$</td>
</tr>
<tr>
<td>(prover)</td>
<td>$\approx \lambda \frac{\ell + 1}{\log_2 (\ell)}$</td>
</tr>
<tr>
<td>Time Complexity (prover)</td>
<td>$O(\lambda \frac{N}{\log_2 N})$</td>
</tr>
<tr>
<td>Number of Emulations</td>
<td>$\approx \lambda \frac{\ell - 1}{\log_2 N}$</td>
</tr>
<tr>
<td>(verifier)</td>
<td>$\approx \lambda \frac{1}{\log_2 N}$</td>
</tr>
<tr>
<td>Time Complexity (verifier)</td>
<td>$O(\lambda \frac{N}{\log_2 N})$</td>
</tr>
<tr>
<td>Restriction</td>
<td>$N \leq</td>
</tr>
</tbody>
</table>

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 $F_2$).

- 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).

Remark 6. Let us mention that the framework TCitH with GGM trees is compatible with the principle of MPCitH with rejection proposed in \cite{FMRV22}. 

18
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:

AIMer [KHS+22, CCH+23], Biscuit [BKPV23], MIRA [ABC+23, ABB+23a], MiRitH [ARZV23, ABB+23b], MQOM [FR23a], 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.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>N</th>
<th>Size</th>
<th>Traditional</th>
<th>Hypercube</th>
<th>$\eta$</th>
<th>Our scheme</th>
<th>Saving</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIMer</td>
<td>16</td>
<td>5904</td>
<td>0.64</td>
<td>0.52</td>
<td>4</td>
<td>0.52</td>
<td>−0%</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>4176</td>
<td>4.53</td>
<td>3.22</td>
<td>8</td>
<td>3.22</td>
<td>−0%</td>
</tr>
<tr>
<td>Biscuit</td>
<td>16</td>
<td>6726</td>
<td>2.81</td>
<td>1.71</td>
<td>1</td>
<td>1.44</td>
<td>−16%</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>4758</td>
<td>17.71</td>
<td>4.65</td>
<td>2</td>
<td>4.24</td>
<td>−9%</td>
</tr>
<tr>
<td>MIRA</td>
<td>32</td>
<td>7376</td>
<td>74.95</td>
<td>15.02</td>
<td>2</td>
<td>8.64</td>
<td>−46%</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>5640</td>
<td>384.26</td>
<td>20.11</td>
<td>2</td>
<td>9.89</td>
<td>−51%</td>
</tr>
<tr>
<td>MiRitH-1a</td>
<td>16</td>
<td>7661</td>
<td>6.81</td>
<td>2.59</td>
<td>1</td>
<td>1.52</td>
<td>−41%</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>5665</td>
<td>54.15</td>
<td>6.60</td>
<td>2</td>
<td>5.42</td>
<td>−18%</td>
</tr>
<tr>
<td>MiRitH-1b</td>
<td>16</td>
<td>8800</td>
<td>11.22</td>
<td>4.04</td>
<td>1</td>
<td>2.11</td>
<td>−48%</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>6298</td>
<td>89.50</td>
<td>8.66</td>
<td>2</td>
<td>6.66</td>
<td>−23%</td>
</tr>
<tr>
<td>MQOM-gf31</td>
<td>32</td>
<td>7621</td>
<td>12.88</td>
<td>4.64</td>
<td>1</td>
<td>3.31</td>
<td>−20%</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>6345</td>
<td>96.41</td>
<td>11.27</td>
<td>2</td>
<td>8.74</td>
<td>−22%</td>
</tr>
<tr>
<td>MQOM-gf251</td>
<td>32</td>
<td>7809</td>
<td>8.56</td>
<td>3.05</td>
<td>1</td>
<td>2.16</td>
<td>−20%</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>6575</td>
<td>44.11</td>
<td>7.56</td>
<td>1</td>
<td>5.97</td>
<td>−21%</td>
</tr>
<tr>
<td>RYDE</td>
<td>32</td>
<td>7446</td>
<td>2.31</td>
<td>1.14</td>
<td>5</td>
<td>1.14</td>
<td>−0%</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>5956</td>
<td>12.41</td>
<td>4.65</td>
<td>8</td>
<td>4.65</td>
<td>−0%</td>
</tr>
<tr>
<td>SDitH-gf256</td>
<td>32</td>
<td>11515</td>
<td>16.85</td>
<td>4.90</td>
<td>1</td>
<td>3.07</td>
<td>−37%</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>8241</td>
<td>78.37</td>
<td>7.23</td>
<td>1</td>
<td>5.31</td>
<td>−27%</td>
</tr>
<tr>
<td>SDitH-gf251</td>
<td>32</td>
<td>11515</td>
<td>4.17</td>
<td>2.17</td>
<td>1</td>
<td>1.79</td>
<td>−18%</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>8241</td>
<td>19.15</td>
<td>7.53</td>
<td>1</td>
<td>6.44</td>
<td>−14%</td>
</tr>
</tbody>
</table>

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).

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

- PERK $\text{ABB}^+23a$ follows the shared-permutation framework $FJR23$ which differs from the standard MPCitH framework.
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 Degree-Enforcing Commitment for TCitH with Merkle Trees

The original TCitH framework \cite{FR23b} suffers a soundness loss because a malicious prover can commit to invalid Shamir’s secret sharings, i.e. sharings for which the underlying polynomials (obtained by interpolation from the sharings) are not of degree $\leq d = \ell + s - 1$ as expected. Specifically, for a constant $\ell$, the soundness of the original TCitH framework (given by Equation (3)) has an overhead of $O(N \cdot p)$ compared to the standard MPCitH soundness (given by Equation (2)) which might be significant in case the false positive probability $p$ is non-negligible.

In the previous section, we showed how to generate and commit Shamir’s secret sharings using GGM trees which solves the latter issue since the degree $d$ is “hardcoded” in this technique. However, the complexity of building such commitments linearly scales with $\binom{N}{d}$ while that of Merkle tree commitments only scales with $N$. This prevents using the GGM variant with parameters leading to a large binomial coefficient $\binom{N}{d}$. Moreover, as we will see later on, using packed secret sharing enables to achieve commitments (and zero-knowledge arguments) which are sublinear in the witness size with Merkle trees but not with GGM trees, making the former more appealing in some contexts. We would thus like to tweak the Merkle tree commitment of Shamir’s secret sharings in a way that ensures the verifier that the committed sharings are of valid degrees.

This section provides such a \textit{degree-enforcing commitment scheme}. We first recall the proximity test for interleaved codes of Ligero \cite{AHIV17,AHIV23} and analyze its application to the TCitH framework. Our analysis put forward some limitations of this test in our context which motivates the introduction of a tweaked version of this test which we integrate to the commitment scheme in order to obtain a better soundness. We finally provide some comparison of the two approaches.

4.1 Analysis of Ligero’s Proximity Test

Ligero \cite{AHIV17,AHIV23} introduces the idea of proximity test for interleaved codes to test the validity of (Reed-Solomon) codewords committed with Merkle trees. The idea is the following: let us have $n$ possibly invalid codewords $[w_1], \ldots, [w_n]$. If at least one of the codewords is $\epsilon$-far from any valid codeword (meaning that it differs from a valid codeword on at least $\epsilon$ positions), then so are nearly all the linear combinations of $[w_1], \ldots, [w_n]$. So, in order to detect the commitment of invalid codewords, one can request the opening of a random linear combination of the committed codewords and check that it forms a valid codeword. In case of success, one concludes that all the codewords are (close to) valid codewords with high probability, otherwise one concludes that there is at least one invalid codeword. Ligero uses such proximity test with Reed-Solomon codewords. Let us recall that a Reed-Solomon code encodes a word $v \in \mathbb{F}^{d+1}$ into a codeword in $\mathbb{F}^N$ by interpolating a degree-$d$ polynomial $P$ such that $(P(\omega_1), \ldots, P(\omega_{d+1})) = v$, for some fixed evaluation points $\{\omega_i\}$, and defining the associated codeword formed by $N$ evaluations $(P(e_1), \ldots, P(e_N))$ for fixed evaluation points $\{e_i\}$ disjoint from $\{\omega_i\}$. In other words, a Shamir’s secret sharing of $w = (w_1, \ldots, w_s) \in \mathbb{F}^s$ with privacy threshold $\ell$ is the Reed-Solomon codeword associated to $(w_1, \ldots, w_s, r_1, \ldots, r_{\ell})$ for random elements $r_1, \ldots, r_{\ell} \in \mathbb{F}$.

Ligero’s proximity test relies on a key lemma (see Lemma 4 in Appendix A) which we can rephrase with a sharing-based formulation as follows. Let $e$ be a positive integer such that $e < \frac{N-d}{\ell}$. Suppose that $[w_1], \ldots, [w_n]$ differ from $n$ valid Shamir’s secret sharings for strictly more than $e$ parties. We get that the probability that a linear combination of those sharings differs from a valid sharing for at most $e$ parties is
less than \( (e + 1)/|\mathbb{F}| \):

\[
\Pr_{\gamma_1, \ldots, \gamma_n \leftarrow \mathbb{F}} \left[ \sum_i \gamma_i \cdot \|w_i\| \right. \text{ differs from a valid sharing for at most } e \text{ parties } \left. \right] \leq \frac{e + 1}{|\mathbb{F}|}.
\]

This result has been further extended in [BCI+20] (see Theorem 3 in Appendix A) which supports larger values of \( e \), namely any \( e < N - d^2 \), for which the above probability is upper bounded by \( N/|\mathbb{F}| \).

We provide in Appendix A an analysis of the soundness error obtained while using Ligero’s proximity test in the TCitH framework. We show that, for an MPC protocol with false positive probability \( p \), we obtain a soundness error \( \epsilon \) satisfying:

\[
\epsilon \leq \max \left\{ \frac{N}{|\mathbb{F}|} + \frac{(N - \left\lceil \frac{\delta}{2} \right\rceil)}{(\frac{N}{\ell})}, \frac{f}{|\mathbb{F}|} + \frac{(N - f)}{(\frac{N}{\ell})} \cdot p + \frac{1}{(\frac{N}{\ell})} \cdot (1 - p) \right\} \quad \text{for } 0 \leq f \leq \left\lceil \frac{\delta}{4} \right\rceil - 1
\]

\[
\cup \left\{ \frac{N}{|\mathbb{F}|} + \frac{(N - f)}{(\frac{N}{\ell})} \cdot p + \frac{1}{(\frac{N}{\ell})} \cdot (1 - p) \right\} \quad \text{and } \left\lceil \frac{\delta}{4} \right\rceil \leq f \leq \left\lfloor \frac{\delta}{2} \right\rfloor - 1.
\]

In [AHIV17, AHIV23], the authors provide a simpler expression of the above soundness error by simply summing the largest terms in the above formula:

\[
\epsilon \leq \frac{N}{|\mathbb{F}|} + \frac{(N - \left\lceil \frac{\delta}{4} \right\rceil)}{(\frac{N}{\ell})} + p + \frac{1}{(\frac{N}{\ell})} \cdot (1 - p) \leq \frac{N}{|\mathbb{F}|} + \left( 1 - \frac{\left\lceil \frac{\delta}{4} \right\rceil}{N} \right)^{\ell} + p + \left( \frac{\ell}{N} \right)^{\ell}.
\]

**Limitations.** The main drawback of Ligero’s and the above analyses using proximity tests is that they require large values of \( \ell \) for the soundness error to be small whereas small values of \( \ell \) are better to minimize the size of zero-knowledge arguments for small statements. This merely comes from the term

\[
\frac{(N - \left\lceil \frac{\delta}{4} \right\rceil)}{(\frac{N}{\ell})} \approx \left( \frac{N + \ell}{\frac{N}{\ell}} \right)^{\ell},
\]

which is a lower bound of the right-side quantity of (31). This term is large and hence implies an important degradation of the soundness whenever \( \ell \) is small. For instance, when \( \ell = 1 \), this term is around 1/2, meaning that the soundness error is larger than 1/2. Therefore, using Ligero’s approach is insufficient to improve the soundness error \( \epsilon = \frac{1}{2} + p \cdot \frac{N}{2} \) obtained by the analysis of [FR23b] for \( \ell = 1 \) (see Equation (3) for the general formula).

To the best of our knowledge, all the proximity tests in the state of the art target contexts with large-degree polynomials and large evaluation domains, which are asymptotically better while the statement size increases. However, with the TCitH framework, we target “small to medium size” statements which might benefit from small-degree polynomials if one can enforce the degree of committed sharings without the above soundness degradation (with the particular case of \( \ell = 1 \) in some contexts). This motivates our proposal which we introduce hereafter.

### 4.2 Degree-Enforcing Commitment Scheme

We now describe our approach to avoid the aforementioned limitations of Ligero’s proximity test. Our tweak consists in adding an extra atomic round for the proximity test before the MPC protocol emulation, i.e., before receiving the MPC randomness from the verifier. With this extra round considered as part of the commitment, the committed shares can be split into two categories:

1. the shares consistent with the requested linear combinations;
2. the shares inconsistent with the requested linear combinations.
We stress that any opening including some share(s) of the second category leads to a verification failure. So denoting $E$ the set of share indices belonging to the first category, only the shares from $\{[w]_i\}_{i \in E}$ can be accepted by the verifier. Our goal is hence to enforce that the shares $\{[w]_i\}_{i \in E}$ form a valid Shamir’s secret sharing (i.e., a sharing of expected degree $d$). We show hereafter that our approach ensures this property (i.e., enforces the degree of committed shares) with probability at least $1 - \left(\frac{N}{t+2}\right)^{\eta}$, where $\eta$ is the number of parallel repetitions of the proximity test. By taking $\eta$ large enough, our commitment scheme hence enforces the degree of committed sharings with overwhelming probability.

Let us first formally define the notion of degree-enforcing commitment scheme. (We do not re-introduce the binding and hiding notions for such a commitment scheme which are similar to the standard notions.)

**Definition 2 (Degree-enforcing commitment scheme).** Let $N, d \in \mathbb{N}$. Let $\mathbb{F}$ be a finite field and $\{e_1, \ldots, e_N\} \subseteq \mathbb{F}$ a set of distinct evaluation points. An interactive degree-enforcing commitment scheme with parameters $(N, d, \mathbb{F})$ is a triplet $(\text{Commit}, \text{Open}, \text{Verif})$ defined as follows:

- **Commit** is a 3-pass 2-party protocol between a (stateful PPT) prover $P = (P_1, P_2)$ and a (PPT) verifier $V$ which takes as input a family of degree-$d$ polynomials $P_1, \ldots, P_n \in \mathbb{F}[X]$ and outputs a commitment $\text{com} = (\text{com}_1, \text{chal}, \text{com}_2)$ and an opening key $\text{key}$ computed as:
  
  $$(\text{com}_1, \text{state}) \leftarrow P_1(P_1, \ldots, P_n) ; \ \text{chal} \leftarrow V(\text{com}_1) ; \ (\text{com}_2, \text{key}) \leftarrow P_2(\text{chal}, \text{state})$$

- **Open** is a deterministic algorithm which takes as input a commitment $\text{com}$, opening key $\text{key}$ and a set of evaluation points $E \subseteq \{e_1, \ldots, e_N\}$ and returns a set of evaluations $\{P_i(e) ; 1 \leq i \leq N, e \in E\}$ and a proof $\pi$.

- **Verif** is a deterministic algorithm which takes as input a commitment $\text{com}$, a set of evaluations $\{P_i(e) ; 1 \leq i \leq N, e \in E\}$ and a proof $\pi$ and returns $\text{ACCEPT}$ or $\text{REJECT}$.

The scheme is

- **correct:** for any family of degree-$d$ polynomials $P_1, \ldots, P_n \in \mathbb{F}[X]$ and any set of evaluation points $E \subseteq \{e_1, \ldots, e_N\}$, we have:
  
  $$\Pr \left[ \text{Verif}(\text{com}, \{P_i(e)\}, \pi) = \text{ACCEPT} \right] = \left( \text{com}, \text{key} \right) \leftarrow \text{Commit}(\{P_i\}) \left( \{P_i(e)\}, \pi \right) = \text{Open}(\text{com}, \text{key}, E)$$

- **$\varepsilon$-degree-enforcing:** for any (stateful PPT) adversary $A = (A_1, A_2)$ we have:
  
  $$\Pr \left[ \text{Verif}(\{\text{com}_1, \text{chal}, \text{com}_2\}, \{P_i(e)\}, \pi) = \text{ACCEPT} \right. \wedge \left. (\deg P_1 > d) \lor \cdots \lor (\deg P_n > d) \right] \leq \varepsilon,$$

  where $\text{Interpol}(E, \{P_i(e)\})$ returns the polynomials $P_1, \ldots, P_n$ interpolated from the evaluations $\{P_i(e)\}$, with $e \in E$. The above probability is over the randomness of $A$ and $\text{Commit}, V$.

We now describe the degree-enforcing commitment scheme we obtain by composing the Merkle commitment of polynomial evaluations (a.k.a. Shamir’s secret sharings) with an atomic round of proximity test (which is performed $\eta$ times in parallel for some parameter $\eta \in \mathbb{N}$):

- **Commit:** To commit degree-$d$ polynomials $P_1, \ldots, P_n \in \mathbb{F}[X]$, the 3-pass commitment protocol runs as follows:
  
  1. For all $j \in \{1, \ldots, N\}$, let $u_j = (P_1(e_j), \ldots, P_n(e_j))$. $\text{Commit}, P_1$ computes $\text{com}_1$ as the root of the Merkle tree with leaves $u_1, \ldots, u_N$ and $\text{state}$ as the set of polynomials $(P_1, \ldots, P_n)$.
  2. On input $\text{com}_1$, $\text{Commit}, V_1$ picks random coefficients $\gamma_{1,1}, \ldots, \gamma_{\eta,n} \in \mathbb{F}$ and returns $\text{chal} = \{\gamma_{k,i}\}$. 

22
3. On input chal = \{γ_{k,i}\}, state = (P_1, \ldots, P_n), \text{Commit}P_2 computes polynomials

\[ R_k = \sum_{i=1}^{n} γ_{k,i} \cdot P_i \quad \text{for all } k \in \{1, \ldots, η\} \]

and returns \text{com}_2 = (R_1, \ldots, R_η) and key = (P_1, \ldots, P_n).

- Open: On input a commitment \text{com} = (\text{com}_1, \text{chal}, \text{com}_2), an opening key key = (P_1, \ldots, P_n) and a set of evaluation points \(E\), returns the evaluations \(\{P_i(e) : 1 ≤ i ≤ n, e ∈ E\}\) (i.e., the evaluations of the Merkle leaves \{u_j ; e_j ∈ E\}) and a proof \(π\) made of the authentication paths of these leaves w.r.t. the Merkle root \text{com}_1.

- Verif: On input a commitment \text{com} = (\text{com}_1, \text{chal}, \text{com}_2), with chal = \{γ_{k,i}\} and \text{com}_2 = (R_1, \ldots, R_η), a set of evaluations \(\{P_i(e) : 1 ≤ i ≤ n, e ∈ E\}\) and a proof \(π\), performs the following checks:
  1. For all \(j\) s.t. \(e_j ∈ E\), verify the authentication path from \(π\) for \(u_j = (P_1(e_j), \ldots, P_n(e_j))\) w.r.t. the Merkle root \text{com}_1.
  2. For all \(e ∈ E\) and all \(k ∈ \{1, \ldots, η\}\), verify the equality \(R_k(e) = \sum_{i=1}^{n} γ_{k,i} \cdot P_i(e)\).
  3. For all \(k ∈ \{1, \ldots, η\}\), verify the degree \(deg R_k ≤ d\).

**Theorem 1.** Assume that an adversary is not able to produce a hash collision. The above commitment scheme is \(ε\)-degree-enforcing with

\[ ε = \left(\frac{N}{d+2}\right)^\eta \]

**Proof.** Along this proof, we denote vector polynomials \(P(X) = (P_1(X), \ldots, P_n(X)) \in (\mathbb{F}[X])^n\) and \(R(X) = (R_1(X), \ldots, R_η(X)) \in (\mathbb{F}[X])^η\) and define their degrees to be \(deg P = \max_i deg P_i\) and \(deg R = \max_i deg R_i\).

We consider an adversary \(A\) against the degree-enforcing game. We assume \(A\) cannot produce hash collisions which implies that, for any given Merkle commitment \text{com}_1, \(A\) can open a single valid leaf \(u_i = P(e_i)\) for each index \(i\). Such an adversary can open the degree-enforcing game if, after receiving the random challenge \text{chal} = \{γ_{k,i}\}, they can come up with a set of evaluation points \(E\) and a vector polynomial \(R(X)\) such that

1. \(R\) is consistent with \(P\) on \(E\), i.e., \(R(e) = (γ_{k,i}) \cdot P(e), \forall e ∈ E, \) where \((γ_{k,i})\) is the matrix composed by the coefficients \{γ_{k,i}\},
2. the interpolation \(P^{(E)} = \text{Interpol}(E, \{P(e) : e ∈ E\})\) is of degree \(deg P^{(E)} > d\).

