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

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

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

Abstract. The MPC-in-the-Head paradigm is instrumental in building zero-knowledge proof systems and post-quantum signatures using techniques from secure multi-party computation. In this work, we extend and improve the recently proposed framework of MPC-in-the-Head based on threshold secret sharing, here called *Threshold Computation in the Head*. We first address the two main limitations of this framework, namely the degradation of the communication cost and the constraint on the number of parties. Our tweak of this framework makes it applicable to the previous MPCitH schemes (and in particular post-quantum signature candidates recently submitted to NIST) for which we obtain up to 50% timing improvements without degrading the signature size. Then we extend the TCitH framework to support quadratic (or higher degree) MPC round functions 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 the state of the art and achieves proofs around 17 KB for LWE instances with similar security as Kyber512.

1 Introduction

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

The MPCitH paradigm can be summarized as follows. By emulating an MPC protocol verifying a witness and by opening some (verifier chosen) parties, the prover convinces the verifier they know the witness with soundness error around $1/N$, for N the number of parties involved in the MPC protocol. In the traditional MPCitH approach, the bottleneck in running times comes from the emulation of the N parties. Recent works have shown how this bottleneck can be mitigated. The hypercube technique proposed at Eurocrypt 2023 [AGH+23] improves the "traditional setting" (additive sharing with GGM tree commitments) by decreasing the emulation phase to $1 + \log_2 N$ parties with no extra communication cost. On the other hand, MPCitH based on threshold secret sharing [FR23b], 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, TCitH suffers some communication penalties which is due to the use of Merkle trees in place of GGM trees for the commitments of shares (this overhead typically represents 2 KB for non-interactive arguments with 128-bit security). Moreover, the number N of parties (and hence the achievable soundness) is limited by the size of the finite field on which the witness is defined ($N \leq |\mathbb{F}|$), which is an issue when dealing with small fields.

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 first show how using techniques from [CDI05], we can generate and commit Shamir's secret shares with the communication cost of a GGM tree (as in the traditional approach) while benefiting the low-cost MPC emulation of TCitH. In this TCitH-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. *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 (more compact) vs. Merkle tree (faster verification). We provide a tight soundness analysis for these two variants with respect to the packing and and degree parameters of the MPC protocol.

3. *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. 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 inspired from the proximity test for Reed-Solomon codes proposed in Ligero [AHIV17] but it crucially relies on different parameters to achieve stronger guaranties (*i.e.*, exact degree of committed sharings) which allows us to reach a better soundness.

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]. *(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 1000 users, both schemes have a running time below 10 ms, while achieving sizes around 5 KB for MQ and 9 KB with SD, which greatly improves the current state of the art. 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 symetric 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 with the best previous MPCitH-based scheme. On the Kyber512 (resp. Dilithium2) LWE instance, we achieve 23 KB (resp. 44 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_∞ norm). For custom instances with same security as Kyber512, our proofs can be lowered to 17 KB.

5. *Unifying MPCitH-like proof systems.* 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 variants and/or instanciations of our framework and other MPCitH-like proof systems, namely VOLE-in-the-Head [BBdSG$^+$23] and Ligero [AHIV17, AHIV23]. Our framework thus unifies different MPCitH-like proof systems under the same umbrella, which provides further insights on these techniques and might foster further 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 coeff_j the function mapping a polynomial P to its degree-j coefficient, so that $P(X) = \sum_{j=0}^{\deg P} \mathsf{coeff}_j(P) X^j$. We call $P = (P_1, \ldots, P_n) \in (\mathbb{F}[X])^n$ a *vector polynomial.* The coeff_j function extends to vector polynomials: for any $P \in (\mathbb{F}[X])^n$, $\mathsf{coeff}_j(P) \in \mathbb{F}^n$ is naturally defined as the tuple of the degree-j coefficients $(\mathsf{coeff}_j(P_1), \ldots, \mathsf{coeff}_j(P_n))$.

2.1 Secret Sharing

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

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

The techniques proposed in the paper rely on *Shamir's secret sharing* [Sha79] and on its packed version [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[3] *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*

$$[\![v]\!] = ([\![v]\!]_1, \ldots, [\![v]\!]_N) \in \mathbb{F}^N \quad s.t. \quad [\![v]\!]_i := P_v(e_i) \quad \forall i \in [1:N] \ ,$$

for some polynomial P_v of degree $\leq d$ 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 $[\![v]\!]_i$ 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 ℓ shares of a fresh sharing $[\![v]\!]$ 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 $[\![v]\!]_J$ works by interpolating the polynomial P_v from the $d + 1$ evaluation points $[\![v]\!]_J = (P_v(e_i))_{i \in J}$ and outputing 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 (ℓ) 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*

$$[\![v]\!] = \begin{pmatrix} [\![(v_1, \ \ldots, \ v_s)]\!] \\ [\![(v_{s+1}, \ldots, v_{2s})]\!] \\ \vdots \end{pmatrix} \in (\mathbb{F}^N)^{|v|_s} \ , \tag{1}$$

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 $(v_{(i-1)s+1}, \ldots, v_{is})$.