Consider the set \(E\) defined as

\[ E = \left\{ E \subseteq \{e_1, \ldots, e_n\} \mid |E| = d + 2 \land deg P^{(E)} > d \right\} \]

Let us remark that this set is fully defined by the Merkle commitment \text{com}_1 from the adversary. Let us also remark that, because of the second condition above, the set of evaluation points returned by the adversary is necessarily a superset of some \(E ∈ E\). For the adversary to win, there should hence exist a set \(E ∈ E\) for which

\[ R^{(E)} := (γ_{k,i}) \cdot P^{(E)} \]

is of degree \(deg R^{(E)} ≤ d\). We can thus upper bound the success probability of the adversary by:

\[
\Pr\left[ \exists E ∈ E : deg R^{(E)} ≤ d \right] \leq \sum_{E ∈ E} \Pr\left[ deg R^{(E)} ≤ d \right] \tag{6}
\]

\[
\leq \sum_{E ∈ E} \prod_{k=1}^{η} \Pr\left[ deg R_k^{(E)} ≤ d \text{ with } R_k^{(E)} := \sum_{i=1}^{n} γ_{k,i} \cdot P_i^{(E)} \right] \tag{7}
\]

\[
\leq \sum_{E ∈ E} \left( \frac{1}{|\mathbb{F}|} \right)^η = \left( \frac{N}{d+2} \right)^\eta \tag{8}
\]
where \((7)\) holds from the union bound, \((7)\) holds by independence of the \(\gamma_{i,j}\)'s and \((8)\) holds because the probability that a random linear combination \(R_k^{(E)} := \sum_{i=1}^{\ell} \gamma_{k,i} \cdot F_i^{(E)}\) with at least one \(F_i^{(E)}\) of degree > \(d\) satisfies \(\deg R_k^{(E)} \leq d\) with probability at most \(1/|\mathbb{F}|\). This concludes the proof.

\[\square\]

### 4.3 Benefits of the Degree-Enforcing Commitment Scheme for the TCitH Framework

We discuss hereafter the benefits of using the degree-enforcing commitment scheme described in the previous section in the TCitH framework.

The obtained soundness error can be deduced by noting that in the presence of a malicious adversary, one of the two following cases occurs:

- Either the set of committed shares \(\{[w]_i\}_{i \in E}\) which are consistent with the proximity test do not form a valid Shamir’s secret sharing. According to Theorem \([1]\) this event occurs with probability at most \((\begin{array}{c} N \\ \ell + 2 \end{array})/|\mathbb{F}|^\eta\).
- Or, all the shares \(\{[w]_i\}_{i \in E}\) consistent with the proximity test form a valid Shamir’s secret sharing.

In that case, the probability that the verifier accepts the proof (for an incorrect witness \(w\)) is at most

\[\frac{1}{\ell} + p \cdot \left(1 - \frac{1}{\ell}\right).\]

We thus obtain that the soundness error is upper bounded by:

\[\left(\frac{\begin{array}{c} N \\ \ell + 2 \end{array}}{|\mathbb{F}|^\eta}\right) + \frac{1}{\ell} + p \cdot \left(1 - \frac{1}{\ell}\right). \tag{9}\]

Let us provide further insights about why adding an extra round of proximity test as part of the commitment allows us to improve the soundness. The crucial aspect of this tweak is that it constrains the malicious prover to choose the linear combinations \(\{R_k\}\) depending on the MPC randomness. They can therefore use the following strategy: (1) commit an invalid \([w]_j\) such that \(\{[w]_j\}_{j=1}^{\ell + 1}\) encode \((\begin{array}{c} N \\ \ell + 1 \end{array})\) different witnesses, (2) upon receiving the MPC randomness, select one of these sharings \([w]_j\) for which a false positive event occurs (i.e., for which the MPC randomness is such that the protocol accepts whereas the soundness error of Ligero (according to our refined analysis of Section \([4.1]\)) and that of TCitH with degree-enforcing commitment scheme with respect to an increasing \(\ell\) and for a number of parties \(N = 256\).

As a particular benefit of our approach, the obtained soundness error (Equation \((9)\)) avoids the bottleneck term \((\begin{array}{c} N \\ \ell \end{array})/|\mathbb{F}|^\eta\) arising in Ligero’s soundness error (and which cannot be decreased by increasing the number \(\eta\) of parallel repetitions of the proximity test). To illustrate this comparison, we draw in Figure \(2a\) the soundness error of Ligero (according to our refined analysis of Section \([4.1]\)) and that of TCitH with degree-enforcing commitment scheme with respect to an increasing \(\ell\) and for a number of parties \(N = 256\).

Let us further discuss the impact of our approach in terms of proof size. To achieve the optimal TCitH soundness error, we should take \(\eta\) large to make the term \((\begin{array}{c} N \\ \ell + 2 \end{array})/|\mathbb{F}|^\eta\) negligible, i.e., \(\leq 2^{-\lambda}\) for a security parameter \(\lambda\). Namely, we should take \(\eta \geq \frac{\lambda + \log_2\left(\begin{array}{c} N \\ \ell \end{array}\right)}{\log_2|F|}\) while communicating \(\eta \cdot d\) elements of \(F\) in the proof for the random linear combinations, where \(d\) is the degree of the committed sharings (which is \(d = \ell\) for the standard case and \(d = \ell + s - 1\) when packing is used – see Section \([3]\)). Paying this communication overhead for the improved soundness is not asymptotically optimal when \(N\) and \(\ell\) increase. However, for “small to medium size” statements, our approach achieves better sizes. This is illustrated on Figure \(2b\), for the MQ protocol \([FR23a]\) (which verifies an MQ statement) for which the best size obtained under Ligero with degree-enforcing is about three times smaller than the best size obtained with Ligero. We provide further results in Section \([6.1]\) which show the same kind of improvements for general arithmetic circuits.
Fig. 2: Comparison between Ligero and TCitH with degree-enforcing commitment scheme (\(N = 256, \eta = 1,\)
\(p\) negligible).

5 Extended TCitH Framework

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 [FR23b]) and then describe our extended TCitH
proof system in two variants (Merkle tree vs. GGM tree).

5.1 MPC Model

We consider an MPC protocol that performs its computation on a base finite field \(F\). We rely on packed
Shamir’s secret sharing with pack size \(s\). All the manipulated variables (including the witness \(w\)) are assumed
to be tuples of elements from \(F\). The size of such a tuple \(v\) shall be denoted \(|v|\) so that \(v \in F^{(|v|)}\). As we shall
make use of packed secret sharing with some fixed pack size parameter \(s\), we shall split such a tuple \(v\) into
several \(s\)-tuples. By convention and according to Definition 1, a sharing \(J\) of \(v \in F^{(|v|)}\) is a vector of packed
sharings:

\[ J_{v}(K) = (J_{v_{1}}, \ldots, J_{v_{|v|/s}}), \ldots) \in (F^{N})^{s}\]

with \(|v|_s\) the number of \(s\)-tuples composing \(v\) which is \(|v|_s = \lceil |v|/s \rceil\).

As in the MPC model formalized in [FR23b] and overviewed in Section 2.3 in our extended MPC model,
the parties jointly run the computation of a function

\[
    f(w, \varepsilon, \beta) = \begin{cases} 
    \text{ACCEPT} & \text{if } g(\alpha) = 0, \\
    \text{REJECT} & \text{otherwise},
    \end{cases}
\]

with \(\varepsilon = (\varepsilon^1, \ldots, \varepsilon^t)\) the random values from a randomness oracle \(O_R\), \(\beta = (\beta^1, \ldots, \beta^t)\) the hints from a
hint oracle \(O_H\), \(\alpha = (\alpha^1, \ldots, \alpha^t) := \Phi(w, \varepsilon, \beta)\) the broadcasted and publicly recomputed values (for some
function \(\Phi\) made explicit below), and \(g\) some final check function. The main differences with the previous
model are the following:

1. The considered protocols apply to packed Shamir’s secret sharings of the form

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

for \(v \in F^s\) as defined in Section 2.1. We recall that \(d + 1\) shares are sufficient to reconstruct \(P_v\) and hence
recover \((v_1, \ldots, v_s) = (P_v(\omega_1), \ldots, P_v(\omega_s))\) whenever the polynomial \(P_v\) is of degree \(\deg(P_v) \leq d\) (the
initial sharings are of degree $d \leq s + \ell - 1$ but the degree can grow throughout the protocol execution as explained hereafter). Manipulated variables can also be tuple larger than $s$, i.e. of size $|v| > s$ or equivalently $|v| > 1$, in which case $\llbracket v \rrbracket$ is a $|v|$-tuple of packed sharings.

2. The round computation functions $\Phi^1, \ldots, \Phi^\ell$ (which are used to compute the broadcast values $\alpha^1, \ldots, \alpha^\ell$) 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 $\ell + s - 1$ (the degree of the input witness sharing). In the following, we denote

$$\deg(\llbracket v \rrbracket) = \deg(P_v)$$

the degree of the polynomial $P_v$ underlying a Shamir’s secret sharing $\llbracket v \rrbracket$, also called the degree of $\llbracket v \rrbracket$. While the input sharing of the protocol is a fresh degree-$(\ell + s - 1)$ sharing, the computation of non-linear round functions $\varphi$ might produce sharings $\|\alpha\|$ of higher degrees.

**Protocol ingredients.** The considered MPC protocol is as follows. At the start, the parties receive as input a fresh $(s, \ell, N)$-packed Shamir’s secret sharing (i.e. a sharing of degree $\ell + s - 1$) $\llbracket w \rrbracket$ of the witness $w$ (one share $\llbracket w \rrbracket_i$, 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) $\varepsilon \in \mathbb{P}^{|s|}$ from a randomness oracle $O_R$. When calling this oracle, all the parties get the same random value $\varepsilon$. Upon application of the TCitH transform, these random values are provided by the verifier as challenges.

- **Receiving hint:** Optionally, the parties receive a sharing $\llbracket \beta \rrbracket$ from a hint oracle $O_H$. For some function $\psi$, the plain hint $\beta$ is computed as $\beta := \psi(w, \varepsilon; r)$ where $\varepsilon = (\varepsilon^1, \varepsilon^2, \ldots)$ is made of the previous random values from $O_R$ and where $r$ is fresh randomness. A fresh degree-$d$ $s$-packed sharing of $\beta$ is then generated and distributed to the parties (one share per party), for some hint degree $d$ (which might be different from $s + \ell - 1$). Upon application of the TCitH transform, the hint $\llbracket \beta \rrbracket$ is generated and committed by the prover.

- **Computing & broadcasting:** The parties compute $\llbracket \alpha \rrbracket := \varphi(\llbracket w \rrbracket, \llbracket \beta \rrbracket, \llbracket \theta \rrbracket)$, which means that they locally compute

$$\llbracket \alpha \rrbracket_i := \varphi(\llbracket w \rrbracket_i, \llbracket \beta \rrbracket_i, \llbracket \theta \rrbracket_i), \quad \forall i \in [1 : N]$$

where $\llbracket \beta \rrbracket = (\llbracket \beta^1 \rrbracket, \llbracket \beta^2 \rrbracket, \ldots)$ is made of the previous outputs of $O_H$ and $\llbracket \theta \rrbracket$ is a fixed (publicly known) sharing. The parties then broadcast $\llbracket \alpha \rrbracket$ and publicly recompute $\alpha$.

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

**Example 1.** A broadcast value could be computed as

$$\llbracket \alpha \rrbracket := \llbracket w_1 \rrbracket \cdot \llbracket w_2 \rrbracket$$

with $w_1, w_2 \in \mathbb{F}^s$ two size-$s$ coordinate packs of the witness. Here the function $\varphi$ is simply the product of these two coordinate packs: $\alpha$ will be equal to $w_1 \cdot w_2$, where $\cdot$ is the coordinate-wise multiplication. This sharing product is computed sharewisely: $\llbracket \alpha \rrbracket_i := \llbracket w_1 \rrbracket_i \cdot \llbracket w_2 \rrbracket_i$, for every $i$. The obtained sharing $\llbracket \alpha \rrbracket$ has underlying polynomial $P_\alpha := P_{w_1} \cdot P_{w_2}$, with $P_{w_1}, P_{w_2}$ the polynomials underlying the sharings $\llbracket w_1 \rrbracket, \llbracket w_2 \rrbracket$. We hence have $\deg(\llbracket \alpha \rrbracket) = 2(\ell + s - 1)$. Upon application of the TCitH framework, the prover must communicate $2(\ell + s - 1) + 1$ shares of $\llbracket \alpha \rrbracket$ 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 $\alpha^1, \ldots, \alpha^\ell$. They output ACCEPT if $g(\alpha^1, \ldots, \alpha^\ell) = 0$ 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 (i.e., the sharing of a requested hint). 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. Each outer round $j \in [1 : t]$ performs $t_{j}^{(in)}$ inner rounds of locally computing and broadcasting $[\alpha^{j,k}] = \varphi^{j,k}(\lfloor w \rfloor, \lfloor \beta^{1} \rfloor, \ldots, \lfloor \beta^{t} \rfloor, \lfloor \theta^{j,k} \rfloor)$ 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 $\Phi^{j}$ the global iterative functions obtained from those $t_{j}^{(in)}$ inner rounds:

$$[\alpha^{j}] = ([\alpha^{j,1}], \ldots, [\alpha^{j,t_{j}^{(in)}}]) = \Phi^{j}(\lfloor w \rfloor, \lfloor \beta^{1} \rfloor, \ldots, \lfloor \beta^{t} \rfloor),$$

where the fixed sharings $[\theta^{j,k}]$ are “hardcoded” in the definition of $\Phi^{j}$.

Following this structure, our general MPC protocol is depicted in Protocol 3.

---

1. The parties take as input an $(s, \ell, N)$-packed Shamir’s secret sharing $\lfloor w \rfloor$.
2. For $j = 1$ to $t$ (outer rounds):
   
   (a) For some function $\psi^{j}$ and some sharing degree $d_{\beta_{j}}$, the parties get a fresh degree-$d_{\beta_{j}}$ $s$-packed sharing $[\beta^{j}]$ from the hint oracle $O_{H}$, such that
   
   $$\beta^{j} \leftarrow \psi^{j}(w, \varepsilon^{1}, \ldots, \varepsilon^{s-1}; r^{j})$$
   
   for a uniform random tape $r^{j}$. (For this packed sharing generation, the size of the randomness is set to $d_{\beta_{j}} = s + 1$ so that the degree of the generated sharing is $d_{\beta_{j}}$. In case $d_{\beta_{j}} = s + \ell - 1$ as for the input sharings, the size of the randomness is $\ell$ and $[\beta^{j}]$ is a fresh $(s, \ell, N)$-packed sharing.)
   
   (b) The parties get a common random $\varepsilon^{j}$ from the oracle $O_{R}$.
   
   (c) Inner rounds: The parties locally compute and broadcast
   
   $$[\alpha^{j}] := \Phi^{j}(\lfloor w \rfloor, \lfloor \beta^{1} \rfloor, \ldots, \lfloor \beta^{t} \rfloor)$$
   
   and publicly recompute $\alpha^{j}$.
   
   This step is detailed in Protocol 4.
3. The parties finally accept if $g(\alpha^{1}, \ldots, \alpha^{t}) = 0$ and reject otherwise.

---

**False positive probability.** The functionality computed by the protocol deterministically depends on the broadcasted values $\alpha$ (through the function $g$), which in turn deterministically depend on the input witness $w$, the sampled random values $\varepsilon$, and the hints $\beta$. It is formally given by the function $f$ from Equation (10), with $\alpha = \Phi(w, \varepsilon, \beta)$ where $\Phi$ is the deterministic function mapping $(w, \varepsilon, \beta)$ to $\alpha$ (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
For $k = 1$ to $t^{(in)}_j$ (inner rounds):

- For some $F$-polynomial function $\varphi^{j,k}$, the parties compute:
  \[
  \lbrack \alpha^{j,k} \rbrack := \varphi^{j,k}(\lbrack w \rbrack, \lbrack \beta^1 \rbrack, \ldots, \lbrack \beta^j \rbrack, \lbrack \theta^{j,k} \rbrack)
  \]
  where $\lbrack \theta^{j,k} \rbrack$ is some fixed sharing.

- The parties broadcast $\lbrack \alpha^{j,k} \rbrack$ and publicly reconstruct $\alpha^j$.

**Remark 7.** We use the terminology of false positive probability to differentiate from the soundness error of the proof of knowledge which is obtained by applying the MPCitH or TCitH transform to such MPC protocol. We further stress that the notion of false positive probability is different from the notion of robustness error existing in the MPC literature. The robustness error corresponds to our false positive probability in the presence of an adversary that can actively corrupt a number of parties and usually in the absence of a hint oracle. In our context, we do not require the MPC protocol to be robust (i.e. it is not required to satisfy any security property in the presence of an active adversary) but we consider a hint oracle which can be malicious: the false positive probability holds for any adversarial choice of the hints. Moreover, in contrast to an MPC context where the robustness error is required to be negligible, we can take advantage of MPC protocols with non-negligible false positive probability.

### 5.2 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 packed secret sharing and 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 GGM tree (TCitH-GGM) as presented in Section 5 and TCitH with Merkle tree (TCitH-MT) and degree-enforcing commitment as introduced in Section 7.
**Tweaking hint commitments.** The proof system described in the original TCitH framework 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 \(\bar{\beta}^2 \in \mathbb{F}[\beta^2], \ldots, \bar{\beta}^t \in \mathbb{F}[\beta^t]\). In the TCitH-MT variant, these sharings are committed using the same Merkle tree. In the following rounds, to generate and commit a sharing of the \(j^\text{th}\) hint \(\beta^j\), 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 \([\bar{\beta}^j]\) of \(\beta^j\) using

\[
[\bar{\beta}^j] \leftarrow [\beta^j] + [\Delta\beta^j],
\]

where \([\Delta\beta^j]\) denotes the “constant sharing” corresponding to the degree-s polynomial \(P_{\Delta\beta^j}\) satisfying \(P_{\Delta\beta^j}(\omega_i) = \Delta\beta^j_i\) for all \(i \in [1 : s]\).

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 in practice.

- 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 \([\bar{\beta}^2], \ldots, [\bar{\beta}^t]\).

- The TCitH-MT variant uses the degree-enforcing commitment scheme introduced in Section 4 to enforce 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 (as introduced in Section 4 and further detailed below);

2. The verifier generates the randomness \(\varepsilon^1\) 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 prover generates the randomness \(\varepsilon^j\) 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 \([\alpha^1]_I, \ldots, [\alpha^t]_I\).

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 \([\alpha^1]_I, \ldots, [\alpha^t]_I\);\(^{9}\)

\(^{9}\) Here, whenever \(|\beta^j| > s\), the sharings \([\beta^j]_I, [\bar{\beta}^j]_I\) and \([\Delta\beta^j]_I\) are vector sharings and \(P_{\Delta\beta^j}\) is a vector polynomial in the sense of Definition 4. Then, \(P_{\Delta\beta^j}(\omega_i) = \Delta\beta^j_i\) is to be interpreted as the vector composed of the \(i\)th coordinates of the packs composing \([\Delta\beta^j]\).
the correct computation of the shares $\lceil \alpha \rceil_I$ from $\lceil u \rceil_I$ (and $\lceil \beta^1 \rceil_I$, \ldots, $\lceil \beta^t \rceil_I$);
- that $g(\alpha, \ldots, \alpha^t) = 0$ (i.e. that the protocol accepts).

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

Degree-enforcing commitment scheme (TCitH-MT). As explained in Section 4 in the original TCitH-MT framework a malicious prover might commit an invalid sharing, i.e., a sharing for which the shares do not correspond to the evaluations of a degree-$(s + \ell - 1)$ 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 in Section 4. We reformulate this commitment in the syntax of the extended TCitH framework and tweak it to ensure the zero-knowledge property, support sharings of different degrees (for hints) and optimize the communication.

We first describe the basic principle ignoring hint commitments for the sake of simplicity. At the beginning, the witness is extended with a random vector $u \in \mathbb{F}^{s \times s}$ so that the extended witness $(u, w)$ is shared and committed. By definition, the sharing $\lceil u \rceil$ is composed of $\eta$ packed secret sharings of random $s$-tuples. Once $\lceil u \rceil$, $\lceil w \rceil$ have been committed by the prover, the verifier samples a random matrix $\Gamma = (\gamma_{j,k})_{j,k}$ of dimensions $\eta \times |w|$, $s$. The prover then computes the sharing $\lceil \xi \rceil = \Gamma \cdot \lceil w \rceil + \lceil u \rceil$, namely the sharing defined as $\lceil \xi \rceil_i = \Gamma \cdot \lceil w \rceil_i + \lceil u \rceil_i$, for all $i \in [1 : N]$, where $\lceil w \rceil_i \in \mathbb{F}^{|w|}$ is the vector composed of the $i$th share of each packed sharing composing $\lceil w \rceil$ and $\lceil u \rceil_i \in \mathbb{F}^{|v|}$ is the vector composed of the $i$th share of each packed sharing composing $\lceil u \rceil$. The prover commits $\lceil \xi \rceil$ to the verifier by sending the hash value of the underlying vector polynomial $P_\xi$. The sharing $\lceil \xi \rceil$ will be later revealed to the verifier which can then check that $\lceil \xi \rceil$ is of right degree $s + \ell - 1$ and that the revealed shares well satisfy (12). This ensures that the committed sharings $\lceil u \rceil$, $\lceil w \rceil$ were of degree $s + \ell - 1$ with high probability as shown in Section 4.

The sharing $\lceil u \rceil$ is used to ensure the zero-knowledge property by masking $\lceil \xi \rceil$ so that revealing $\lceil \xi \rceil$ does not leak any information on $w$.

When hints are used, we must further ensure that the committed sharings $\lceil \beta^1 \rceil$, $\lceil \beta^2 \rceil$, \ldots, $\lceil \beta^t \rceil$ are of the right degrees, which might be different for the different hints. Let $d_{\beta_j}$ denote the degree of the hint $\lceil \beta^j \rceil$ and let $d_{\beta} = \max(s + \ell - 1, d_{\beta_1}, \ldots, d_{\beta_t})$. Our goal is to enforce that the polynomials $P_{w}$, $P_{\beta_1}$, \ldots, $P_{\beta_t}$, underlying the committed sharings are of right degrees $\text{deg}(P_w) = s + \ell - 1$, $\text{deg}(P_{\beta_j}) = d_{\beta_j}$ 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 where $P_w \in (\mathbb{F}[X])^{|w|}$ and $P_u \in (\mathbb{F}[X])^{|\eta|}$ are the vector polynomials underlying the packed sharing $\lceil w \rceil$ and $\lceil u \rceil$. To extend this to further polynomials with different degrees, we shall align all the polynomials to the degree $d_{\beta}$ by multiplying each polynomial $P$ by the monomial $X^{d_{\beta} - \text{deg}(P)}$. Thus, we define the global vector polynomial $Q \in (\mathbb{F}[X])^{|Q|}$ as

$$Q(X) := (X^{d_{\beta} - \ell} \cdot P_w(X) \mid X^{d_{\beta} - d_{\beta_1}} \cdot P_{\beta_1}(X) \mid \cdots \mid X^{d_{\beta} - d_{\beta_t}} \cdot P_{\beta_t}(X))$$

which is of length $|Q| = |w| + |\beta^1| + \cdots + |\beta^t|$. In this general setting, the mask sharing $\lceil u \rceil$ randomly generated and committed by the prover is of degree $d_{\beta}$ and the random matrix $\Gamma$ 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

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

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

1. The prover generates the sharing of the witness $[u]$, the sharing of the random mask $[u]$, the sharing of the first hint $[\beta^1]$ and sharings $[\beta^2], \ldots, [\beta^\ell]$ of uniform random vectors $\beta^j \in \mathbb{F}[\beta]^1$ for all $j \in [2 : \ell]$.
2. The prover commits these sharings using a Merkle tree. Specifically, they compute the leaf commitments $\text{com}_i := \text{Com}([u], [u], [\beta^1], [\beta^2], \ldots, [\beta^\ell]; \rho_i)$ and the root $h_1 := \text{MerkleRoot(}\text{com}_1, \ldots, \text{com}_N\text{)}$ and send $h_1$ to the verifier.
3. The verifier samples a random matrix $\Gamma$ of dimensions $\eta \times |Q|$ where $|Q| = |w|_s + |\beta^1|_s + \cdots + |\beta^\ell|_s$ and sends it to the prover.
4. The prover computes the degree-enforcing polynomial $P_\ell \in (\mathbb{F}[X])^n$ using Equation (14) and sends $h'_1 := \text{Hash}(P_\ell)$ to the verifier.

The rest of the protocol runs as overviewed above with the following tweaks. During the opening phase, the prover further reveal $P_u(e_i)$ for all $i \in S$ with $S$ some set of cardinality $|S| = d_\beta + 1 - \ell$ and disjoint of $I$ (the set of opened shares). During the final checks, the verifier computes $P_\ell(e_i) = \Gamma \cdot Q(e_i) + P_u(e_i)$ for all $i \in I$ from opened shares $[w], [u], [\beta^1], [\beta^2], \ldots, [\beta^\ell]$ (by definition $Q(e_i)$ and $P_u(e_i)$ are linear functions of these shares). From $\{P_\ell(e_i)\}_{i \in S \cup I}$ the verifier reconstructs $P_\ell$ by interpolation and check the hash $h'_1 = \text{Hash}(P_\ell)$.

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

To generate a pseudorandom degree-$d$ packed sharing $[x]$ of a value $x \in \mathbb{F}^s$, the expanded randomness $s_T$ is of length $(d - \ell + 1) \cdot \log_2 |\mathbb{F}|$. Then the underlying polynomial $P_x$ is defined as

$$P_x(X) = \sum_{k=1}^{s} \Delta x_k \cdot P_{T_0,k}(X) + \sum_{T \in S^N} \sum_{k=1}^{d-\ell+1} s_T^{(k)} \cdot P_{T,k}(X) \in (\mathbb{F}[X])^{[x]};$$

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

$$[x]_i := \sum_{T \in S^N, i \notin T} s_T^{(k)} \cdot P_{T,k}(e_i) + \left\{ \begin{array}{ll} \sum_{k=1}^{s} \Delta x_k \cdot P_{T_0,k}(e_i) & \text{if } i \notin T_0, \\ 0 & \text{otherwise.} \end{array} \right.$$
Protocol description. The zero-knowledge protocol obtained by applying our extended TCitH framework to the general MPC protocol (Protocol 3) is formally described in Protocol 5. 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 $\beta^1$, and a root seed $\text{rseed}$, and it generates the sharings $[w], [\beta^1]$ of $w$ and $\beta^1$, the random sharings $[w], [\beta^2], \ldots, [\beta^t]$ and a commitment $h_1$ of these sharings.

- OpenShares: This routine takes as input the witness $w$, the root seed $\text{rseed}$, the hint correction $\Delta_1$, the hint corrections $\{\Delta_i\}^t_{i=2}$ and a set $I \subseteq [1 : N]$ such that $|I| = \ell$, and it returns $\text{views}_I$ an opening of the shares in $I$ as well as $\text{decom}_I$, the necessary data to decommit $\text{views}_I$ (namely to check the consistency of $\text{views}_I$ with the commitment $h_1$). If the TCitH-GGM framework, $\text{views}_I$ is defined as the opening of the shares in $I$ as well as $\text{decom}_I$, the necessary data to decommit $\text{views}_I$ (namely to check the consistency of $\text{views}_I$ with the commitment $h_1$). In the TCitH-GGM framework, $\text{views}_I$ is defined as the opening of the shares in $I$ as well as $\text{decom}_I$, the necessary data to decommit $\text{views}_I$ (namely to check the consistency of $\text{views}_I$ with the commitment $h_1$). In the TCitH-MT framework, $\text{views}_I$ is defined as the opening of the shares in $I$ as well as $\text{decom}_I$, the necessary data to decommit $\text{views}_I$ (namely to check the consistency of $\text{views}_I$ with the commitment $h_1$). In the TCitH-MT framework, $\text{views}_I$ is defined as the opening of the shares in $I$ as well as $\text{decom}_I$, the necessary data to decommit $\text{views}_I$ (namely to check the consistency of $\text{views}_I$ with the commitment $h_1$). In the TCitH-MT framework, $\text{views}_I$ is defined as the opening of the shares in $I$ as well as $\text{decom}_I$, the necessary data to decommit $\text{views}_I$ (namely to check the consistency of $\text{views}_I$ with the commitment $h_1$). In the TCitH-MT framework, $\text{views}_I$ is defined as the opening of the shares in $I$ as well as $\text{decom}_I$, the necessary data to decommit $\text{views}_I$ (namely to check the consistency of $\text{views}_I$ with the commitment $h_1$). In the TCitH-MT framework, $\text{views}_I$ is defined as the opening of the shares in $I$ as well as $\text{decom}_I$, the necessary data to decommit $\text{views}_I$ (namely to check the consistency of $\text{views}_I$ with the commitment $h_1$). In the TCitH-MT framework, $\text{views}_I$ is defined as the opening of the shares in $I$ as well as $\text{decom}_I$, the necessary data to decommit $\text{views}_I$ (namely to check the consistency of $\text{views}_I$ with the commitment $h_1$). In the TCitH-MT framework, $\text{views}_I$ is defined as the opening of the shares in $I$ as well as $\text{decom}_I$, the necessary data to decommit $\text{views}_I$ (namely to check the consistency of $\text{views}_I$ with the commitment $h_1$). In the TCitH-MT framework, $\text{views}_I$ is defined as the opening of the shares in $I$ as well as $\text{decom}_I$, the necessary data to decommit $\text{views}_I$ (namely to check the consistency of $\text{views}_I$ with the commitment $h_1$).

- VerifyDecommitment: This routine takes as input an opening $\text{views}_I$, some associated decommitment data $\text{decom}_I$ and the set $I$ and it recomputes the commitment $h_1$.

- RetrieveShares: This routine takes as input an opening $\text{views}_I$ and the associated set $I$ and returns the witness shares $[w]_I$ and the hint shares $[[\beta^1]]_I$.

The formal description of these routines is given in Figure 3. In the formal description of OpenShares, some values must be retrieved from ($w, \beta^1, \text{rseed}$) which have already been computed in GenerateAndCommitShares. We denote this by ($w, \beta^1, \text{rseed}$) $\mapsto (\ldots)$. Of course, in practice, this computation does not need to be performed twice. Moreover, the routines in Figure 3 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,
- GetSiblingPath the subroutine which computes the sibling paths of the leaves indexed by $I$,
- RetrieveLeavesFromPath 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 which 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 $N$ (a.k.a. the number of parties), the privacy threshold $\ell$, the pack size $s$, the maximal degree of the broadcast sharings $d_\alpha$, the maximal degree for the hint sharings $d_\beta$ (wlog. $d_\beta \leq d_\alpha$), the false positive probability $p$. With these parameters, the input sharing is a $(s, \ell, N)$-packed Shamir’s secret sharing and the MPC protocol is required to be $\ell$-private. The theorem proof is provided in Appendix 3.

Theorem 2. Let $\Pi$ be an MPC protocol of parameters $(N, \ell, s, d_\alpha, d_\beta, p)$ complying to the format of Protocol 3. In particular, $\Pi$ is $\ell$-private in the semi-honest model and of false positive probability $p$. Then, Protocol 3 built from $\Pi$ satisfies three following properties:

- Completeness. A prover $P$ who knows a witness $w$ such that $(x, w) \in R$ and who follows the steps of the protocol always succeeds in convincing the verifier $V$. 

32
1. The prover samples a root seed $\text{seed} \in (0,1)^k$, compute the plain hint $\beta^1 = \psi(w; r^1)$ with $r^1 \leftarrow \text{PRG}$(seed), and computes:

$$([w], [\langle \beta^1 \rangle], [\langle \beta_1^2 \rangle], \ldots, [\langle \beta_{s_1}^t \rangle], h_1) \leftarrow \text{GenerateAndCommitShares}(w, \beta^1, \text{seed})$$

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 $\Gamma$ from $\mathbb{F}_{q^k}^{\ell \times s_1}$ and sends it to the prover;
(b) The prover computes $P_t := \Gamma \cdot Q^t + P_a$ where $Q$ is computed from the polynomials of the sharings $([w], [\beta^1], [\beta_2^1], \ldots, [\beta_{s_1}^t])$ using $\mathbb{P}$ and $P_a$ is the polynomial of $[u]$. The prover sends $h'_1 := \text{Hash}(P_t)$ to the verifier.

2. The verifier samples at random a challenge $\xi$ and sends it to the prover;
3. The prover runs the MPC computation in their head. Specifically, the prover computes the shares

$$[\alpha^1], = \Phi^1([w], [\beta^1]) \forall i \in [1 : d_{s_1} + 1].$$

4. For $j = 2$ to $t$:
(a) The prover computes the plain hint $\beta^j = \psi(w; \xi^1, \ldots, \xi^{j-1}; r^j)$ with $r^j \leftarrow \text{PRG}$(seed) and deduce the hint correction $\Delta \beta^j = \beta^j - \beta^1$ and the hint sharing $[\beta^j] = [\beta^j_1] = \frac{1}{|J_\alpha|}[\beta^1_1]$ following Equation (11). The prover then computes the hash $h_j = \text{Hash}([\alpha^1], [\beta^j_1], \Delta \beta^j)$ and sends it to the verifier.
(b) The verifier samples at random a challenge $\xi$ and sends it to the prover;
(c) The prover runs the MPC computation in their head. Specifically, the prover computes the shares

$$[\alpha^j], = \Phi^j([w], [\beta^j]) \forall i \in [1 : d_{s_1} + 1].$$

5. The prover computes $h_{j+1} = \text{Hash}([\alpha^1], [\beta^j_1], \Delta \beta^j)$ and sends it to the verifier.

6. The verifier samples at random a subset $I \subseteq [1 : N]$ of $\ell$ 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

$$\text{views}_I, \text{decom}_I \leftarrow \text{OpenShares}(w, \text{seed}, \beta^1, [\Delta \beta^j], I).$$

The prover sends $\text{views}_I, \text{decom}_I \leftarrow \text{OpenShares}(w, \text{seed}, \beta^1, [\Delta \beta^j], I)$ to the verifier, where $S^I \subseteq [1 : N]$ is of cardinality $|S^I| = d_{s_1} + 1 - \ell$ and such that $S^I \cap I = \emptyset$, the prover computes $[\alpha^1], \alpha^1 \in [\beta^1, \beta^1_{s_1}]$.

In the TCitH-MT variant, the prover further sends $\{P_i(c_j)\}_{j \in S}$ for a set $S \subseteq [1 : N]$ of cardinality $|S| = d_3 + 1 - \ell$ 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:

$$h_1 \leftarrow \text{VerifyDecommitment}(\text{views}_I, \text{decom}_I, I).$$

If $h_1 \neq h_1$, the verifier stops and outputs REJECT.
- (Parties’ computation) Then, the verifier computes

$$([w], [\langle \beta^1 \rangle], [\langle \beta^j \rangle]) \leftarrow \text{RetrieveShares}(\text{views}_I, I)$$

and

$$\forall j \in [1 : t], \{\Phi^j([w], [\beta^j])\} \forall i \in I \forall j \in [1 : t].$$

For all $j \in [1 : t]$, the verifier recovers the shares $[\alpha^j], \alpha^j \in [\beta^1, \ldots, \beta_{s_1}^t]$ and checks that $h_{j+1} = \text{Hash}([\alpha^j], [\beta^j_{s_1}], \Delta \beta^j)$ (if $j < t$) or $h_{j+1} = \text{Hash}([\alpha^j], [\beta^j_{s_1}], \Delta \beta^j)$ (if $j = t$). If the check fails, the verifier stops and outputs REJECT.

In the TCitH-MT variant, the verifier further computes $P_i(c_j) = \Gamma \cdot Q^t + P_a$ for all $i \in I$ from opened shares. From $\{P_i(c_j)\}_{i \in S, j}$ the verifier reconstructs $P_t$ by interpolation and check that $h'_1 = \text{Hash}(P_t)$. If the check fails, the verifier stops and outputs REJECT.
- (Protocol outcome) The verifier recovers the plain broadcast value $\alpha$ from $\{\alpha\}_{I \cap S}$ 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.

Fig. 3: Sharing and commitment routines, where the polynomial $F(X)$ is defined as $\prod_{k=1}^{s}(X - \omega_k)$ and $I_0$ is the polynomial defined by interpolation such that $\forall i \in [1 : s], I_0(\omega_i) = v_i$. 

<table>
<thead>
<tr>
<th><strong>TCitH-GGM</strong></th>
<th><strong>TCitH-MT</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>GenerateAndCommitShares$(w, \beta^3, \text{seed})$:</td>
<td></td>
</tr>
<tr>
<td>$(\text{seed})_{t \in SP} \leftarrow \text{TreePRG(\text{seed})}$</td>
<td></td>
</tr>
<tr>
<td>For all $T$, $s_f \leftarrow (s_{I_f}, s_{I_f, T}) \leftarrow \text{PRG(\text{seed}, 0)}$</td>
<td></td>
</tr>
<tr>
<td>$\Delta w \leftarrow w - \sum_T s_{I_f}$</td>
<td></td>
</tr>
<tr>
<td>$\Delta \beta^3 \leftarrow \beta^3 - \sum_T s_{I_f, T}$</td>
<td></td>
</tr>
<tr>
<td>$[w]<em>j \leftarrow \text{GenerateSharing}(\Delta w, (s</em>{I_f})_T, t)$</td>
<td></td>
</tr>
<tr>
<td>$[\beta^3]<em>j \leftarrow \text{GenerateSharing}(\Delta \beta^3, (s</em>{I_f, T})_T, dhn)$</td>
<td></td>
</tr>
<tr>
<td>For all $j \in [2 : t]$, $s_{f,j} \leftarrow (s_{I_f}, s_{I_f, T}) \leftarrow \text{PRG(\text{seed}, j)}$</td>
<td></td>
</tr>
<tr>
<td>$[\beta^3]<em>j \leftarrow \text{GenerateSharing}(0, (s</em>{I_f, T})_T, dhn)$</td>
<td></td>
</tr>
<tr>
<td>For all $T \in SP^T$, $\text{comy} \leftarrow \text{Com(seed}_j, \rho_T)$</td>
<td></td>
</tr>
<tr>
<td>Else $\text{comy} \leftarrow \text{Com(seed}_j, \Delta w, \Delta \beta^3, \rho_T)$</td>
<td></td>
</tr>
<tr>
<td>$h_1 \leftarrow \text{Hash(comy}_T)$</td>
<td></td>
</tr>
<tr>
<td>Return $([w], [\beta^3], [\beta^3], \ldots, [\beta^3], h_1)$</td>
<td></td>
</tr>
<tr>
<td>OpenShares$(w, \text{seed}, \beta^3, {\Delta \beta^3}_{j=2}^{t}, I)$:</td>
<td></td>
</tr>
<tr>
<td>$(w, \text{seed}, \beta^3) \leftarrow (\Delta w, \Delta \beta^3, \text{comy})$</td>
<td></td>
</tr>
<tr>
<td>path, $\leftarrow \text{GetSiblingPath(\text{seed})}$</td>
<td></td>
</tr>
<tr>
<td>views, $\leftarrow$ (path, $\Delta w, \Delta \beta^3, \ldots, \Delta \beta^3$)</td>
<td></td>
</tr>
<tr>
<td>decom, $\leftarrow \text{comy}$</td>
<td></td>
</tr>
<tr>
<td>Return (views, decom)</td>
<td></td>
</tr>
<tr>
<td>VerifyDecommitment$(\text{views}, \text{decom}, I)$:</td>
<td></td>
</tr>
<tr>
<td>$(\text{path}_T, \Delta w, \Delta \beta^3, \ldots, \Delta \beta^3) \leftarrow \text{views}$</td>
<td></td>
</tr>
<tr>
<td>$(\text{seed})_{t \in SP} \leftarrow \text{RetrieveLeavesFromPath(path}_T)$</td>
<td></td>
</tr>
<tr>
<td>$\text{comy} \leftarrow \text{decom}$</td>
<td></td>
</tr>
<tr>
<td>For all $T \neq I_0$, $\text{comy} \leftarrow \text{Com(seed}_j, \rho_T)$</td>
<td></td>
</tr>
<tr>
<td>Else $\text{comy} \leftarrow \text{Com(seed}_j, \Delta w, \Delta \beta^3, \rho_T)$</td>
<td></td>
</tr>
<tr>
<td>$h_1 \leftarrow \text{Hash(comy}_T)$</td>
<td></td>
</tr>
<tr>
<td>Return $h_1$</td>
<td></td>
</tr>
<tr>
<td>RetrieveShares$(\text{views}, I)$:</td>
<td></td>
</tr>
<tr>
<td>$(\text{path}, \Delta w, \Delta \beta^3) \leftarrow \text{views}$</td>
<td></td>
</tr>
<tr>
<td>$(\text{seed})_{t \in SP} \leftarrow \text{RetrieveLeavesFromPath(path}_T)$</td>
<td></td>
</tr>
<tr>
<td>For all $T \neq I_0$, $s_f \leftarrow \text{PRG(\text{seed}, j)}$ for $j \in [0, \ldots, t]$</td>
<td></td>
</tr>
<tr>
<td>For all $i \in I$, $[w]<em>i \leftarrow \text{GeneratePartyShare}((s</em>{f,i})_T, \Delta w, t)$</td>
<td></td>
</tr>
<tr>
<td>For all $j \in [1 : t]$, $[\beta^3]<em>j \leftarrow \text{GeneratePartyShare}((s</em>{f,j})_T, \Delta \beta^3, dhn)$</td>
<td></td>
</tr>
<tr>
<td>Return $([w], [\beta^3], [\beta^3], \ldots, [\beta^3], \beta^3)$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>TCitH-GGM</strong></th>
<th><strong>TCitH-MT</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>GenerateAndCommitShares$(w, \beta^3, \text{seed})$:</td>
<td></td>
</tr>
<tr>
<td>$(\text{seed})_{t \in SP} \leftarrow \text{PRG(\text{seed}, 0)}$</td>
<td></td>
</tr>
<tr>
<td>$u_1, (\text{seed})_{t \in SP} \leftarrow \text{PRG(\text{seed}, -1)}$</td>
<td></td>
</tr>
<tr>
<td>For all $i \in [1 : N]$, $P_{\beta}(X) = I_{\beta}(X) + F(X) \cdot \sum_{k=1}^{s} \omega_k \cdot \sum_{i=0}^{t} r_i^k \cdot X^k$</td>
<td></td>
</tr>
<tr>
<td>$P_{\beta^3}(X) = I_{\beta}(X) + F(X) \cdot \sum_{k=1}^{s} \omega_k \cdot \sum_{i=0}^{t} r_i^k \cdot X^k$</td>
<td></td>
</tr>
<tr>
<td>For all $j \in [2 : t]$, $\beta^3, (r_j)_{k \in \omega_k-1} \leftarrow \text{PRG(\text{seed}, j)}$</td>
<td></td>
</tr>
<tr>
<td>For all $i \in [1 : N]$, $P_{\beta}(X) = I_{\beta} + F(X) \cdot \sum_{k=1}^{s} \omega_k \cdot \sum_{i=0}^{t} r_i^k \cdot X^k$</td>
<td></td>
</tr>
<tr>
<td>For all $i \in [1 : N]$, $[w]<em>i \leftarrow P</em>{\beta}(\omega_i)$</td>
<td></td>
</tr>
<tr>
<td>$[w]<em>i \leftarrow P</em>{\beta^3}(\omega_i)$ for all $j \geq 2$</td>
<td></td>
</tr>
<tr>
<td>$\text{com}_i \leftarrow \text{Com}([w], | [w], | [\beta^3], | [\beta^3], \ldots, [\beta^3], \rho_i)$</td>
<td></td>
</tr>
<tr>
<td>$h_1 \leftarrow \text{MerkleRoot}((\text{com}_i), I)$</td>
<td></td>
</tr>
<tr>
<td>Return $([w], [\beta^3], [\beta^3], \ldots, [\beta^3], h_1)$</td>
<td></td>
</tr>
<tr>
<td>OpenShares$(w, \text{seed}, \beta^3, {\Delta \beta^3}_{j=2}^{t}, I)$:</td>
<td></td>
</tr>
<tr>
<td>$(w, \text{seed}, \beta^3) \leftarrow (\Delta w, \Delta \beta^3, \text{com}_i)$</td>
<td></td>
</tr>
<tr>
<td>path, $\leftarrow \text{GetSiblingPath(\text{seed})}$</td>
<td></td>
</tr>
<tr>
<td>views, $\leftarrow$ (path, $\Delta w, \Delta \beta^3, \ldots, \Delta \beta^3$)</td>
<td></td>
</tr>
<tr>
<td>decom, $\leftarrow \text{com}_i$</td>
<td></td>
</tr>
<tr>
<td>Return (views, decom)</td>
<td></td>
</tr>
<tr>
<td>VerifyDecommitment$(\text{views}, \text{decom}, I)$:</td>
<td></td>
</tr>
<tr>
<td>$(w, \beta^3, \ldots, \beta^3) \leftarrow \text{views}$</td>
<td></td>
</tr>
<tr>
<td>For all $i \in I$, $\text{com}_i \leftarrow \text{Com}([w], | [\beta^3], | [\beta^3], \ldots, [\beta^3], \rho_i)$</td>
<td></td>
</tr>
<tr>
<td>$h_1 \leftarrow \text{RetrieveRootFromPath}((\text{com}_i), I)$</td>
<td></td>
</tr>
<tr>
<td>Return $h_1$</td>
<td></td>
</tr>
<tr>
<td>RetrieveShares$(\text{views}, I)$:</td>
<td></td>
</tr>
<tr>
<td>$(w, \beta^3, \ldots, \beta^3) \leftarrow \text{views}$</td>
<td></td>
</tr>
<tr>
<td>$(\text{seed})_{t \in SP} \leftarrow \text{shares}_i$</td>
<td></td>
</tr>
<tr>
<td>For all $i \in I$, $\text{shares}_i \leftarrow \text{Com}([w], | [\beta^3], | [\beta^3], \ldots, [\beta^3], \rho_i)$</td>
<td></td>
</tr>
<tr>
<td>$h_1 \leftarrow \text{RetrieveRootFromPath}((\text{shares}_i), I)$</td>
<td></td>
</tr>
<tr>
<td>Return $([w], [\beta^3], [\beta^3], \ldots, [\beta^3])$</td>
<td></td>
</tr>
</tbody>
</table>
- **Soundness.** Suppose that there is an efficient prover \( \hat{P} \) that, on input \( x \), convinces the honest verifier \( V \) to accept with probability
\[
\hat{\epsilon} := \Pr[(\hat{P}, V)(x) \rightarrow 1] > \epsilon
\]
where the soundness error \( \epsilon \) is defined as
\[
\epsilon := \begin{cases} 
p + (1 - p) \cdot \left( \frac{\Delta_0}{\ell} \right) & \text{for the TCitH-GGM variant,} 
p + (1 - p) \cdot \left( \frac{\Delta_0}{\ell} \right) + \left( \frac{\Delta_{N+2}}{|F|_N} \right) & \text{for the TCitH-MT variant.} \end{cases}
\]