Linear and multiplicative homomorphisms. The (packed) Shamir's secret sharing enjoys linear and multiplicative homomorphisms. Consider the packed sharings $[\![v_1]\!], [\![v_2]\!]$ of tuples $v_1, v_2 \in \mathbb{F}^s$. Then for any scalars $a_1, a_2 \in \mathbb{F}$ 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 $[\![v_1]\!], [\![v_2]\!] \in (\mathbb{F}^N)^t$ encoding packed tuples $v_1, v_2 \in \mathbb{F}^{s \cdot t}$, we denote by $[\![v_3]\!] = \langle [\![v_1]\!], [\![v_2]\!] \rangle$ 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 $v_3 \in \mathbb{F}_s$ for which the i^{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 ith 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

[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.

first recall the general principle of this paradigm before introducing a formal model for the underlying MPC protocols and their transformation into zero-knowledge proofs.[4]

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

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

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

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

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

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

2.3 General Model for MPCitH-Friendly MPC Protocols

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

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

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

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

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

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

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

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

2.4 MPCitH Transform based on Additive Sharing and GGM Trees

Any MPC protocol complying with the above description gives rise to a practical short-communication zero-knowledge proof in the MPCitH paradigm. The resulting zero-knowledge proof is described in Protocol 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.

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

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

Protocol 1: Zero-knowledge protocol - Application of the MPCitH principle to the general MPC protocol of [FR22].

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

$$\frac{1}{N} + \left(1 - \frac{1}{N}\right) \cdot p \ . \tag{2}$$

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

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

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

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

Some coordinates of the β^j might be uniformly distributed over $\mathbb{F}$ (remember that the β^j are tuples of $\mathbb{F}$ elements). We denote β^{unif} the sub-tuple composed of those uniform coordinates. In this context, the last share $\llbracket \beta^{\mathrm{unif}} \rrbracket_N$ can be built as $\llbracket \beta^{\mathrm{unif}} \rrbracket_N \leftarrow \mathrm{PRG}(\mathsf{seed}_N)$ so that a seed seed_N can be committed in com_N^1 (instead of committing $\llbracket \beta^{\mathrm{unif}} \rrbracket_N$). This way the prover can save communication by revealing seed_N instead of $\llbracket \beta^{\mathrm{unif}} \rrbracket_N$ whenever the latter is larger;

– the messages $\llbracket \alpha^1 \rrbracket_{i^*}, \ldots, \llbracket \alpha^t \rrbracket_{i^*}$ broadcasted by the unopened party. Let us stress that one can sometimes save communication by sending only some elements of $\llbracket \alpha^1 \rrbracket_{i^*}, \ldots, \llbracket \alpha^t \rrbracket_{i^*}$ and use the relation $g(\alpha^1, \ldots, \alpha^t) = 0$ to recover the missing ones;

– the hash digests $h_1, \ldots, h_{t+1}$ and the unopened commitments $\mathsf{com}_{i^*}^1, \ldots, \mathsf{com}_{i^*}^t$ (as explained above, we have $\mathsf{com}_{i^*}^j = \emptyset$ for $j > 1$ if $i^* \neq N$).

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

2.5 Threshold Computation in the Head: Original Framework

In [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 ℓ-privacy from the sharings, this transformation is straightforward. Then, the TCitH transform compiles this MPC protocol into the following proof of knowledge:

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

Since only a small number of commitments need to be open in the TCitH framework, one relies on a Merkle tree instead of a GGM tree. Namely, the commitment h_1 is computed as the root of a Merkle tree with leaves com_i. Then, while opening the commitments, the prover further sends the authentication paths of the opened leaves $\{\mathsf{com}_i\}_{i\in I}$ to allow the verifier to recompute and check h_1. In [FR23b], the soundness error of the obtained proof system is shown to be

$$\frac{1}{\binom{N}{\ell}} + p \cdot \frac{\ell(N-\ell)}{\ell+1} \ , \tag{3}$$

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

The hypercube technique introduced in [AGH+23] already reduces the cost of emulating the MPC protocol to $1 + \log_2 N$ parties (instead of N) without impacting the communication cost. On the other hand, the original TCitH framework [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 2λ bits while a GGM sibling path is made of $\log N$ seeds of λ bits.
- There is a soundness loss since a malicious prover can commit an invalid sharing (see Equation (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 Pseudorandom Sharing and 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, ℓ, N) with $s = 1$. The general case of is described later in