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

- **Honest-Verifier Zero-Knowledge.** Let the pseudorandom generator PRG used in Protocol 2 be \( (t, \epsilon_{\text{PRG}}) \)-secure and the commitment scheme \( \text{Com} \) be \( (t, \epsilon_{\text{Com}}) \)-hiding. There exists an efficient simulator \( S \) which, given random challenge \( I \) outputs a transcript which is \( (t, \epsilon_{\text{PRG}} + \epsilon_{\text{Com}}) \)-indistinguishable from a real transcript of Protocol 2.

### Dealing with small fields.
One of the drawbacks of using the (packed) Shamir’s secret sharing scheme is that the number \( N \) of parties should be smaller than the field size \( F \). It can be an issue when working with statements based on a small field \( F_0 \) as the binary field. To handle that, we should share the secret in an larger field extension \( F \) and lift all the multiparty computation in this larger field. The concrete impact on the proof size depends on the considered variant.

With TCitH-MT, each revealed share shall weight more in communication while living in the extension field \( ([F : F_0] \text{ times more expensive, where } [F : F_0] \text{ is the degree of the extension}) \). Moreover, the MPC protocol should include a test to check that the shared values live in the base field \( F_0 \) (for example, by checking that a random \( F_0 \)-linear combination of the shared values is in \( F_0 \)).

With TCitH-GGM, the situation is different. As in Section 3.2, we tweak a bit the sharing generation of the witness \( w \) so that the expanding randomness \( \{s_T\} \) and the auxiliary value lie in the small field. Specifically, after expanding all \( s_T \in F_0^{[w]} \) for all \( T \in S^N \), we compute the auxiliary value \( \Delta w \in F_0^{[w]} \) as
\[
\Delta w := w - \sum_{T \in S^N} s_T \in F_0^{[w]},
\]
but we compute the shares \( [w]_i \) using Equation (4) with parties’ evaluation points \( \{e_j\} \) living in the field extension \( F \) (instead of \( F_0 \)). As consequence, the shares \( \{[w]_i\}_i \) live in \( F^{[w]} \) instead of \( F_0^{[w]} \). Since the proof transcript contains the auxiliary value \( \Delta w \) of \( w \), but no shares of \( [w] \), the communication cost induced by revealing the opened shares is unchanged. The same observation holds for the hint sharings \( [\beta^1], \ldots, [\beta^j] \). However, the lifting impacts the communication cost due to the broadcasted sharings \( [\alpha^1], \ldots, [\alpha^j] \).

### 6 Application of the Extended TCitH Framework

We present hereafter several applications of the TCitH framework. Before describing these applications, we start by introducing a useful building block which is a protocol to generate high-degree random, partially structured, sharings (Section 6.1). We introduce two general zero-knowledge proof systems obtained by applying the TCitH framework to simple MPC protocols: the first one, TCitH-\( \Pi_{\text{pc}} \), checks polynomial constraints on the witness in the non-packed setting (Section 6.2) while the second one, TCitH-\( \Pi_{\text{lppc}} \), checks global linear constraints and parallel polynomial constraints on the witness in the packed setting (Section 6.3). We show how these proof systems results in competitive zero-knowledge arguments for (low-degree) arithmetic circuits (Section 6.4). We further described improved post-quantum (ring) signatures from TCitH-\( \Pi_{\text{pc}} \) (Sections 6.5 and 6.6) as well as zero-knowledge arguments for lattices from TCitH-\( \Pi_{\text{lppc}} \) (Section 6.7).
6.1 Generation of High-Degree Sharings

We describe hereafter a protocol to generate a uniformly-random degree-$d'$ sharing $[v]$ of a random tuple $v \in \mathbb{F}_s$, where $d'$ is strictly larger than $d = s + \ell - 1$. For the TCitH-GGM variant, such a sharing $[v]$ can be directly obtained from the hint oracle $O_H$. We note that we could do the same with the TCitH-MT variant but this implies a penalty in communication (or argument size). Indeed, a degree-$d'$ hint implies $d_\beta = d'$ which increases the communication of the degree-enforcing commitment (compared to $d_\beta = d$ in the absence of high-degree hints). For this reason, we introduce the protocol which increases the communication of the degree-enforcing commitment (compared to $d_\beta = d$).

Generation of a Random Sharing $(\Pi_8)$. The idea behind our approach is to define the polynomial $P_v$ underlying the random sharing $[v]$ from lower degree polynomials $P_{u_1}, \ldots, P_{u_n}$ as follows:

$$P_v(X) = \sum_{j=1}^{n} P_{u_j}(X) \cdot X^{(j-1)\delta},$$

for some $n, \delta \in \mathbb{N}$. Here, using polynomials $P_{u_1}, \ldots, P_{u_{n-1}}$ of degree $d_\beta$ and $P_{u_n}$ of degree $d' - (n-1)\delta \leq d_\beta$ we get that $P_v$ is of degree $d'$. Based on this definition of $P_v$, $\Pi_8$ is formally defined in Protocol 6.

1. The parties get $n - 1$ random sharings $[u_1], \ldots, [u_{n-1}]$ of degree $d_\beta$ from the hint oracle $O_H$.
2. The parties get a random sharing $[u_n]$ of degree $d' - (n-1)\delta$ from the hint oracle $O_H$.
3. The parties locally computes

$$[v]_1 = \sum_{j=1}^{n} [u_j]_1 \cdot (e_j)^{(j-1)\delta}.$$

4. The parties now hold a random degree-$d'$ sharing $[v]$.

Protocol 6: $\Pi_8$ – Generation of a random degree-$d'$ sharing.

**Lemma 1.** In Protocol 6 let $n := \lceil (d' + 1 - \ell)/(d_\beta - \ell + 1) \rceil$ and $\delta := d_\beta - \ell + 1$. Then, given the views of $\ell$ parties of indices $i \in I$ for some set $I \subseteq [N]$ s.t. $|I| = \ell$, the sharing $[v]$ is uniformly random of degree $d'$ conditioned to $[v]_I$, being consistent with the views. Namely, for any set $J$ of cardinality $|J| \leq d' + 1 - \ell$ s.t. $I \cap J = \emptyset$, $[v]^J$ is a uniform random tuple of $\mathbb{F}^{|J|}$ mutually independent of the views.

**Proof.** Let us take a set $I \subseteq [N]$ s.t. $|I| = \ell$. For all $j \in [1 : n]$, the polynomial $P_{u_j}$ can be written as

$$P_{u_j}(X) = I_{u_j,I}(X) + \prod_{i \in I} (X - e_i) \cdot R_{u_j,I}(X),$$

where

- $I_{u_j,I}$ is defined by interpolation such that $I_{u_j,I}(e_i) = [u_j]_i$ for all $i \in I$, and
- the distribution of $R_{u_j,I}$ knowing $[u_j]_I$ is the uniform law distribution over polynomials of degree at most $d_\beta - \ell$ (for $R_{u_n,I}$, of degree at most $d' - (n-1)\delta$ since $P_{u_n}$ is a uniformly-random polynomial satisfying $P_{u_n}(e_i) = [u_n]_i$ for all $i \in I$.

36
Specifically, the constant sharing $d$ from the polynomials of degree at most $d$.

We have

$$P_v(X) = \sum_{j=1}^{n} \left[ I_{u_j},I(X) + \prod_{i \in I} (X - e_i) \cdot R_{u_j},I(X) \right] \cdot X^{(j-1)\delta}$$

$$= \left[ \sum_{j=1}^{n} I_{u_j},I(X) \cdot X^{(j-1)\delta} \right] + \prod_{i \in I} (X - e_i) \cdot \left[ \sum_{j=1}^{n} R_{u_j},I(X) \cdot X^{(j-1)\delta} \right]$$

By defining $I(X) := \sum_{j=1}^{n} I_{u_j},I(X) \cdot X^{(j-1)\delta}$ and $R(X) := \sum_{i=1}^{n} R_{u_j},I(X) \cdot X^{(j-1)\delta}$, we have that $P_v(X) = I(X) + \prod_{i \in I} (X - e_i) \cdot R(X)$. The distribution of the polynomial $R$ knowing $\{[u_j]\}_{I}$ is a uniform distribution from the polynomials of degree at most $d' - \ell$, when $\delta := \deg R_{u_j} + 1 = \beta \ell + 1$. So, $P_v(x)$ is a random polynomial of degree at most $d'$ such that $P_v(e_i) = [v]$, for all $i \in I$.

For a target degree $d'$ and a target maximal degree $\beta$ for the hints, Lemma\[1\] gives the parameters $n$ and $\delta$ of the generation protocol $\Pi_8$. For the TCitH-MT variant, the parameter $d'$ can then be chosen to minimize the communication by balancing its impact on the degree enforcement parameter $\eta$ and the number $n$ of hints $[u_i]$ (both impacting the proof size).

**Generation of Sharings of Zero ($\Pi_8$).** Let us now consider the case of the generation of a uniformly-random degree-$d'$ sharing of $(0, \ldots, 0) \in \mathbb{F}^s$. This can be achieved by using the protocol $\Pi_8$ to generate a uniformly-random degree-$(d' - s)$ sharing which is then locally multiplied by the constant sharing $[0]$ corresponding to the degree-$s$ polynomial $P_0(X) = \prod_{i=1}^{s} (X - \omega_i)$ (where we recall that the $\omega_i$'s are the evaluation points for the plain coordinates). Formally, the protocol $\Pi_8$ for the generation of a uniformly-random degree-$d'$ sharing of $(0, \ldots, 0) \in \mathbb{F}^s$ performs the following steps:

1. The parties run protocol $\Pi_8$ to obtain a uniformly-random degree-$(d' - s)$ sharing $[v]$,.
2. The parties locally compute $[z]_i = [0]_i \cdot [v]_i$.
3. The parties now hold a random degree-$d'$ sharing $[z]$ of $(0, \ldots, 0) \in \mathbb{F}^s$.

In terms of privacy, Lemma\[1\] implies that given the views of any $\ell$ parties of indices in $J$, the generated sharing $[z]$ is uniformly-random conditioned to $(P_z(\omega_1), \ldots, P_z(\omega_s)) = (0, \ldots, 0)$ and to $[z]_J$ consistent with the views.

**Generation of Sharings Summing to Zero ($\Pi_\Sigma_0$).** We finally consider the case of the generation of a uniformly-random degree-$d'$ sharing $[v]$ of a uniformly-random $v = (v_1, \ldots, v_s) \in \mathbb{F}^s$ such that $\sum_{i=1}^{s} v_i = 0$. This can be achieved using a tweak version of protocol $\Pi_8$. In this tweaked version, the sharings $[u_1], \ldots, [u_n]$ returned by the hint oracle $O_H$ are not fully random: the constant term of the polynomial $P_{u_i}$ is defined w.r.t. the random polynomials $\{P_{u_i} \mid i \geq 2\}$ such that $P_v(\omega_1) + \cdots + P_v(\omega_s) = 0$. In practice, this tweak simply involves one hint correction $\Delta u_1 \in \mathbb{F}$ to correct one evaluation, say $P_{u_1}(\omega_1)$\[10\] This tweak of the protocol $\Pi_8$ is referred to as the protocol $\Pi_\Sigma_0$ in the following. In terms of privacy, Lemma\[1\] implies that given the views of any $\ell$ parties of indices in $J$, the generated sharing $[v]$ is uniformly-random conditioned to $[v]_J$ consistent with the views and to $\sum_{i=1}^{s} v_i = 0$.

### 6.2 Proof System for Polynomial Constraints

We instantiate the TCitH framework with an MPC protocol verifying for general polynomial constraints without using packed sharing, i.e. with $s = 1$. 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).

\[10\] Specifically, the constant sharing $[\Delta u_1]$ added to $[u_1]$ for correction corresponds to the polynomial $P_{\Delta u_1}$ such that $P_{\Delta u_1}(\omega_1) = \Delta u_1$ and $P_{\Delta u_1}(\omega_i) = 0$ for all $i \in [2 : s]$. 

37
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 \( d \). The protocol \( \Pi_{\text{PC}} \) checking such polynomial constraints is formally described in Protocol 7.

1. The parties receive a sharing \([w]\) with \( s = 1 \) and \( \deg[w] = \ell \).
2. The parties get a uniformly-random degree-(\( d\ell \)) sharing \([v]\) of \( v = 0 \in \mathbb{F} \) using \( \Pi_0 \) (Section 6.1).
3. The parties receive random values \( \gamma_1, \ldots, \gamma_m \in \mathbb{F} \) from \( \mathcal{O}_R \).
4. The parties locally compute
   \[
   \alpha = [v] + \sum_{j=1}^{m} \gamma_j \cdot f_j([w]) .
   \]
5. The parties open \([\alpha]\) to publicly recompute \( \alpha \).
6. The parties output \( \text{Accept} \) if and only if \( \alpha = 0 \).

Protocol 7: \( \Pi_{\text{PC}} \) – Verification of polynomial constraints.

**Lemma 2.** The protocol \( \Pi_{\text{PC}} \) is correct, namely it always outputs \( \text{Accept} \) when \( w \) vanishes the polynomials \( f_1, \ldots, f_m \). The protocol \( \Pi_{\text{PC}} \) is also sound with false positive probability \( \frac{1}{|\mathbb{F}|} \) and \( \ell \)-private.

**Proof.** It is easy to check that \( \Pi_{\text{PC}} \) outputs \( \text{Accept} \) when \( w \) vanishes the polynomials \( f_1, \ldots, f_m \) (and when the hints are well-formed, i.e. \( v = 0 \)) since
\[
\alpha = v + \sum_{j=1}^{m} \gamma_j \cdot f_j(w) = 0 .
\]
When \( w \) does not vanish the polynomials, there exists \( j' \) such that \( f_{j'}(w) \neq 0 \). In that case, \( \alpha \) is uniformly random in \( \mathbb{F} \) since \( \gamma_{j'} \) has been chosen uniformly at random in \( \mathbb{F} \). We get that \( \Pi_{\text{PC}} \) shall output \( \text{Accept} \) with probability \( \frac{1}{|\mathbb{F}|} \), which corresponds to its false positive probability. Finally, the \( \ell \)-privacy of the protocol holds for the following reason. In our MPC model, the protocol is \( \ell \)-private as long as the broadcast sharings do not leak information on the witness. In the present case, we have that \([v]\) is a uniformly-random degree-(\( d\ell \)) sharing of 0, which implies that \([\alpha]\) is a uniformly-random degree-(\( d\ell \)) sharing of \( \sum_{j=1}^{m} \gamma_j \cdot f_j(w) = 0 \). The protocol \( \Pi_{\text{PC}} \) is thus \( \ell \)-private. \( \square \)

**Parallel repetitions.** The false positive probability can be made arbitrarily small by performing several repetitions of \( \Pi_{\text{PC}} \) in parallel. The obtained protocol \( \Pi_{\text{PC}}^{(\rho)} \) is similar to \( \Pi_{\text{PC}} \) with the following differences:
- \( \rho \) sharings \([v_1], \ldots, [v_\rho]\) are generated using \( \Pi_0 \) in Step 2,
- \( \rho \) batches of \( m \) random values \( (\gamma_{1,i}), \ldots, (\gamma_{\rho,i}) \) are requested from \( \mathcal{O}_R \) in Step 3,
- \( \rho \) sharings \([\alpha_1], \ldots, [\alpha_\rho]\) are computed and broadcasted in Step 4, s.t., \([\alpha_k] = [v_k] + \sum_{j=1}^{m} \gamma_{k,j} \cdot f_j([w])\),
- the parties output \( \text{Accept} \) if the \( \rho \) recomputed values verify \( \alpha_1 = \cdots = \alpha_\rho = 0 \).

A simple adaptation of the proof of Lemma 2 implies that \( \Pi_{\text{PC}}^{(\rho)} \) is sound with false positive probability \( \frac{1}{|\mathbb{F}|^\rho} \).

**The TCitH-\( \Pi_{\text{PC}} \) proof system.** By Theorem 2 when applying the TCitH framework to the protocol \( \Pi_{\text{PC}}^{(\rho)} \) we obtain a complete, sound, and HVZK proof system of soundness error
\[
\epsilon := \left\{ \begin{array}{ll}
\frac{1}{|\mathbb{F}|^\rho} + \left( 1 - \frac{1}{|\mathbb{F}|}\right) \cdot \left( \frac{d\ell}{N}\right) \frac{1}{|\mathbb{F}|^\rho} + \left( 1 - \frac{1}{|\mathbb{F}|}\right) \cdot \left( \frac{d\ell}{N}\right) \frac{d s + 2}{|\mathbb{F}|^\rho} & \text{for the TCitH-GGM variant}, \\
\frac{1}{|\mathbb{F}|^\rho} + \left( 1 - \frac{1}{|\mathbb{F}|}\right) \cdot \left( \frac{d\ell}{N}\right) \frac{1}{|\mathbb{F}|^\rho} + \left( 1 - \frac{1}{|\mathbb{F}|}\right) \cdot \left( \frac{d\ell}{N}\right) \frac{d s + 2}{|\mathbb{F}|^\rho} & \text{for the TCitH-MT variant}. 
\end{array} \right.
\]

38
To obtain a zero-knowledge non-interactive argument of knowledge, we perform $\tau$ parallel repetitions of the protocol and apply the Fiat-Shamir transform. Since the proof system has 5 rounds (with TCitH-GGM) or 7 rounds (with TCitH-MT) and uses parallel repetitions, one must be careful while selecting the parameters to avoid potential KZ-like forgery attacks \cite{KZ20a}. To achieve $\lambda$ bits of security in the non-interactive setting, we propose to select

- the parameter $\eta$ of the degree-enforcing commitment in TCitH-MT such that $(d_\beta + 2)/|F|$\footnote{\cite{KZ20a}} \leq 2^{-\lambda};
- the number $\rho$ of MPC repetitions in the protocol $\Pi_{\text{MC}}^{(\rho)}$ such that $1/|F|\rho \leq 2^{-\lambda}$;
- the number $\tau$ of PoK repetitions such that $\left(\frac{d_\beta + 2}{|F|}\right)^\tau \leq 2^{-\lambda}$.

Here, $N$, $\ell$ and $d_\beta$ are flexible parameters which can be chosen to optimize the proof size. The proof transcript includes:

- The opened shares $\|w\|_l$ of the witness $w \in F^{|w|}$, for each of the $\tau$ PoK repetitions.
- The opened shares of the hints used to build the sharing $\|v\|$, for each of the $\rho$ MPC repetitions of the $\tau$ PoK repetitions. With the variant TCitH-GGM, these shares are communication-free.
- The degree-$(d\ell)$ sharing $\|a\|$, for each of the $\rho$ MPC repetitions of the $\tau$ PoK repetitions. Since $\|a\|_l$ can be recomputed by the verifier and since the $a$ should be zero, the prover just needs to send $(d\ell + 1) - \ell - 1 = (d - 1)\ell$ shares.
- The sibling paths in the GGM tree in the variant TCitH-GGM together with the unopened seed commitments, or the authentication paths in the variant TCitH-MT.
- In TCitH-MT, the communication cost due to the sharing degree enforcing, which consists in the opened shares $\|u\|_l$ used to mask the output of the sharing-degree test (where $u \in F^{n^\ast}$) together with $d_\beta + 1 - \ell$ additional shares of $\|\xi\|$ (where $\xi \in F^{n^\ast}$), for each of the $\tau$ PoK repetitions.

We obtain the following proof size when applying TCitH-GGM:

$$\text{SIZE}_{\text{GGM}} = 4\lambda + \tau \cdot \left(\frac{|w| \cdot \log_2 |F|}{\Delta w} + \frac{(d - 1)\ell \cdot \rho \cdot \log_2 |F|}{|a|} + \frac{\lambda \cdot \log_2 N}{\ell} + 2\lambda\right)$$

and when applying TCitH-MT:

$$\text{SIZE}_{\text{MT}} = 6\lambda + \tau \cdot \left(\frac{\ell \cdot (|w| + n_{\text{hints}} \cdot \rho) \cdot \log_2 |F|}{|w|_l,|v|_l} + \frac{(d - 1)\ell \cdot \rho \cdot \log_2 |F|}{|a|} + \frac{2\lambda \cdot \ell \cdot \log_2 N}{\tau} + \frac{(d_\beta + 1) \cdot \eta \cdot \log_2 |F|}{\text{degree enforcing}}\right)$$

where $n_{\text{hints}} = \left\lceil (d - 1)\ell / (d_\beta - \ell + 1) \right\rceil$.

### 6.3 Proof System for Linear and Parallel Polynomial Constraints

We now introduce a general MPC protocol to efficiently instantiate the TCitH framework with packed secret sharing ($s > 1$). We consider a witness $w = (w_1, \ldots, w_{ns}) \in F^{ns}$ arranged in $n$ packs:

$$w = \begin{pmatrix} w_1, & \ldots, & w_s, \\ \vdots \end{pmatrix} \quad \text{s.t. } \|w\| = \begin{pmatrix} \|w_1, \ldots, w_s\| \\ \vdots \end{pmatrix}$$
following the formalism of Section 2.1. We shall further denote \(w^{(1)}, \ldots, w^{(s)}\) the “column tuples” of the arranged witness, namely \(w^{(k)} = (w_k, w_{s+k}, \ldots, w_{(n-1)s+k})\) for every \(k \in [1 : s]\).

The MPC protocol we introduce here checks polynomial constraints in parallel on each pack slot as well as global linear constraints. With \(n_1\) the number of (parallel) polynomial constraints and \(n_2\) the number of (global) linear constraints, the protocol \(\Pi_{\text{LPPC}}\) checks that \(w\) verifies:

1. (parallel polynomial constraints) the \(w^{(k)}\)'s satisfy some polynomial relations:
   \[
   \forall j \in [1 : n_1], \forall k \in [1 : s], \ f_j(w^{(k)}) = 0
   \]
   where \(f_1, \ldots, f_{n_1}\) are polynomials from \(F[X_1, \ldots, X_n]\) of total degree at most \(d\);

2. (global linear constraints) \(w\) satisfies a linear relation \(A \cdot w = t\) where \(A\) is a matrix from \(F^{n_2 \times (ns)}\) and \(t\) is a vector from \(F^{n_2}\).

Checking the polynomial constraints works as previously: the \(f_j\)'s are locally applied to the shares and the protocol checks that the \(f_j([w])\) are (high degree) sharings of \((0, \ldots, 0)\) in \(F^s\) in the same way as protocol \(\Pi_{\text{FC}}\). To check the global linear constraints, we use a similar approach as Ligero [AHIV17]. Each row \(A_j = (A_{j,1}, \ldots, A_{j,n_s})\) of the matrix gives rise to a constant sharing \([A_j]\) defined as:

\[
[A_j] = \left(\begin{array}{c}
([A_{j,1}], \ldots, A_{j,s}) \\
([A_{j,s+1}], \ldots, A_{j,2s}) \\
\vdots
\end{array}\right)
\]

where each packed sharing \([([A_{j,1}], \ldots, A_{j,s}]), \ldots\) is a constant degree-\((s-1)\) sharing (a sharing interpolated from the \(s\) matrix coefficients and without randomness). Let \([b_j] := ([A_j], [w])\), we have that the \(j\)th linear constraint \((A_j, w) = t_j\) is satisfied if and only if \([b_j]\) shares a tuple \(b_j = (b_{j,1}, \ldots, b_{j,s}) \in F^s\) satisfying

\[
\sum_{i=1}^s b_{j,i} = t_j.
\]

The \(n_2\) linear constraints are batched by computing \([\alpha'] = [v'] + \sum_{j=1}^{n_2} \gamma'_{j}[b_j]\), for \([v']\) a random degree-\((2s + \ell - 2)\) sharing of a random tuple \(v' = (v'_1, \ldots, v'_{ns}) \in F^s\) satisfying \(\sum_{i=1}^{ns} v'_i = 0\). We then have \(A \cdot w = t\) if \(\|\alpha'\|\) is such that \(\sum_{i=1}^s \alpha'_i = \sum_{j=1}^{n_2} \gamma'_j t_j\) with high probability (specifically with false positive probability \(1/|F|\)). The obtained protocol \(\Pi_{\text{LPPC}}\) is formally described in Protocol 8.

1. The parties receive a sharing \([w]\), with pack size \(s\) and degree \(\ell + s - 1\).
2. The parties get a uniformly-random degree-\((d \cdot (s + \ell - 1))\) sharing \([v]\) of \(v = (0, \ldots, 0) \in F^s\) using \(\Pi_0\) (Section 6.1).
3. The parties get a uniformly-random degree-\((2s + \ell - 2)\) sharing \([v']\) of a random tuple \(v' = (v'_1, \ldots, v'_{ns}) \in F^s\) satisfying \(\sum_{i=1}^{ns} v'_i = 0\) using \(\Pi_{\text{LPC0}}\) (Section 6.1).
4. The parties receive random values \(\gamma_1, \ldots, \gamma_{n_2} \in F\) and \(\gamma'_1, \ldots, \gamma'_{n_2} \in F\) from \(O_R\).
5. The parties locally compute
   \[
   [\alpha] = [v] + \sum_{j=1}^{n_1} \gamma_j \cdot f_j([w])
   \]
   \[
   [\alpha'] = [v'] + \sum_{j=1}^{n_2} \gamma'_j \cdot [b_j] \quad \text{with} \quad [b_j] = ([A_j], [w])
   \]
6. The parties open \([\alpha]\) and \([\alpha']\) to publicly recompute \(\alpha, \alpha' \in F^s\).
7. The parties output \(\text{Accept}\) if and only if \(\alpha = (0, \ldots, 0)\) and \(\sum_{i=1}^s \alpha'_i = \sum_{j=1}^{n_2} \gamma'_j t_j\).

Protocol 8: \(\Pi_{\text{LPPC}}\) - Verification of linear & parallel polynomial constraints.

**Lemma 3.** The protocol \(\Pi_{\text{LPPC}}\) is correct, namely it always outputs \(\text{Accept}\) when \(w\) satisfies the parallel polynomial constraints and the global linear constraints described above. The protocol \(\Pi_{\text{LPPC}}\) is also sound with false positive probability \(\frac{1}{|F|}\) and \(\ell\)-private.
Proof. It is easy to check that $Π_{\text{lppc}}$ outputs ACCEPT when the witnesses satisfy the desired properties and when the hints are well-formed, since

$$\forall i \in [1 : s], \alpha_i = v_i + \sum_{j=1}^{m_1} \gamma_j f_j(w(l)) = 0 + \sum_{j=1}^{m} \gamma_i \cdot 0 = 0,$$

$$\sum_{i=1}^{s} \alpha_i' = \sum_{i=1}^{s} v_i' + \sum_{i=1}^{s} \sum_{j=1}^{m_2} \gamma_j b_{j,i} = 0 + \sum_{j=1}^{m_2} \gamma_j' \left( \sum_{i=1}^{s} b_{j,i} \right) = \sum_{j=1}^{m_2} \gamma_j' t_j.$$

On the other hand, when $w$ does not satisfy one of the constraints:

- Either there exists $(j', k') \in [1 : m_1] \times [1 : s]$ such that $f_{j'}(w(k')) \neq 0$. In that case, $\alpha_{k'}$ is uniformly random in $\mathbb{F}$ since $\alpha_{k'}$ has been chosen uniformly at random in $\mathbb{F}$.
- Or there exists $j' \in [1 : m_2]$ such that $(A_j, w) \neq t_j$. In that case, $\sum_{i=1}^{s} \alpha_i'$ is uniformly random in $\mathbb{F}$ since $\gamma_{j'}$ has been chosen uniformly at random in $\mathbb{F}$.

In both cases, we get that $Π_{\text{lppc}}$ shall output ACCEPT with probability $1/|\mathbb{F}|$, which corresponds to its false positive probability. In our MPC model, the protocol is $\ell$-private as long as the broadcast sharings do not leak information on the witness. In the present case, we have that $\mathcal{V}$ is a uniformly random $(d \cdot (s + \ell - 1))$ sharing of $(0, \ldots, 0) \in \mathbb{F}^s$, which implies that $\mathcal{O}$ is a uniformly random $(d \cdot (s + \ell - 1))$ sharing of $(\sum_{j=1}^{m} \gamma_j \cdot f_j(w(k))) \in (0, \ldots, 0)$. Moreover, we have that $\mathcal{V}'$ is a uniformly random $(2s + \ell - 2)$ sharing of $\mathcal{V}' \in \mathbb{F}^s$ satisfying $\sum_{j=1}^{s} v_j' = 0$, which implies that $\mathcal{O}'$ is a uniformly random $(2s + \ell - 2)$ sharing of $\alpha' = v' + \sum_{j=1}^{m_2} \gamma_j b_{j}$. Thanks to the randomness of $v'$, $\alpha'$ is a uniformly random tuple of $\mathbb{F}$ conditioned to $\sum_{i=1}^{s} \alpha_i' = \sum_{j=1}^{m_2} \gamma_{j'} t_j$. The protocol $Π_{\text{lppc}}$ is thus $\ell$-private.

Parallel repetitions. As for the protocol $Π_{\text{pc}}$, the protocol $Π_{\text{lppc}}$ gives rise to a parallel-repetition version $Π_{\text{lppc}}^{(\rho)}$ which lowers the false positive probability to $\frac{1}{|\mathbb{F}|^{\rho}}$. Here again, the principle is to compute and broadcast $\rho$ version of the sharings $\|\mathcal{O}\|$ and $\|\mathcal{O}'\|$ (as when the protocol $Π_{\text{lppc}}$ was executed $\rho$ times in parallel) and check that the $\rho$ versions all satisfy the final checks.

The TCitH-$Π_{\text{lppc}}$ proof system. By Theorem 2 when applying the TCitH framework to $Π_{\text{lppc}}^{(\rho)}$ we obtain a complete, sound, and HVZK proof system of soundness error

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

To obtain a zero-knowledge non-interactive argument of knowledge, we perform $\tau$ parallel repetitions of the protocol and apply the Fiat-Shamir transform. Since the proof system has 5 rounds (with TCitH-GGM) or 7 rounds (with TCitH-MT) and uses parallel repetitions, one must be careful while selecting the parameters to avoid potential KZ-like forgery attacks \cite{KZ20}. To achieve $\lambda$ bits of security in the non-interactive setting, we propose to select

- the parameter $\eta$ of the degree-enforcing commitment in TCitH-MT such that $\binom{N}{d_{\beta} + 2}/|\mathbb{F}|^{\eta} \leq 2^{-\lambda}$,
- the number $\rho$ of MPC repetitions in the protocol $Π_{\text{lppc}}^{(\rho)}$ such that $1/|\mathbb{F}|^\rho \leq 2^{-\lambda}$,
- the number $\tau$ of PoK repetitions such that $\left(\frac{(d(s + \ell - 1))}{\binom{d}{\ell}}\right)^\tau \leq 2^{-\lambda}$.

Here, $N$, $\ell$ and $d_{\beta}$ are flexible parameters which can be chosen to optimize the proof size. The proof transcript includes:

41
One could try to optimize the above protocol by batching the proof transcript, and, on the other hand, would increase the soundness error of a single repetition (see Remark 10). We can observe that \( s = 1 \) is equivalent to \( \Pi_{\text{pc}} \). Indeed, \( \Pi_{\text{LiPC}} \) with \( s = 1 \) is equivalent to \( \Pi_{\text{pc}} \). To see this, observe that the broadcast sharing of \( \alpha' \) is communication-free (since its cost per iteration was \( 2s - 2 \) field elements) and it can be removed since the global linear constraints can be handled by the polynomial constraints (which are also global when \( s = 1 \)).

Remark 9. The protocol \( \Pi_{\text{LiPC}} \) can be seen as a generalization of the protocol \( \Pi_{\text{pc}} \). Indeed, \( \Pi_{\text{LiPC}} \) with \( s = 1 \) is equivalent to \( \Pi_{\text{pc}} \). To see this, observe that the broadcast sharing of \( \alpha' \) is communication-free (since its cost per iteration was \( 2s - 2 \) field elements) and it can be removed since the global linear constraints can be handled by the polynomial constraints (which are also global when \( s = 1 \)).

Remark 10. We can observe that \( \text{Size}_{\text{GGM}} \) is linearly impacted by the size of the witness \( |w| = ns \) (to communicate the correction term \( \Delta w \)). On the other hand, \( \text{Size}_{\text{MT}} \) is only linearly impacted by \( n \) (the number of packs) and by \( s \) (the pack size). This size can hence be sublinear in the witness size by taking,
e.g., $n \approx s \approx \sqrt{|w|}$. For this reason, using packed secret sharing is particularly interesting for the Merkle tree variant. Moreover, while the GGM variant has a “smaller tree” (because a sibling path made of seeds is twice smaller than a Merkle path made of hash digests for the same number of parties $N$), the overhead of the Merkle tree is compensated by the packing while moving from “small size” witnesses to “medium size” witnesses. The two variants hence offer two interesting tradeoffs with the best choice (in terms of proof size) depending on the size of the underlying statement.

6.4 Zero-Knowledge Arguments for Arithmetic Circuits

In this section, we apply the previous zero-knowledge arguments TCitH-$\Pi_{pc}$ and TCitH-$\Pi_{lppc}$ to general (low-degree) arithmetic circuits.

Arguments for Low-Degree Arithmetic Circuits. TCitH-$\Pi_{pc}$ is particularly efficient when applied to an arithmetic circuit $C$ of low degree $d$ (i.e. a circuit for which the output coordinates are degree-$d$ polynomials in the input coordinates). 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 $j$th coordinate of the output $y$ and $C_j$ denotes the subcircuit computing this output coordinate. From the analysis of Section 6.2, 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 $d$.

Arguments for General Arithmetic Circuits. TCitH-$\Pi_{lppc}$ can be used to get a sublinear arguments for arithmetic circuits without restrictions on the circuit degree. For this purpose, an arithmetic circuit statement $C(x) = y$ can be expressed as an LPPC instance as in the Ligero proof system [AHIV17].

The witness $w$ is defined as the concatenation of three vectors $a$, $b$ and $c$ whose coordinates $a_j$, $b_j$ and $c_j$ are respectively the first operand, the second operand and the result of the $j$th multiplication gate of $C$ on input $x$. Then proving $C(x) = y$ is similar to proving $c = a \circ b$, where $\circ$ is the coordinate-wise multiplication, and proving the linear constraints of the circuit, which can be expressed as $a = A_1 \cdot w$, $b = A_2 \cdot w$ and $y = A_3 \cdot w$ for some matrices $A_1$, $A_2$, $A_3$.

Denoting $|C|$ the number of multiplication gates in $C$ and assuming that $|C|$ is a multiple of $s$, the witness (of size $|w| = 3|C|$) is easily arranged so that we have $m_1 = |C|/s$ quadratic polynomial constraints of the form $f(a, b, c) = c - a \cdot b$ to verify (each of them verifying $s$ multiplication gates in parallel). We further have $m_2 = 2|C| + |y|$ global linear constraints to verify (for the computation of $a$, $b$ and $y$). For this particular LPPC instance, the MPC protocol $\Pi_{lppc}$ is a slightly optimized version of the Ligero MPC protocol, which we therefore denote $\Pi_{\text{Ligero}}$ here. The zero-knowledge argument obtained when applying our framework is denoted TCitH-$\Pi_{\text{Ligero}}$.

Remark 11. Let us mention that the TCitH-$\Pi_{lppc}$ proof system can deal with circuits with higher-degree gates instead of just considering multiplication gates. This could provide significant gain for some applications but highly depends on the topology of the underlying circuit.

Comparison. Figures 4a and 4b compare our two schemes TCitH-$\Pi_{pc}$ and TCitH-$\Pi_{\text{Ligero}}$ to other MPCitH-based proof systems performing well on small-to-medium size arithmetic circuits, namely Ligero [AHIV17] and Limbo [DOT21]. For completeness, we further plot an optimized version of Ligero (curve “Ligero (opt.)”).

11 Here, the linear constraints of $A_1$ and $A_2$ are not trivial (e.g., forwarding the $a$-part of $w$ to $a$ through an identity matrix $A_1$) but encode the linear relations between the input and the multiplication operands.

12 Compared to the original Ligero MPC protocol, $\Pi_{\text{Ligero}}$ relies on an optimized arithmetization: a textbook application of the Ligero arithmetization includes the input $x$ and all the outputs of the addition gates of $C$ in the witness $w$, which we avoid here by an alternative definition of the linear constraints.
which uses the opened shares to interpolate the broadcast polynomials, instead of sending the entire polynomials (this optimization being natively integrated in the TCitH framework).

From those figures, we observe that TCitH-Π<sub>PC</sub> achieves very competitive cost when the circuit degree is small (below 50). For general circuits, both Ligero and our scheme quickly outperform Limbo when the circuit size increases, and our scheme performs better than Ligero for small-to-medium size circuits, specifically circuits with \( \leq 2^{16} \) multiplication gates. The following table further provides concrete parameters for Ligero vs. TCitH-Π<sub>Ligero</sub> for a circuit with 256 multiplications:

| Scheme       | \(|F|\) | \(|x|\) | \(|C|\) | \(\tau\) | \(N\) | \(\ell\) | \(s\) | \(\eta\) | \(\rho\) | \(\text{Size}\) |
|--------------|--------|--------|--------|--------|------|-------|------|------|------|--------|
| Ligero       | 8192   | 100    | 256    | 1      | 975  | 219   | 106  | –    | 10   | 49.2 kB |
| TCitH-Π<sub>Ligero</sub> | 1924   | 50     | 28     | 40     | 19.2 kB |

| Scheme       | \(|F|\) | \(|x|\) | \(|C|\) | \(\tau\) | \(N\) | \(\ell\) | \(s\) | \(\eta\) | \(\rho\) | \(\text{Size}\) |
|--------------|--------|--------|--------|--------|------|-------|------|------|------|--------|
| Ligero       | 2<sup>32</sup> | 100    | 256    | 1      | 975  | 219   | 106  | –    | 4    | 54.9 kB |
| TCitH-Π<sub>Ligero</sub> | 1924   | 50     | 28     | 17     | 22.6 kB |

(a) Over the field \(F_{8192}\)  
(b) Over the field \(F_{2^{32}}\)

Fig. 4: Comparison of TCitH-Π<sub>PC</sub>, TCitH-Π<sub>Ligero</sub>, Ligero [AHIV17] and Limbo [DOT21] for arithmetic circuits (with input \(x \in F_{100}\)).

### 6.5 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 for polynomial constraints (without packing) TCitH-Π<sub>PC</sub> (Section 6.2) for classical post-quantum one-way functions, namely the multivariate quadratic problem, as used in MQOM [FR23a] 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. We further show how TCitH-Π<sub>PC</sub> can apply to other MPCitH-based schemes submitted to the recent NIST call for additional post-quantum signatures.

**Multivariate Quadratic (MQ) problem over \(\mathbb{F}_q\).** Given matrices \(A_1, \ldots, A_m \in \mathbb{F}^{n \times n}_q\), vectors \(b_1, \ldots, b_m \in \mathbb{F}^n_q\), and scalars \(y_1, \ldots, y_m \in \mathbb{F}_q\), the MQ problem consists in finding a vector \(x\) such that, for all \(j\), \(x^T A_j x + b_j^T x = y_j\).
$y_j$. 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
\[
\forall j \in [1 : m], \quad f_j(x) = x^T A_j x + b_j^T x - y_j.
\]

**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 $y \in \mathbb{F}_q^n$, the SD problem consists in finding a vector $x$ such that $y = Hx$ and such that $x$ has at most $\omega$ 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-$\omega$ 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
\[
\forall j \in [1 : m], \quad f_j(x_A, Q) = \left( e_j^\omega + \sum_{i=0}^{\omega-1} Q_i \cdot e_j^i \right) \cdot \begin{cases} (x_A)_j & \text{if } j \leq k, \\ (y - H x_A)_j & \text{if } j > k. \end{cases}
\]

**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.

<table>
<thead>
<tr>
<th></th>
<th>TCitH-GGM</th>
<th></th>
<th></th>
<th>TCitH-MT</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Size</td>
<td>Signing</td>
<td>Verif</td>
<td>Size</td>
<td>Signing</td>
<td>Verif</td>
</tr>
<tr>
<td>MQ over $\mathbb{F}_{251}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$(n = 43)$</td>
<td>6.575 B</td>
<td>5.57 ms</td>
<td>6.44 ms</td>
<td>7.177 B</td>
<td>3.55 ms</td>
<td>6.63 ms</td>
</tr>
<tr>
<td>SD over $\mathbb{F}_{251}$</td>
<td>8.241 B</td>
<td>6.44 ms</td>
<td>6.45 ms</td>
<td>10.117 B</td>
<td>1.15 ms</td>
<td>6.17 ms</td>
</tr>
<tr>
<td>$(n = 126, k = 79)$</td>
<td>7.335 B</td>
<td>6.43 ms</td>
<td>6.45 ms</td>
<td>10.255 B</td>
<td>1.35 ms</td>
<td>6.30 ms</td>
</tr>
</tbody>
</table>

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.