Section 4.2. So in the context of this section, whenever a tuple $w \in \mathbb{F}^{|w|}$ is shared, the sharing $[\![w]\!]$ is to be interpreted as a tuple of the sharings $([\![w_i]\!])_{1 \le i \le |w|}$ of the coordinates $w_i \in \mathbb{F}$. Without loss of generality, we shall assume that the evaluation point ω_1 revealing the secret coordinate is fixed to $\omega_1 = 0$, 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 $\binom{N}{\ell}$ shares, each labelled by a different set from $\{T \subset [1:N], |T| = \ell\}$:

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

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

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

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

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

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

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

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

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

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

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

$$[\![w]\!]_i := \sum_{T \in \mathcal{S}_\ell^N, i \notin T} s_T \cdot P_T(e_i) + \begin{cases} \Delta w \cdot P_{T_0}(e_i) & \text{if } i \notin T_0, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

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

$$P_w(X) \ := \ \Delta w \cdot P_{T_0}(X) + \sum_{T \in \mathcal{S}_\ell^N} s_T \cdot P_T(X) \ , \tag{5}$$

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

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

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

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

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

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

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

Protocol description. We rely on the same zero-knowledge protocol as in the original TCitH framework tweaked with the above sharing generation. Instead of committing $\{[\![w]\!]_i\}_i$, the prover commits $\{\mathsf{seed}_T\}_{T \in \mathcal{S}_\ell^N}$ and Δw. To open a party i, the prover then needs to reveal all the seeds $\{\mathsf{seed}_T\}_{T \in \mathcal{S}_\ell^N, i \notin T}$ and Δw (if $i \notin T_0$), from which the verifier can recompute the share $[\![w]\!]_i$ thanks to Equation (4). In practice, the prover should reveal a subset I of ℓ parties, so they will reveal

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

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

Protocol 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 ψ^j are the functions defining the hints.[6]

Computing on polynomial coefficients. As briefly mentioned in [FR22], instead of emulating a subset of $\ell + 1$ parties (*i.e.* applying the MPC computation for $\ell + 1$ shares), the prover can directly emulate the MPC protocol *on the $\ell + 1$ coefficients of the polynomial P_w* (underlying the Shamir's secret sharing $[\![w]\!]$). Since the constant term of the polynomial corresponds to the plain secret value, emulating the MPC protocol on the constant coefficient is equivalent to computing the function underlying the MPC protocol (the function f in Equation (6)) on the plain input witness. Such "emulation" is often cheaper than a party emulation, since, in the MPCitH context, the prover already know some expected values α^j (which satisfy $g(\alpha^1, \ldots, \alpha^t) =$ ACCEPT) and can thus save some computation.

[6] A formal description of the general MPC protocol is also provided in Section 4.1 (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 Φ^j round functions are made of inner functions $\varphi^{j,k}$ which are linear.