To complete the overview, we give in Table 4 the sizes we would obtain with the TCitH-GGM framework using alternative parameter sets than those of MQOM and SDitH, namely the multivariate quadratic problem over $\mathbb{F}_4$ and the binary syndrome decoding problem (with the split variant proposed in [FJR22]). In each case, we give the size of the former scheme which led to the best signature size with the considered security assumption.

<table>
<thead>
<tr>
<th></th>
<th>$n$</th>
<th>$m$</th>
<th>$k$</th>
<th>$w$</th>
<th>Former size</th>
<th>TCitH-GGM</th>
<th>Saving</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQ over $\mathbb{F}_4$</td>
<td>88</td>
<td>88</td>
<td>-</td>
<td>-</td>
<td>8,609 B [Wan22]</td>
<td>3,858 B</td>
<td>-55%</td>
</tr>
<tr>
<td>SD over $\mathbb{F}_2$</td>
<td>1280</td>
<td>640</td>
<td>132</td>
<td>11,166 B [FJR22]</td>
<td>7,354 B</td>
<td>-34%</td>
<td></td>
</tr>
<tr>
<td>6-split SD over $\mathbb{F}_2$</td>
<td>1536</td>
<td>888</td>
<td>120</td>
<td>12,066 B [FJR22]</td>
<td>6,974 B</td>
<td>-42%</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Signature sizes using alternative MQ and SD problems.
Application to other NIST post-quantum signature candidates. As explained in Section \ref{sec:other-nist-candidates}, 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 \ref{table:signature-sizes-nist} a list of NIST candidates for which an application of our TCitH-II$_{lc}$ proof system (GGM variant) provides an improvement of the signature size, namely all the MPCitH-based candidates except PERK \cite{ABB+23a} (which is based on the shared-permutation technique \cite{FJR23} and does not fit our framework) and Aim4er \cite{CCH+23}.

As it was the case for the non-structured MQ problem, adapting the NIST candidate Biscuit \cite{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 \cite{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 \cite{ABB+23d} is another NIST candidate that relies on the MinRank problem, but using another verification circuit (based on $q$-polynomials). The idea of MIRA is to show that there exists $\beta \in \mathbb{F}_q^m$ such that $x_j^q + \sum_{i=0}^{r-1} \beta_i \cdot x_j^q = 0$ for all $j$, where $x_j$ is the $j$th column of $E := M_0 + \sum_{j=1}^{k} x_j \cdot M_j \in \mathbb{F}_q^{m \times n}$ written as a field element of $\mathbb{F}_q^m$. Since $\cdot \rightarrow \cdot^q$ is a $\mathbb{F}_q$-linear application, the previous relations lead to degree-2 polynomial constraints. Let us remark that the transcript 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 $x \in \mathbb{F}_q^n$ 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 $\mathbb{F}_q^m$ (which represents a monic degree-$q$ $q$-polynomial of $\mathbb{F}_q[X]$). We thus have that adapting MIRA will lead to larger signature sizes than adapting MiRitH (for the same MinRank parameters). Finally, RYDE \cite{ABB+23c} relies on the syndrome decoding problem in the rank metric which consists, given a matrix $H \in \mathbb{F}_q^{(n-k) \times n}$ and a vector $y \in \mathbb{F}_q^m$, in finding $x \in \mathbb{F}_q^m$ such that the rank of $x$ is smaller than a public bound $r$ and $y = Hx$. As for MIRA, RYDE relies on $q$-polynomials. The idea is to show that $x$ satisfies the desired linear relation and that there exists $\beta \in \mathbb{F}_q^m$ such that $x_j^q + \sum_{i=0}^{r-1} \beta_i \cdot x_j^q = 0$ for all $j$.

<table>
<thead>
<tr>
<th></th>
<th>Original size</th>
<th>Our variant</th>
<th>Saving</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biscuit \cite{BKPV23}</td>
<td>4758 B</td>
<td>4048 B</td>
<td>-15%</td>
</tr>
<tr>
<td>MIRA \cite{ABB+23d}</td>
<td>5640 B</td>
<td>5340 B</td>
<td>-5%</td>
</tr>
<tr>
<td>MIRith-$1a$ \cite{ABB+23b}</td>
<td>5665 B</td>
<td>4694 B</td>
<td>-17%</td>
</tr>
<tr>
<td>MIRith-$1b$ \cite{ABB+23b}</td>
<td>6298 B</td>
<td>5215 B</td>
<td>-17%</td>
</tr>
<tr>
<td>MQOM \cite{FR23a} (over $\mathbb{F}_{251}$)</td>
<td>6355 B</td>
<td>4257 B</td>
<td>-35%</td>
</tr>
<tr>
<td>MQOM \cite{FR23a} (over $\mathbb{F}_{31}$)</td>
<td>6348 B</td>
<td>4027 B</td>
<td>-37%</td>
</tr>
<tr>
<td>RYDE \cite{ABB+23c}</td>
<td>5956 B</td>
<td>5281 B</td>
<td>-11%</td>
</tr>
<tr>
<td>SDitH \cite{FR22} \cite{AFG+23}</td>
<td>8241 B</td>
<td>7335 B</td>
<td>-11%</td>
</tr>
</tbody>
</table>

Table 5: Signature sizes for NIST MPCitH-based candidates.

6.6 Short Post-Quantum Ring Signatures

As another application of TCitH-II$_{lc}$ (Section \ref{sec:other-nist-candidates}), we introduce a new ring signature scheme. Such a scheme allows a user 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 $y_1, \ldots, y_r$ (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 \(v_j \in [1 : r]\), \(s_j = 1\) 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 \(s^{(1)}, \ldots, s^{(d)}\) of size \(\sqrt{r}\) such that

\[
s^{(j)}_{j^*} = s^{(j)}_{j^* + 1} = \cdots = s^{(j)}_{j^* + r - 1} = 1
\]

while the other coordinates of the \(s^{(j)}\)’s equal zero, where \((j^*_1, \ldots, j^*_d)\) is the base-\(\sqrt{r}\) decomposition of \(J^*\) “shifted by one” (which means that \((j^*_1 - 1, \ldots, j^*_d - 1)\) is the standard base-\(\sqrt{r}\) 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 \([s^{(1)}], \ldots, [s^{(d)}]\) by

\[
\forall j, \ [s_j] = [s^{(1)}_{j_1}] \times \cdots \times [s^{(d)}_{j_d}], \tag{17}
\]

with \((j_1, \ldots, j_d)\) the base-\(\sqrt{r}\) 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 6.2, 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 \(\sqrt{r} - 1\) 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 \(\{s^{(j)}\}_j\):

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

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

\[
\forall j \in [1 : d], \ \forall k \in [1 : \sqrt{r}], \ s^{(j)}_k \cdot (\gamma_k - p_j) = 0, \tag{18}
\]

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

\[
\sum_{j=1}^{r} s_j = 1, \tag{19}
\]

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 \([s^{(1)}], \ldots, [s^{(d)}]\) of the one-hot vectors, and the sharings \([p_1], \ldots, [p_d]\) of the non-zero position encodings, then runs the following steps:

1. Compute \([s]\) from \([s^{(1)}], \ldots, [s^{(d)}]\) using Equation \((17)\);
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 \((18)\) and \((19)\).

47
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 6.5 to handle the one-hot vector \( s \). We also propose two additional ring signature schemes based on one-way functions relying on AES.

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:

\[
\begin{align*}
    f_j(x, s^{(1)}, \ldots, s^{(d)}, p) &= x^TA_j x + b_j^T x - \sum_{h=1}^r g_h(s^{(1)}, \ldots, s^{(d)}) \cdot y_h \quad \forall j \in [1 : m], \\
    f'_{j,k}(x, s^{(1)}, \ldots, s^{(d)}, p) &= s_k^{(j)} \cdot (\gamma_j - p_j) \quad \forall j \in [1 : d] \quad \forall k \in [1 : \sqrt{r}], \\
    f''(x, s^{(1)}, \ldots, s^{(d)}, p) &= \sum_{h=1}^r g_h(s^{(1)}, \ldots, s^{(d)}) - 1.
\end{align*}
\]

All these polynomials have degrees at most \( \max\{d, 2\} \), and we can apply the proof system of Section 6.2 (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:

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

All these polynomials have degrees at most \( d + 1 \), and we can apply the proof system of Section 6.2 (with the Fiat-Shamir transformation) to get the desired signature scheme.

**Ring signatures from AES.** We rely on the same arithmetization as the FAEST scheme \( [BBD+23] \); we include the outputs of the S-boxes (split in bits) in the witness and prove the correctness of each S-box by checking a multiplication between two linear combinations of the witness. The exact constraints depend on the considered one-way function:

- When considering \( pk := (x, \text{AES}_{sk}(x)) \), we also include the signer’s public key in the witness. The polynomial constraints are split in two categories: one checks the correctness of the S-boxes using the shared signer’s public key, while the other checks that the shared signer’s public key is consistent with the one-hot encoding of the signer’s index. All the constraints have degree at most \( \max\{d, 2\} \). We could avoid including the signer’s public key in the witness, but in that case, the polynomial constraints would be of degrees \( d + 1 \) (\( d \) to reconstruct \( pk \) from the one-hot encoding and +1 for the multiplications with linear combinations of the witness checking the last S-box layer).

- When considering the Even-Mansour (EM) construction \( pk := (x, \text{AES}_x(sk) \oplus sk) \), we shall use the same AES key \( x \) for all the users of the ring so that only \( \text{AES}_x(sk) \oplus sk \) shall differ between the different public keys. With this variant, we only include the block-cipher output \( \text{AES}_x(sk) \oplus sk \) in the witness to
avoid an increase of the constraint degree. As in the previous case, the polynomial constraints are split in two categories: some check the correctness of the S-boxes, while the other check the correctness of the signer’s public key w.r.t. the one-hot encoding of the signer’s index. All the constraints have degree at most \( \max\{d, 2\} \).

**Remark 12.** Let us mention that we do not consider multi-user attacks over the private key in this article for the sake of simplicity (to have simpler polynomial constraints). Taking the same coefficients for the MQ instances, the same matrix for the SD instances, or the same AES key for a given ring would degrade the overall security vs. multi-user attacks if a large ring is considered. To prevent this type of attacks, one should use independent public keys (i.e. different coefficients for the MQ instances, different AES key \( x \) for the EM variant, etc.). This would slightly increase the signature size because the witness and/or the constraint degree would be larger. For example, a MQ-based ring signature scheme secure against multi-user attacks has size around 9.7 KB (instead of 8.2, see Table 6) for \( 2^{20} \) users when relying on \( \mathbb{F}_{251} \). This would also slightly increase the signing/verification time since the MPC protocol would be heavier.

**Benchmark.** The obtained MQ-based and SD-based signature sizes and running times are depicted in Figure 5 with respect to an increasing ring size. 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 the MQ-based and SD-based 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 6. We can remark that we achieve the smallest signature sizes in all the cases with our MQ-based scheme.

### 6.7 Exact Zero-Knowledge Arguments for Lattices

We now present an application of our generic proof system TCitH-\( \Pi \rangle_{\text{LPPC}} \) (Section 6.3) to obtain short and exact zero-knowledge arguments for lattice problems. Previous tries using the MPC-in-the-Head paradigm lead to large proof transcripts. This comes from the fact that an MPCitH-based proof includes a share of the secret vector and that the latter lives in a large space. In [FMRV22], the authors use sharing over the integers with rejection in order to lower the communication cost of the shares of the secret vector since the latter is known to be small (i.e. of a small norm). Another option to reduce this cost is to use packed secret sharing with our TCitH-MT framework, which we develop here.

In what follows, we show how to apply TCitH-\( \Pi \rangle_{\text{LPPC}} \) to check an instance of an Inhomogeneous Short Integer Solution (ISIS) problem. Such a scheme proves the knowledge of a vector \( e \in \mathbb{F}_q^n \) such that \( \|e\|_{\infty} \leq \beta \) and \( t = A \cdot e \), where \( A \in \mathbb{F}_q^{m \times n} \) and \( t \in \mathbb{F}_q^m \) are publicly known. We stress that the Learning With Error (LWE) problem (as well as its ring or module variants) can be expressed as a particular form of ISIS with \( A = (A' | I_n) \).
Let us denote $s$ the pack size of the used secret sharing and assume that $n$ is a multiple of $s$ (otherwise we shall pad $e$ and $A$ with zeros). We decompose $e$ as $k := \lceil \log_2(2\beta+1) \rceil$ vectors $(e^{(0)}, \ldots, e^{(k-1)})$ of $\{0,1\}^n$ such that
\[
e = \sum_{i=0}^{k-2} 2^i e^{(i)} + (2\beta - 2^{k-1} + 1)e^{(k-1)} - \beta
\]  
(20)
where $\beta = (\beta, \ldots, \beta) \in \mathbb{F}_q^n$. If all vectors $e^{(i)}$ belong to $\{0,1\}^n$, the above relation gives that $\|e\|_{\infty} \leq \beta$. So, as in FMRV22, we give the sharing $[w]$ of the witness $w = (e^{(0)} \parallel \cdots \parallel e^{(k-1)})$ to the MPC protocol instead of $[e]$. The latter can then check that $[w]$ is the sharing of a binary vector and that $A \cdot [e]$ corresponds to $t$ modulo $q$ where $[e]$ is recovered by linearity of (20). This can be expressed as an instance of the $\Pi$ protocol (Section 6.3) with the following global linear constraints:
\[
A \cdot (I_n | 2I_n | \cdots | 2^{k-2}I_n | (2\beta - 2^{k-1} + 1)I_n) \cdot \begin{pmatrix}
e^{(0)} \\
\vdots \\
e^{(k-1)}
\end{pmatrix} = t + A \cdot \beta
\]
\[e+\beta\]
and the following parallel polynomial constraints:
\[
\forall j \in \{0, \ldots, k-1\}, \ e^{(j)} \circ (e^{(j)} - 1) = 0 ,
\]
where \(0 = (0, \ldots, 0) \in \mathbb{F}_q^n\), \(1 = (1, \ldots, 1) \in \mathbb{F}_q^n\) and \(\circ\) is the coordinate-wise multiplication. By applying the TCitH-II_{LPPC} proof system of Section 6.2 with the variant TCitH-MT, the size of the proof transcript is given by Equation (10), where the degree \(d\) is 2 and the packed witness size is \(k \cdot \lceil n/s \rceil\).

We apply this proof system on toy ISIS instances with \(q \approx 2^{32}\) and \(q \approx 2^{64}\) from the lattice zero-knowledge literature as well as on Kyber \(ABD^{+}21\) and Dilithium \(BDK^{+}21\) instances. The results are depicted in Table 7. We further compare our results on the toy instances to previous works in Table 8. Our arguments are 3 to 5 times smaller than the previous best MPCitH-based arguments \(FMRV22\). Let us remark that \(FMRV22\) supports ISIS instances with any value of \(q\) while we require a prime \(q\) (due to the use of Shamir’s secret sharing), which is not a strong restriction in practice. Previous works relying on lattice techniques \(LNS21\) \(LNP22\) achieve better sizes on the first toy instance \(33\, KB \text{ and } 15\, KB \text{ vs. } 41\, KB\) but either rely on NTT-friendly fields \(LNS21\) or prove the \(L_2\) norm \(LNP22\) (instead of proving the \(L_\infty\) norm as the other schemes). Regarding the Kyber512 instance, our size \(21\, KB\) is close to the 19 KB obtained by \(LNP22\) on a slightly different instance arising in a verifiable encryption use case (and still for the \(L_2\) norm against \(L_\infty\) in our case).

Table 7: Proof sizes when applied on some SIS/LWE instances. “\(\beta = 1/2\)” means that the secret is a binary vector.

<table>
<thead>
<tr>
<th>SIS/LWE Instance</th>
<th>(q)</th>
<th>(n)</th>
<th>(m)</th>
<th>(\beta)</th>
<th>(N)</th>
<th>(s)</th>
<th>(n)</th>
<th>(\rho)</th>
<th>Proof Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toy Example 1</td>
<td>(\approx 2^{32})</td>
<td>2048</td>
<td>1024</td>
<td>1</td>
<td>1024</td>
<td>1</td>
<td>1024</td>
<td>4</td>
<td>38,081 B</td>
</tr>
<tr>
<td>Toy Example 2</td>
<td>(\approx 2^{64})</td>
<td>4096</td>
<td>512</td>
<td>(1/2)</td>
<td>1024</td>
<td>1</td>
<td>1024</td>
<td>1044</td>
<td>57,409 B</td>
</tr>
<tr>
<td>Kyber512</td>
<td>3329</td>
<td>((2 + 2) \times 256)</td>
<td>2 \times 256</td>
<td>3</td>
<td>1024</td>
<td>1</td>
<td>1024</td>
<td>67,988</td>
<td>21,185 B</td>
</tr>
<tr>
<td>Kyber768</td>
<td>3329</td>
<td>((3 + 3) \times 256)</td>
<td>3 \times 256</td>
<td>2</td>
<td>1024</td>
<td>1</td>
<td>1024</td>
<td>24,938</td>
<td>21,185 B</td>
</tr>
<tr>
<td>Kyber1024</td>
<td>3329</td>
<td>((4 + 4) \times 256)</td>
<td>4 \times 256</td>
<td>2</td>
<td>1024</td>
<td>1</td>
<td>1024</td>
<td>27,241</td>
<td>21,185 B</td>
</tr>
<tr>
<td>Dilithium2</td>
<td>(330,081)</td>
<td>((4 + 4) \times 256)</td>
<td>4 \times 256</td>
<td>2</td>
<td>1024</td>
<td>1</td>
<td>1024</td>
<td>28,241</td>
<td>10,010 B</td>
</tr>
<tr>
<td>Dilithium3</td>
<td>(330,081)</td>
<td>((6 + 5) \times 256)</td>
<td>6 \times 256</td>
<td>4</td>
<td>1024</td>
<td>1</td>
<td>1024</td>
<td>40,100</td>
<td>57,526 B</td>
</tr>
<tr>
<td>Dilithium3</td>
<td>(330,081)</td>
<td>((8 + 7) \times 256)</td>
<td>8 \times 256</td>
<td>2</td>
<td>1024</td>
<td>1</td>
<td>1024</td>
<td>68,088</td>
<td>57,526 B</td>
</tr>
</tbody>
</table>

Table 8: Comparison with the existing exact protocols which prove the knowledge of the solution of a ISIS instance. (*): All the schemes prove the infinity norm, except \(LNP22\) that proves the \(L_2\)-norm.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Year</th>
<th>Any (q)</th>
<th>Toy Example 1</th>
<th>Toy Example 2</th>
<th>Proof Size</th>
<th>Rej. Rate</th>
<th>Proof Size</th>
<th>Rej. Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>LNSW13</td>
<td>2013</td>
<td>(\checkmark)</td>
<td>3600 KB</td>
<td>0</td>
<td>8988 KB</td>
<td>0</td>
<td>8988 KB</td>
<td>0</td>
</tr>
<tr>
<td>Lingo</td>
<td>2017</td>
<td>(q) prime + NTT</td>
<td>157 KB</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Aurora</td>
<td>2019</td>
<td>(q) prime + NTT</td>
<td>71 KB</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BLS19</td>
<td>2019</td>
<td>(q) prime + NTT</td>
<td>384 KB</td>
<td>0.92</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BN20</td>
<td>2020</td>
<td>(q) prime</td>
<td>-</td>
<td>-</td>
<td>4077 KB</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ben20</td>
<td>2020</td>
<td>(q) prime</td>
<td>233 KB</td>
<td>0</td>
<td>444 KB</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LNS20</td>
<td>2020</td>
<td>(q) prime + NTT</td>
<td>47 KB</td>
<td>0.95</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LNP22</td>
<td>2021</td>
<td>(q) prime + NTT</td>
<td>33.7 KB</td>
<td>0.86</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FMRV22</td>
<td>2022</td>
<td>(\checkmark)</td>
<td>291 KB</td>
<td>0.04</td>
<td>291 KB</td>
<td>0.04</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LNP22</td>
<td>2022</td>
<td>(\checkmark)</td>
<td>184 KB</td>
<td>0.05</td>
<td>184 KB</td>
<td>0.05</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Our scheme</td>
<td>2023</td>
<td>(q) prime</td>
<td>38 KB</td>
<td>0</td>
<td>57 KB</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Remark 13. Let us remark that we can directly check that the ISIS secret $e$ satisfies the relation $(e - \beta) \circ (e - \beta + 1) \circ \ldots \circ (e + \beta) = 0$, which is a $(2\beta + 1)$-degree constraint (instead of decomposing the secret). We can also mix: decompose $e$ in a larger base to have constraints between $2$ and $2\beta + 1$. According to our tests, we obtain similar or higher argument sizes.

We also apply our proof system to custom instances of LWE. The selected parameters have similar security than Kyber512 according to the lattice estimator [APS15]. In our experiments, we observed that taking a smaller $q$ led to shorter arguments up to a point where the constraint on the number of parties ($N + s \leq q + 1$) became detrimental to efficient packing. Table 9 provides the selected parameters for $q = 251$ and $q = 1021$ ($\beta = 1/2$ means that the small vectors are binary). For the latter, we achieve argument size close to 15 KB.

Table 9: Proof sizes for LWE instances.

<table>
<thead>
<tr>
<th>LWE Instance</th>
<th>$q$</th>
<th>$n$</th>
<th>$p$</th>
<th>$\beta$</th>
<th>$N$</th>
<th>$T$</th>
<th>$d$</th>
<th>$s$</th>
<th>$\ell$</th>
<th>$\rho$</th>
<th>$\eta$</th>
<th>Proof Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>LWE over $F_{1021}$</td>
<td>1021</td>
<td>561</td>
<td>1/2</td>
<td>1000</td>
<td>1.34</td>
<td>148</td>
<td>12</td>
<td>13</td>
<td>15</td>
<td>13</td>
<td>15.365</td>
<td>B</td>
</tr>
<tr>
<td>LWE over $F_{251}$</td>
<td>251</td>
<td>473</td>
<td>1/2</td>
<td>240</td>
<td>1.10</td>
<td>6</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17.781</td>
<td>B</td>
</tr>
</tbody>
</table>

Remark 14. It would be interesting to investigate the application of our proof system to FrodoKEM [ABD+23]. The security of FrodoKEM relies on unstructured lattice assumptions, unlike the security of Kyber [ABD+21] and Dilithium [BDK+21a] which relies on module lattice assumptions. However, FrodoKEM is defined over $\mathbb{Z}/2^k\mathbb{Z}$ for $k \in \{15, 16\}$. Therefore, our techniques are not directly applicable since Shamir’s secret sharing requires working on a field, or at least on a ring for which there exists a large subset $E$ such that $x - y$ is invertible for all $(x, y) \in E^2$. To apply our proof system to FrodoKEM, one could try to adapt techniques from [BdSGJ+24]. Such an adaptation is left for future work.

7 Connections to Other MPCitH-like Proof Systems

This section compares our framework to other MPCitH-like proof systems, namely VOLE-in-the-Head [BBdSG+23] and Ligero [AHIV17, AHIV23]. While the TCitH framework consists in an extension of standard MPC-in-the-Head proof systems with (packed) Shamir’s secret sharing, we establish strong connections between some of its variants and/or instantiations and these proof systems. Those connections are illustrated in Figure 6.

7.1 Connections to VOLE-in-the-Head

The VOLE-in-the-Head (VOLEitH) framework [BBdSG+23] was published at CRYPTO 2023 concurrently to the early stages of our work. This framework provides a way to compile any 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.

Let us recall that a VOLE (for vector oblivious linear evaluation) is a multiparty gadget, where one party called prover, learns a pair $u \in \mathbb{F}_p^\ell, v \in \mathbb{F}_p^\ell$, while the verifier learns a random $\Delta \in \mathbb{F}_p^\ell$ and $q = u \cdot \Delta + v \in \mathbb{F}_p^\ell$. 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 (non-packed) $(\ell + 1 = 2, N)$-threshold Shamir’s secret sharing of $u$, for which the secret is stored at $P_u(\infty)$ (i.e., for evaluation point $\omega_1 = \infty$ instead of the usual $\omega_1 = 0$). Namely, to share $u$,
MPCitH with additive sharing, e.g. [KKW18, BN20, DOT21]

Original TCitH framework [FR23b]

Application of Shamir’s secret sharing with Merkle tree commitments

+ GGM variant
+ packed secret sharing
+ non-linear round functions
+ degree-enforcing MT commitment

General TCitH framework (this work)

TCitH with $H_{\text{Ligero}}$ - optimised version of the Ligero concrete scheme

Ligero [AHIV17, AHIV23]

VOLE-in-the-Head [BBdSG23]

Fig. 6: The TCitH framework in the landscape of MPCitH-like proof systems.

- 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 $\ell = s = 1$ derived from [CDI05] (see 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 (4), 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-$\ell$ 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 (4) with $\ell = 1$, we obtain (up to the auxiliary value)

$$
[u]_i = \sum_{j \neq i} s_{(j)} \cdot P_{j}(e_i)
= \sum_{j \neq i} s_{(j)} \cdot (e_i - e_j)
$$

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 $\Pi_{\text{pc}}$ of Section 6.2 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 $\ell = s = 1$ applied to $\Pi_{\text{pc}}$.

53
Large field embedding. Compared to the TCitH-GGM-$P_{\text{rec}}$ proof system, the VOLEitH construction relies on an additional optimization which we shall call large field embedding. In what follows, we explain this optimization under our formalism (i.e., $\ell = s = 1$ Shamir’s secret sharings rather than VOLE correlation), thereby showing that it can also be applied to the TCitH framework. Let us assume that we have $\tau$ sharings $[u]_1^{(1)}, \ldots, [u]_1^{(\tau)}$ 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 $\phi$ between $\mathbb{F}_{q^\tau}$ and $\mathbb{F}_{q^\tau}$. Then, the $N^\tau$-sharing $[w]_1^{(\phi)}$ defined as

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

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

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

So, $\tau N$-sharings of the same value can be seen as a single $N^\tau$-sharing of this value (living in the field extension $\mathbb{F}_{q^\tau}$). Then, instead of executing $\tau$ times the MPC protocol on these $\tau$ 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 $\frac{d}{d\alpha}$ instead of $(\frac{d}{d\alpha})^{\tau}$, where $d_\alpha$ is the maximal degree of the broadcasted sharings. However, it implies executing the MPC protocol in $\mathbb{F}_{q^\tau}$ instead of executing it $\tau$ 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 $\tau$ 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.

Let us compare a pure usage of TCitH-GGM (with $\ell = s = 1$) and VOLEitH (featuring the large field embedding). We provide in Tables 10 and 11 the sizes we obtain when applying TCitH-GGM and VOLEitH to the MPCitH-based candidates to the new NIST call for post-quantum signatures, which we adapt as explained in Section 6.5. In Table 10 the estimated sizes are obtained when using seed trees with 256 leaves (most common choice in the NIST MPCitH-based submissions), while in Table 11 the estimated sizes are obtained when using seed trees with 2048 leaves (let us remark that the short version of FAEST [BRD+23], the VOLEitH-based candidate to the NIST call, relies on seed trees with 2048 to 4096 leaves). In the first case, the size differences are smaller than 600 bytes (except for SDitH), and they are even smaller in the second case (less than 250 bytes). However, with a pure application of TCitH, we do not rely on a large field extension as in VOLEitH, as stressed in the columns “Computat. Field” of those tables. For example, for MQOM-251 in Table 11 we run 13 times the MPC protocol over $\mathbb{F}_{2512}$ while VOLEitH run the MPC protocol only once, but in the field extension $\mathbb{F}_{25124}$. We thus expect the overall computation cost for the MPC emulation to be smaller with a pure usage of TCitH-GGM. This difference will be particularly significant when for use cases where the MPC emulation is the bottleneck as, e.g., the ring signatures presented in Section 6.6.

Let us stress that VOLEitH when interpreted as a particular case of the TCitH-GGM variant is limited to Shamir’s secret sharings with $\ell = s = 1$. In some contexts, it might be interesting to use the TCitH-GGM variant with greater values of $s$ and $\ell$. Moreover, while the large field embedding of VOLEitH only supports $\ell = s = 1$, it could also be used in the TCitH-MT variant hence benefiting from a faster verification.

7.2 Connections to Ligero

As in Ligero, our framework with packed secret sharing, the TCitH-MT variant, relies on committing Shamir’s secret sharings (a.k.a. randomized Reed-Solomon codewords) using a Merkle tree and opening some shares
<table>
<thead>
<tr>
<th>Name</th>
<th>TCitH-GGM</th>
<th>VOLEitH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Size</td>
<td>Comput. Field</td>
</tr>
<tr>
<td>AIMer [CCH\textsuperscript{23}]</td>
<td>4 352 B</td>
<td>19× GF(2\textsuperscript{8})</td>
</tr>
<tr>
<td>Biscuit [BKPV23]</td>
<td>4 048 B</td>
<td>19× GF(16\textsuperscript{2})</td>
</tr>
<tr>
<td>MIRA [ABB\textsuperscript{23d}]</td>
<td>5 340 B</td>
<td>19× GF(16\textsuperscript{2})</td>
</tr>
<tr>
<td>MiRitH-Ia [ABB\textsuperscript{23b}]</td>
<td>4 694 B</td>
<td>19× GF(16\textsuperscript{2})</td>
</tr>
<tr>
<td>MiRitH-Ib [ABB\textsuperscript{23b}]</td>
<td>5 245 B</td>
<td>19× GF(16\textsuperscript{2})</td>
</tr>
<tr>
<td>MQOM (over $\mathbb{F}_{251}$) [FR23a]</td>
<td>4 257 B</td>
<td>19× GF(251)</td>
</tr>
<tr>
<td>MQOM (over $\mathbb{F}_{31}$) [FR23a]</td>
<td>4 027 B</td>
<td>19× GF(31\textsuperscript{2})</td>
</tr>
<tr>
<td>RYDE [ABB\textsuperscript{23c}]</td>
<td>5 281 B</td>
<td>19× GF(2\textsuperscript{11})</td>
</tr>
<tr>
<td>SDitH (over $\mathbb{F}_{251}$) [AFG\textsuperscript{23}]</td>
<td>7 335 B</td>
<td>19× GF(251\textsuperscript{12})</td>
</tr>
<tr>
<td>SDitH (over $\mathbb{F}_{256}$) [AFG\textsuperscript{23}]</td>
<td>7 335 B</td>
<td>19× GF(256\textsuperscript{12})</td>
</tr>
</tbody>
</table>

Table 10: Comparison between TCitH-GGM and VOLEitH using GGM trees of 256 leaves.

<table>
<thead>
<tr>
<th>Name</th>
<th>TCitH-GGM</th>
<th>VOLEitH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Size</td>
<td>Comput. Field</td>
</tr>
<tr>
<td>AIMer [CCH\textsuperscript{23}]</td>
<td>3 639 B</td>
<td>13× GF(2\textsuperscript{11})</td>
</tr>
<tr>
<td>Biscuit [BKPV23]</td>
<td>3 431 B</td>
<td>13× GF(16\textsuperscript{3})</td>
</tr>
<tr>
<td>MIRA [ABB\textsuperscript{23d}]</td>
<td>4 314 B</td>
<td>13× GF(16\textsuperscript{3})</td>
</tr>
<tr>
<td>MiRitH-Ia [ABB\textsuperscript{23b}]</td>
<td>3 873 B</td>
<td>13× GF(16\textsuperscript{3})</td>
</tr>
<tr>
<td>MiRitH-Ib [ABB\textsuperscript{23b}]</td>
<td>4 250 B</td>
<td>13× GF(16\textsuperscript{3})</td>
</tr>
<tr>
<td>MQOM (over $\mathbb{F}_{251}$) [FR23a]</td>
<td>4 257 B</td>
<td>13× GF(251\textsuperscript{3})</td>
</tr>
<tr>
<td>MQOM (over $\mathbb{F}_{31}$) [FR23a]</td>
<td>4 118 B</td>
<td>13× GF(31\textsuperscript{3})</td>
</tr>
<tr>
<td>RYDE [ABB\textsuperscript{23c}]</td>
<td>4 274 B</td>
<td>13× GF(2\textsuperscript{11})</td>
</tr>
<tr>
<td>SDitH (over $\mathbb{F}_{251}$) [AFG\textsuperscript{23}]</td>
<td>5 673 B</td>
<td>13× GF(251\textsuperscript{2})</td>
</tr>
<tr>
<td>SDitH (over $\mathbb{F}_{256}$) [AFG\textsuperscript{23}]</td>
<td>5 673 B</td>
<td>13× GF(256\textsuperscript{2})</td>
</tr>
</tbody>
</table>

Table 11: Comparison between TCitH-GGM and VOLEitH using GGM trees of 2048 leaves.
(a.k.a. codeword coordinates) requested by the verifier. Although conceptually close, our framework is less restrictive in the covered MPC protocols and it achieves smaller proof sizes for “small to medium size” statements.

As our framework, Ligero first address a “general case” which provides a generic transformation from an MPC protocol to a zero-knowledge proof system. Our framework distinguishes itself from this transformation in several key aspects. Firstly, our framework eliminates the need for the robustness property from the MPC protocol and simply requires the latter to be secure against a passive adversary. In addition, unlike the “general case” of Ligero, we consider MPC protocols that might have non-negligible false positive probability and which might rely on hints (obtained through a hint oracle). This makes our framework compatible with many existing protocols from the MPC-in-the-Head literature which commonly rely on hints and usually have non-negligible false positive probability. Finally, the soundness error achieved by our general transformation is about \((\frac{d}{\alpha})/\left(\frac{N}{\ell}\right)^{\ell/2}\), where \(d\) is the maximal degree of broadcast shares (which is much smaller than \(N\)) against \((1 - t_r/N)^{\ell}\) for the general Ligero transformation, where \(t_r\) is the robustness threshold, which gives us better sizes for “small to medium size” circuits given the selection of the parameters \(d\) and \(t_r\) in this regime.

Ligero also provides a concrete zero-knowledge proof system for arithmetic circuits. As shown in Section 5.3, our framework can be applied to the MPC protocol \(H_{\text{Ligero}}\), which is an optimized version of the Ligero protocol to check an arithmetic circuit. The obtained specific instantiation of our framework, TCitH-\(H_{\text{Ligero}}\), is close to this concrete proof system but achieves smaller proof sizes (see Figures 4a and 4b) for circuits with \(\leq 2^{16}\) multiplication gates. One key ingredient to this improvement is our degree-enforcing commitment scheme described in Section 3 (and further detailed in Section 5.2) and the related soundness analysis. The concrete scheme of Ligero includes a proximity test ensuring that the committed sharings are “close” to sharings of degree \(d\). Here close means that the distance between the sharing and the closest degree-\(d\) sharing is lower than \((N - d)/2\) (i.e., less than \((N - d)/2\) non-equal evaluations). This leaves a lot of room to a malicious prover for cheating by committing inconsistent sharings and answering MPC challenges in a way to maximize their cheating probability. In comparison, our degree-enforcing commitment scheme ensures that the committed sharings are exactly of the expected degree (which can be \(d = s + \ell - 1\) or larger) by adding an additional prover-verifier interaction, with a tunable soundness error (which is usually made negligible in practical instantiations).

To conclude, the TCitH framework provides a versatile framework to build zero-knowledge arguments and post-quantum signatures from (passively secure) MPC protocols. Surprisingly, this framework originated by applying (packed) Shamir’s secret sharing to standard MPC-in-the-Head constructions bridges this paradigm to other proof systems such as VOLE-in-the-Head and Ligero while offering further genericity and improvement which we showcase through different applications. We hope this unification will foster further innovation in the area of short hash-based (or symmetric cryptography-based) zero-knowledge arguments.

Acknowledgements. This work was partly supported by the RESQUE project, Bpifrance, France national quantum strategy. The authors would like to thank Loïc Bidoux and Philippe Gaborit for fruitful discussions relating to the small field handling in the TCitH framework.

References


A Analysis of Ligero’s Proximity Test

In the following, $\Delta(v, w)$ denotes the distance between any two vectors $v, w \in \mathbb{F}^N$, namely the number of non-zero coordinates in $v - w$. We further denote $\Delta(v, L)$ the minimal distance $\min_{w \in L} \Delta(v, w)$ for a set $L \subseteq \mathbb{F}^N$. We say that $v$ is $e$-far from $w$ (resp. $L$) if $\Delta(v, w) \geq e$ (resp. $\Delta(v, L) \geq e$) and that $v$ is $e$-close to $w$ (resp. $L$) if $\Delta(v, w) \leq e$ (resp. $\Delta(v, L) \leq e$).

Let us recall that a Reed-Solomon code encodes a word $v \in \mathbb{F}^{d+1}$ into a codeword in $\mathbb{F}^N$ by interpolating a degree-$d$ polynomial $P$ such that $(P(\omega_1), \ldots, P(\omega_d+1)) = v$, for some fixed evaluation points $\{\omega_i\}$ and defining the associated codeword formed by $N$ evaluations $(P(e_1), \ldots, P(e_N))$ for fixed evaluation points $\{e_i\}$ disjoint from $\{\omega_i\}$. In other words, a Shamir’s secret sharing of $w = (w_1, \ldots, w_s) \in \mathbb{F}^s$ with privacy threshold $\ell$ is the Reed-Solomon codeword associated to $(w_1, \ldots, w_s, r_1, \ldots, r_\ell)$ for random elements $r_1, \ldots, r_\ell \in \mathbb{F}$. Let us denote $L$ the Reed-Solomon code with the above parameters and $L^n$ the interleaved code composed of $n$-row matrices whose rows are elements of $L$. Let us denote $U^*$ the matrix where the $i$th row corresponds to the possibly-invalid $i$th committed sharing $\llbracket w_i \rrbracket$. We denote $\Delta(U^*, L^n)$ the minimal distance between $U^*$ and $L^n$, i.e. the minimal number of columns of $U^*$ which should be changed to obtain an element of $L^n$. We have the following result from [AHIV17, Lemma 4.2]:

**Lemma 4 ([AHIV17]).** Let $e$ be a positive integer such that $e < \delta/4$, with $\delta = N - d$. Suppose $\Delta(U^*, L^n) > e$. Then, for a random $w^*$ in the row-span of $U^*$, we have

$$\Pr[\Delta(w^*, L) \leq e] \leq \frac{e + 1}{|\mathbb{F}|}.$$ 

Let us rephrase this result with a sharing-based formulation. Let $e$ be a positive integer such that $e < \frac{N - d}{4}$. Suppose that $\llbracket w_1 \rrbracket, \ldots, \llbracket w_n \rrbracket$ differ from $n$ valid Shamir’s secret sharings for strictly more than $e$ parties. We get that the probability that a linear combination of those sharings differs from a valid sharing for at most $e$ parties is less than $(e + 1)/|\mathbb{F}|$:

$$\Pr_{\gamma_1, \ldots, \gamma_n \in \mathbb{F}} \left[ \sum_i \gamma_i \cdot \llbracket w_i \rrbracket \text{ differs from a valid sharing for at most } e \text{ parties} \right] \leq \frac{e + 1}{|\mathbb{F}|}. $$

We can use this approach to fix the soundness degradation of the original TCitH framework [FR23b]. After generating and committing the shares, the prover would receive random coefficient $\gamma_1, \ldots, \gamma_n$ from the verifier (e.g., together with some challenge for the MPC protocol) and return a response supposedly equal to $w^* = \sum_i \gamma_i \cdot \llbracket w_i \rrbracket$ before receiving the party opening challenge. In the final check, the verifier would check this response to be of the right degree and to be consistent with the opened parties. We analyze hereafter how such a tweak improves the soundness of the original TCitH framework. Let us recall that in the absence of false positive event a malicious prover against the original TCitH framework is left with cheating on the computation of exactly $N - \ell$ parties, which yields an acceptance probability of $1/\binom{N}{\ell}$ [13]. Let us note accept the event that the verifier accepts the proof transcript. Against this tweaked TCitH framework, a malicious prover has three possible strategies:

---

[13] This comes from the fact that the malicious prover should choose broadcasted sharings in the MPC emulation which are valid, meaning that they correspond to the evaluations of degree-$\ell$ polynomials (since we consider linear multiparty computation as in [FR23b], all the sharings involved in the MPC protocol corresponds to the evaluations of degree-$\ell$ polynomials; we consider a more general setting in the next section). Let us denote $Q_1, \ldots, Q_{|\omega|}$ those polynomials and $Q'_1, \ldots, Q'_{|\omega|}$ the degree-$\ell$ polynomials which correspond to the broadcasted sharing in a honest MPC emulation with $\llbracket w_1 \rrbracket, \ldots, \llbracket w_n \rrbracket$. Since there is no false positive event, we get that there is (at least) one index $j'$ such that $Q_{j'} \neq Q'_{j'}$. The probability that the verifier is convinced is the probability that $Q_{j'}$ and $Q'_{j'}$ match for the evaluation points of the $\ell$ opened parties. Since there are of degree $\ell$, these polynomials can match on at most $\ell$ points according to the Schwartz-Zippel Lemma, and so the verifier accepts the proof transcript if and only if these evaluation points are exactly those of the opened parties.
1. They commit valid secret sharings of the witness \( \text{i.e. } \Delta(U^*,L^{m}) = 0 \). In that case, the optimal cheating strategy consists in trying to obtain a false positive events in the MPC protocol (namely, the MPC protocol fails to detect that the witness does not satisfy the proved statement), and in the absence of false positive, cheating on the computation of \( N - \ell \) parties. We get

\[
\Pr[\text{accept}] \leq \frac{1}{(\ell)} + p \cdot \left(1 - \frac{1}{(\ell)}\right).
\]

2. They commit invalid secret sharings which are \( f \)-far from valid sharings, with \( f \geq \frac{\delta}{4} \). According to Lemma with \( e = \left\lceil \frac{\delta}{4} \right\rceil - 1 \), the probability that a random linear combination is \( (\left\lceil \frac{\delta}{4} \right\rceil - 1) \)-close from a valid sharing is at most \( \left(\frac{\delta}{4}\right) \). When this event does not occur, we know that there are at least \( \left\lceil \frac{\delta}{4} \right\rceil - 1 \) invalid parties, and so the probability that the malicious prover convinces the verifier corresponds to the probability that the \( \ell \) open parties are not among those invalid parties. Let us denote \( f' = \Delta(w^*,L) \), where \( w^* = \sum_i \gamma_i \cdot [w_i] \) is the requested random linear combination of the committed sharings. We thus have

\[
\Pr[\text{accept}] \leq \Pr[f' < \frac{\delta}{4}] + \Pr[\text{accept} | f' \geq \frac{\delta}{4}] \leq \frac{\left\lceil \frac{\delta}{4} \right\rceil}{(\ell)} + \frac{(N-\left\lceil \frac{\delta}{4} \right\rceil)}{(\ell)}.
\]

3. They commit invalid secret sharings which are \( f \)-far from valid sharings, with \( 1 \leq f < \frac{\delta}{4} \). According to Lemma with \( e = f - 1 \), the probability that a random linear combination is \( (f - 1) \)-close from a valid sharing is at most \( \frac{f}{\ell} \). Since we are below the unique decoding bound \( (f < \frac{\delta}{4}) \), there exist unique valid sharings \( [w_1]', \ldots, [w_n]' \) which are \( f \)-close from \( [w_1], \ldots, [w_n] \). As above, let us denote \( f' = \Delta(w^*, L) \), where \( w^* = \sum_i \gamma_i \cdot [w_i] \) is the requested random linear combination of the committed sharings. Let us further denote \( \hat{w} \) the actual response from the prover. Whenever \( f' \geq f \), we have \( f' = f \) (since by definition \( f' \leq f \)) and \( \sum_i \gamma_i \cdot [w_i]' \) is the closest valid sharing to \( w^* \) (which is precisely \( f \)-close to \( w^* \)). Assume the prover always choose to reply \( \hat{w} := \sum_i \gamma_i \cdot [w_i]' \). We then have

\[
\Pr[\text{accept}] \leq \Pr[f' < f] + \Pr[\text{accept} | f' \geq f, \text{FP}] \cdot \Pr[\text{FP}] + \Pr[\text{accept} | f' \geq f, \text{\neg FP}] \cdot \Pr[\text{\neg FP}]
\]

with FP the false positive event occurring for the witness encoding in \( [w_1]', \ldots, [w_n]' \). The left probability is bounded using Lemma. Then if \( f' \geq f \), we have at least \( f \) invalid parties, implying \( \Pr[\text{accept} | f' \geq f, \text{FP}] \leq \left(\frac{f}{\ell}\right) \). Finally, if \( f' \geq f \) and no false positive event occurs, the prover must cheat on exactly \( N - \ell \) parties as explained above and we have \( \Pr[\text{accept} | f' \geq f, \text{\neg FP}] = 1/\left(\binom{N}{\ell}\right) \).

Now (still whenever \( f' \geq f \)), assume the prover does not respond \( \hat{w} := \sum_i \gamma_i \cdot [w_i]' \). This means that the prover replies with \( \hat{w} \) which distance to \( \sum_i \gamma_i \cdot [w_i]' \) is \( \delta \) (the minimal distance of the RS code). Since we have \( \Delta(w^*, \sum_i \gamma_i \cdot [w_i]) = f \), we deduce \( \Delta(w^*, \hat{w}) \geq \delta - f > 3\delta/4 \). In that case, we know that there are at least \( 3\delta/4 \) invalid parties, and so the probability that the malicious prover convinces the verifier corresponds to the probability that the \( \ell \) open parties are not among those invalid parties. We deduce

\[
\Pr[\text{accept}] \leq \Pr[f' < f] + \Pr[\text{accept} | f' \geq f] < \frac{\delta/4}{(\ell)} + \frac{(N-\delta/4)}{(\ell)}.
\]

The above probability is strictly smaller than the probability obtained with the second strategy. Therefore, we assume that the adversary always respond with \( \hat{w} = \sum_i \gamma_i \cdot [w_i]' \) while using the third strategy (since otherwise they would rather use the second strategy).

We can remark that the upper bound on \( \Pr[\text{accept}] \) in the first cheating strategy corresponds to the upper bound of the third strategy with \( f = 0 \). From these upper bounds, we obtain the following global upper
bound on the soundness error:

\[
\epsilon \leq \max \left\{ \frac{\lfloor \frac{\ell}{4} \rfloor}{|\mathcal{F}|} + \frac{(N-\lfloor \frac{\ell}{4} \rfloor)}{(\frac{\ell}{4})} \right\} \cup \left\{ \frac{f}{|\mathcal{F}|} + \frac{(N-f)}{(\frac{\ell}{4})} \cdot p + \frac{1}{(\frac{\ell}{4})} \cdot (1-p) \bigg| 0 \leq f \leq \left\lceil \frac{\delta}{4} \right\rceil - 1 \right\}.
\]

As noted in the updated version of Ligero [AHIV23], Lemma 1 has been improved in recent works. In particular, [BCI+20] Theorem 1.2 provides a result for greater values of \(e\) (up to \(\delta/2\):

**Theorem 3 (BCI+20).** Let \(e\) be a positive integer such that \(e < \delta/2\), with \(\delta = N-\ell\). Suppose \(\Delta(U^*, L^n) \geq e\). Then, for a random \(w^*\) in the row-span of \(U^*\), we have

\[
\Pr[\Delta(w^*, L) \leq e] \leq \frac{N}{|\mathcal{F}|}.
\]

The above result might improve the soundness w.r.t. a malicious prover committing \(f\)-far sharings with \(f \leq \delta/2\) or \(\delta/4 \leq f < \delta/2\), as follows:

- If \(f \geq \delta/2\) and by analogy to the second strategy: According to Theorem 3 (with \(e = \left\lceil \delta/2 \right\rceil - 1\), the probability that a random linear combination is \(f\)-far from a valid sharing with \(f' < \delta/2\) is at most \(\frac{N}{|\mathcal{F}|}\). When this event does not occur, we know that there are at least \(\left\lceil \frac{\delta}{2} \right\rceil\) invalid parties, and so the probability that the malicious prover convinces the verifier corresponds to the probability that the \(\ell\) parties the verifier asks to open are not among those parties. We thus have:

\[
\Pr[\text{accept}] \leq \Pr[f' < \frac{\delta}{2}] + \Pr[\text{accept} | f' \geq \frac{\delta}{2}] \leq \frac{N}{|\mathcal{F}|} + \frac{(N-\left\lfloor \frac{\delta}{4} \right\rfloor)}{(\frac{\delta}{4})}.
\]

- If \(\frac{\delta}{4} \leq f < \frac{\delta}{2}\) and by analogy to the third strategy: The probability that a random linear combination is \(f\)-far from a valid sharing with \(f' < f\) is at most \(\frac{N}{|\mathcal{F}|}\), according to Theorem 3 (with \(e = f - 1\)). Using the same reasoning than in the third case above, we have:

\[
\Pr[\text{accept}] \leq \frac{N}{|\mathcal{F}|} + \frac{(N-f)}{(\frac{\ell}{4})} \cdot p + \frac{1}{(\frac{\ell}{4})} \cdot (1-p).
\]

This analysis refines the above soundness error, which now satisfies

\[
\epsilon \leq \max \left\{ \frac{N}{|\mathcal{F}|} + \frac{(N-\left\lfloor \frac{\delta}{4} \right\rfloor)}{(\frac{\delta}{4})} \right\} \cup \left\{ \frac{f}{|\mathcal{F}|} + \frac{(N-f)}{(\frac{\ell}{4})} \cdot p + \frac{1}{(\frac{\ell}{4})} \cdot (1-p) \bigg| 0 \leq f \leq \left\lceil \frac{\delta}{4} \right\rceil - 1 \right\}
\]

\[
\cup \left\{ \frac{N}{|\mathcal{F}|} + \frac{(N-\left\lceil \frac{\delta}{4} \right\rceil)}{(\frac{\delta}{4})} \cdot p + \frac{1}{(\frac{\ell}{4})} \cdot (1-p) \bigg| \left\lceil \frac{\delta}{4} \right\rceil \leq f \leq \left\lceil \frac{\delta}{2} \right\rceil - 1 \right\}.
\]

In [AHIV17] [AHIV23], the authors provide a simpler expression for the soundness error simply by summing the largest terms in the above formula:

\[
\epsilon \leq \frac{N}{|\mathcal{F}|} + \frac{(N-\left\lfloor \frac{\delta}{4} \right\rfloor)}{(\frac{\delta}{4})} \cdot p + \frac{1}{(\frac{\ell}{4})} \leq \frac{N}{|\mathcal{F}|} + \left(1 - \left\lceil \frac{\delta}{4} \right\rceil \right)^\ell + \left(\frac{\ell}{N}\right)\ell.
\]

Let us mention that, instead of a single random linear combination of the input sharings, the test can involve \(\eta\) independent random linear combinations to amplify the soundness. In that case, the terms \(\frac{1}{|\mathcal{F}|}\) in Lemma 1, Theorem 3, Equation (21) and the above inequality are replaced by terms \(\frac{1}{|\mathcal{F}|^\eta}\) (with numerator unchanged).
We first recall the theorem statement, and then give the proof.

**Theorem 2.** Let $\Pi$ be an MPC protocol of parameters $(N, \ell, s, d_\alpha, d_\beta, p)$ complying to the format of Protocol 3. In particular, $\Pi$ is $\ell$-private in the semi-honest model and of false positive probability $p$. Then, Protocol 5 built from $\Pi$ satisfies the three following properties:

- **Completeness.** A prover $P$ who knows a witness $w$ such that $(x, w) \in R$ and who follows the steps of the protocol always succeeds in convincing the verifier $V$.

- **Soundness.** Suppose that there is an efficient prover $\tilde{P}$ that, on input $x$, convinces the honest verifier $V$ to accept with probability $	ilde{\epsilon} := \Pr[\langle \tilde{P}, V \rangle(x) \to 1] > \epsilon$

where the soundness error $\epsilon$ is defined as

$$
\epsilon := \begin{cases} 
    p + (1 - p) \cdot \binom{\ell}{\ell} & \text{for the TCitH-GGM variant}, \\
    p + (1 - p) \cdot \binom{\ell}{\ell} + \frac{d_{\alpha + 2}}{|F|} & \text{for the TCitH-MT variant}.
\end{cases}
$$

(22)

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

- **Honest-Verifier Zero-Knowledge.** Let the pseudorandom generator PRG used in Protocol 3 be $(t, \epsilon_{\text{PRG}})$-secure and the commitment scheme $\text{Com}$ be $(t, \epsilon_{\text{Com}})$-hiding. There exists an efficient simulator $S$ which, given random challenge $I$ outputs a transcript which is $(t, \epsilon_{\text{PRG}} + \epsilon_{\text{Com}})$-indistinguishable from a real transcript of Protocol 3.

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

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

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 produced by $\tilde{P}$, a single preimage of this commitment/hash can be extracted from $\tilde{P}$. In particular, the commitment $h_1$ corresponds to a single extractable value of the sharing $\llbracket w \rrbracket$.

**Proof of soundness for the TCitH-GGM framework.** Since $\tilde{P}$ is malicious, their first message $h_1$ commits a sharing $\llbracket w \rrbracket$ of an invalid witness $w$, i.e. which does not satisfy $(x, w) \in R$. 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 $\Pi$, 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 $\binom{d_{\alpha}}{\ell} + 1$-special-sound protocol. The probability that the verifier is convinced after the last round is thus $\binom{\ell}{\ell} / \binom{d_{\alpha}}{\ell}$, since $\binom{\ell}{\ell}$ is the size of the challenge space.
Denoting \( FP \) the false positive event, we get that the soundness error satisfies:

\[
\epsilon = \Pr[FP] + \Pr[-FP] \cdot \Pr[Succ \mid -FP] = p + (1 - p) \cdot \frac{\binom{d}{\ell}}{\binom{N}{\ell}}.
\]

We now show how to upper bound \( \Pr[Succ \mid -FP] \) to obtain the above inequality. 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 \((\binom{d}{\ell} + 1)\)-special-sound protocol. Namely, we will show that if \( \binom{d}{\ell} + 1 \) 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 \((\binom{d}{\ell})\) valid transcripts can be obtained from the malicious prover if no false positive event occurs, implying \( \Pr[Succ \mid -FP] \leq \frac{\binom{d}{\ell}}{\binom{N}{\ell}} \).

Let us consider \((\binom{d}{\ell}) + 1\) accepting transcripts \(\{\mathcal{T}_k\}_{k \in [1:(\binom{d}{\ell})+1]}\). Each of them corresponds to some party opening set \( I_k \) (with \( I_k \neq I_{k'} \) for \( k \neq k' \)) and satisfies:

\[
\mathcal{T}_k := (h_{1}, e^{1}, \ldots, h_{t}, e^{t}, h_{t+1}, I_k, \sigma_k := (\text{views}_{I_k}, \text{decom}_{I_k}, \{\alpha^{j}\}_{i \in [1:t]}))
\]

where \(\text{views}_{I_k} = (\text{path}_{I_k}, \Delta w, \Delta \beta^{1}, \ldots, \Delta \beta^{t})\) (the \(\Delta\)'s being omitted if \( I_k = T_0 \)), \(\text{decom}_{I_k} = \text{com}_{I_k} = \text{Com}(\text{seed}_{I_k}; \mu_{I_k})\) and \(S^I_{\alpha} \) denotes some set of cardinality \( |S^I_{\alpha}| = d_{\alpha} + 1 - \ell \) disjoint of \( I_k \) (as defined in Step 7 of Protocol 5). From the revealed \(\{\alpha^{j}\}_{i \in [1:t]}\) and the opened views, one can recover the full sharings \(\{\alpha^{j}\}_{j}\) and the plain broadcast values \(\{\alpha^{j}\}_{j}\). 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, one can always recover the auxiliary values \((\Delta w, \Delta \beta^{1}, \ldots, \Delta \beta^{t})\) from \(\{\mathcal{T}_k\}_k\) since at most one transcript does not contain those values (if \( I_k = T_0 \) for one of the \(\mathcal{T}_k\)). We can then build the Shamir’s polynomials \(P_w, P_{\beta^1}, \ldots, P_{\beta^t}\) using the Equation 5. 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

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

since \(\alpha^j = P_{\alpha^j}(e_i)\), \(\mu^j = P_{\mu^j}(e_i)\) and \(\beta^k = P_{\beta^k}(e_i)\). Since the polynomials in the above relation are of degree at most \(d_{\alpha}\) and since the relations hold for at least \(d_{\alpha} + 1\) evaluation points \(\{e_i\}_{i \in \bigcup_k I_k}\), we have that the relations hold directly on the polynomials: \(P_{\alpha^j} = \Phi^j(P_w, (P_{\beta^k})_{k \leq j})\) for all \( j \). In particular, the relations are true when evaluating the polynomials in \(\omega_1, \ldots, \omega_s\), the evaluation points revealing the packed secrets, which implies

\[
\alpha^j = \Phi^j(w, (\beta^k)_{k \leq j}).
\]

(In the above equation \(\Phi^j\) applies independently on each “slot” of the \( s \)-tuples composing the plain values.) 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 \text{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 \( [w], [\mu], [\beta^1], [\beta^2], \ldots, [\beta^t] \) where each leaf of the tree is a commitment of the \(i\)th shares of all the sharings, for \( i \in [1 : N] \). Since only \( \ell \) leaves are eventually open, the malicious prover may commit some sharings that do not form valid Shamir’s secret sharings of the
expected degrees. The second message \( h'_i \) sent by the prover is the hash digest of the \( \eta \) polynomials \( P_x \). We denote \( E \) the set of the indices of the shares which are consistent with \( P_x \), namely
\[
E := \{ i \in [1 : N] : P_x(e_i) = \Gamma : \lambda (\| w \| \beta^1 \| \beta^2 \| \ldots \| \beta^t) \}_{i} + \| u \}_{i} \}.
\]
Let us define the polynomials \( Q \) and \( P_x \) by interpolation of respectively \( \{ (\| w \| \beta^1 \| \beta^2 \| \ldots \| \beta^t) \}_{i} \}_{i \in E} \) and \( \{ (\| u \}_{i} \}_{i \in E} \), and let us consider the degree-enforcing failure event, defined as
\[
\text{DEF} \equiv "Q \ or \ P_x \ is \ of \ degree \ strictly \ larger \ than \ d_\beta".
\]
According to Theorem 1, we have
\[
\Pr[\text{DEF}] \leq \frac{(d_\beta + 2)^N}{|F|^\eta}.
\]
The soundness error of the protocol then satisfies:
\[
\epsilon = \Pr[\text{DEF}] + \Pr[\neg \text{DEF}] \cdot \Pr[\text{Succ} | \neg \text{DEF}] \leq \Pr[\text{Succ} | \neg \text{DEF}] + \frac{(d_\beta + 2)^N}{|F|^\eta}.
\]
Let us now focus on \( \Pr[\text{Succ} | \neg \text{DEF}] \), namely we consider a valid transcript for which \( \neg \text{DEF} \) occurs. The polynomials \( Q \) and \( P_x \) are of degree at most \( d_\beta \), meaning that the shares \( \{ (\| w \| \beta^1 \| \beta^2 \| \ldots \| \beta^t) \}_{i} \}_{i \in E} \) are valid Shamir’s secret shares of unique values \( w, u, \beta^1, \beta^2, \ldots, \beta^t \). 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 \( \epsilon^1, \ldots, \epsilon^t \) sampled by the verifier is such that an honest execution of the MPC protocol produces the output \text{ACCEPT}. By definition of the considered MPC protocol \( \Pi \), 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 \( \left( \binom{d_\beta}{|F|} + 1 \right) \)-special-sound protocol. The probability that the verifier is convinced after the last round is thus \( \left( \binom{d_\beta}{|F|} \right) / \left( \binom{N}{\ell} \right) \), since \( \binom{N}{\ell} \) is the size of the challenge space.

We get that the soundness error satisfies:
\[
\epsilon \leq \Pr[\text{FP}] + \Pr[\neg \text{FP}] \cdot \Pr[\text{Succ} | \neg \text{FP} \land \neg \text{DEF}] + \Pr[\text{DEF}]
\]
\[
\leq p \cdot (1 - p) \cdot \binom{d_\beta}{|F|} \binom{N}{\ell} + \frac{(d_\beta + 2)^N}{|F|^\eta}.
\]
(23)

We now show how to upper bound \( \Pr[\text{Succ} | \neg \text{FP} \land \neg \text{DEF}] \) to obtain the above inequality. 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 responses form a \( \left( \binom{d_\beta}{|F|} + 1 \right) \)-special-sound protocol. Namely, we will show that if \( \binom{d_\beta}{|F|} + 1 \) 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 \( \binom{d_\beta}{|F|} \) valid transcripts can be obtained from the malicious prover if no false positive event occurs, implying \( \Pr[\text{Succ} | \neg \text{FP} \land \neg \text{DEF}] \leq \binom{d_\beta}{|F|} / \binom{N}{\ell} \).

Let us consider \( \binom{d_\beta}{|F|} + 1 \) accepting transcripts \( \{ T_k \}_{k \in [1 : \binom{d_\beta}{|F|} + 1]} \). Each of them corresponds to some party opening set \( I_k \) (with \( I_k \neq I_{k'} \) for \( k \neq k' \)) and satisfies:
\[
T_k := (\lambda \Gamma, h'_1, \epsilon^1, \ldots, h'_t, \epsilon^t, h'_t+1, I_k, \sigma_k) := (\text{views}_{I_k} \cdot \text{decom}_{I_k}, \{ \| \alpha \}_{i} \}_{i \in [1 : d]})
\]
Prover’s last response

Same for all transcripts
where \( \text{views}_{I_k} = (\{ [w]_i, [\beta^1]_i, [\beta^2]_i, \ldots, [\beta^t]_i \}_{i \in I_k}, \Delta \beta^2, \ldots, \Delta \beta^t) \), \( \text{decom}_{I_k} = \text{path}_{I_k} \) and \( S'_k \) denotes some set of cardinality \( |S'_k| = d_{\alpha, j} + 1 - \ell \) disjoint of \( I_k \) (as defined in Step 7 of Protocol [5]). From these transcript, since \( I_k \neq I_{k'} \) for all \( k \neq k' \), we can retrieve at least \( d_{\alpha} + 1 \geq d_{\beta} + 1 \) shares of the committed sharings \([w], [\beta^1], [\beta^2], \ldots, [\beta^t]\) (which pass the degree-enforcement test, i.e. which are consistent with \( P_\xi \)). 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 \( \{ T_k \}_k \) are valid transcripts, it means that, for all \( i \in \bigcup_k I_k \) and \( j \in [1 : t] \), we have
\[
P_{\alpha, j}(e_i) = P_w(e_i), (P_{\beta^k}(e_i))_{k \leq j}
\]
since \([\alpha^j]_i = P_{\alpha, j}(e_i), [u]_i = P_w(e_i) \) and \([\beta^k]_i = P_{\beta^k}(e_i) \). Since the polynomials in the above relation are of degree at most \( d_{\alpha} \) and since the relations hold for at least \( d_{\alpha} + 1 \) evaluation points \( \{ e_i \}_{i \in I_k} \), we have that the relations hold directly on the polynomials: \( P_{\alpha, j} = \Phi^j(P_w, (P_{\beta^k})_{k \leq j}) \) for all \( j \). In particular, the relations are true when evaluating the polynomials in \( \omega_1, \ldots, \omega_s \), the evaluation points revealing the packed secrets, which implies
\[
\alpha^j = \Phi^j(w, (\beta^k)_{k \leq j}) .
\]
(In the above equation \( \Phi^j \) applies independently on each “slot” of the \( s \)-tuples composing the plain values.) Moreover, since \( \{ T_k \}_k \) 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 \text{Accept} on the witness \( w \). Since we assumed that there were no false positive events \( \neg \text{FP} \), we get that \( w \) must be a valid witness, which concludes the proof. \( \square \)