1. The prover samples a root seed $\mathsf{rseed} \in \{0,1\}^\lambda$ and expands it through a GGM tree to obtain $\binom{N}{\ell}$ leaf seeds $\{\mathsf{seed}_T\}_{T\in\mathcal{S}_\ell^N}$.
2. The prover expands each leaf seeds, $s_T^0 \leftarrow \mathrm{PRG}(\mathsf{seed}_T)$ for all T, builds the auxiliary value $\Delta w := w - \sum_T s_T^0$, and computes
$$P_w \leftarrow \mathsf{GenerateSharingPolynomial}(\Delta w, \{s_T^0\}_{T\in\mathcal{S}_\ell^N}).$$
3. The prover emulates "in her head" the MPC protocol.
For $j = 1$ to t:
   (a) the prover computes
   $$\beta^j = \psi^j(w, \{\varepsilon^k\}_{k<j}; r^j)$$
   using fresh uniform random tape r^j.
   (b) the prover expands more randomness $\{s_T^j\}_{T\in\mathcal{S}_\ell^N}$ from the leaf seeds (such that $|s_T^j| = |\beta^j|$ for all T), builds the auxiliary value $\Delta\beta^j := \beta^j - \sum_T s_T^j$, and computes
   $$P_{\beta^j} \leftarrow \mathsf{GenerateSharingPolynomial}(\Delta\beta^j, \{s_T^j\}_{T\in\mathcal{S}_\ell^N}).$$
   (c) the prover expands some commitment randomness ρ_T^j from the leaf seeds (such that $|\rho_T^j| = \lambda$ for all T) and computes the commitments
   $$\mathsf{com}_T^j := \begin{cases} \mathrm{Com}(\mathsf{seed}_T; \rho_T^j) & \text{if } j = 1 \text{ and } T \neq T_0 \\ \mathrm{Com}(\mathsf{seed}_T, \Delta w, \Delta\beta^j; \rho_T^j) & \text{if } j = 1 \text{ and } T = T_0 \\ \emptyset & \text{if } j > 1 \text{ and } T \neq T_0 \\ \mathrm{Com}(\Delta\beta^j; \rho_T^j) & \text{if } j > 1 \text{ and } T = T_0 \end{cases}$$
   for all $T \in \mathcal{S}_\ell^N$.
   (d) the prover sends
   $$h_j := \begin{cases} \mathrm{Hash}(\{\mathsf{com}_T^1\}_T) & \text{if } j = 1 \\ \mathrm{Hash}(\{\mathsf{com}_T^j\}_T, P_{\alpha^{j-1}}) & \text{if } j > 1 \end{cases}$$
   to the verifier;
   (e) the verifier picks at random a challenge ε^j and sends it to the prover;
   (f) the prover compute the plain broadcast
   $$\alpha^j := \mathsf{coeff}_0(P_{\alpha^j}) = \Phi^j\big(w, (\beta^k)_{k\le j}\big).$$
   (g) the prover computes, for $i \in [1:\ell]$,
   $$\mathsf{coeff}_i(P_{\alpha^j}) := \Phi^j\big(\mathsf{coeff}_i(P_w), (\mathsf{coeff}_i(P_{\beta^k}))_{k\le j}\big).$$

   The prover further computes $h_{t+1} := \mathrm{Hash}(P_{\alpha^t})$ and sends it to the verifier.
4. The verifier picks at random a subset $I \subset [1:N]$ of ℓ parties (*i.e.* $|I| = \ell$) and sends it to the prover.
5. The prover reveals the views of all the parties in I, namely they send the sibling path of seed_I in the GGM tree to the verifier, together with $\Delta w, \{\Delta\beta^j\}_{j\in[1:t]}$ when $I \neq T_0$. The prover further sends the digests of the unopened commitments $\{\mathsf{com}_I^j\}_{j\in[1:t]}$ and the plain broadcast values $\{\alpha^j\}_{j\in[1:t]}$.
6. The verifier recomputes the commitments com_T^j for $T \neq T_0$ and $j \in [1:t]$ from the sibling path and the auxiliary values (in the same way as the prover). They expand all the randomness $(s_T^0, s_T^1, \dots, s_T^t)_{T\neq I}$ from seeds and build the share of the open parties: for all $i \in I$,
$$\begin{cases} [\![w]\!]_i = \mathsf{GeneratePartyShare}_i(\{s_T^0\}_{T:i\notin T}, \Delta w) \\ [\![\beta^j]\!]_i = \mathsf{GeneratePartyShare}_i(\{s_T^j\}_{T:i\notin T}, \Delta\beta^j) \text{ for all } j \in [1:t] \end{cases}$$
where Δw and $\{\Delta\beta^j\}_j$ are omitted if not provided by the prover. The verifier can emulate the MPC protocol on the open parties: for all $i \in I$ and $j \in [1:t]$,
$$[\![\alpha^j]\!]_i := \Phi^j\big([\![w]\!]_i, ([\![\beta^k]\!]_i)_{k\le j}\big)$$
Finally, the verifier can recompute the Shamir's polynomials P_{α^j} of all $\{[\![\alpha^j]\!]\}_j$: for all $j \in [1:t]$,
$$P_{\alpha^j} = \mathsf{RecomputeSharing}_I(\alpha^j, ([\![\alpha^j]\!]_i)_{i\in I}).$$
7. The verifier accepts if and only if:
   (a) the views of the opened parties are consistent with each other, with the committed input shares and with the hash digest of the broadcast messages, *i.e.* for $j = 1$ to $t+1$,
   $$h_j \overset{?}{=} \begin{cases} \mathrm{Hash}(\{\mathsf{com}_T^1\}_T) & \text{if } j = 1 \\ \mathrm{Hash}(\{\mathsf{com}_T^j\}_T, P_{\alpha^{j-1}}) & \text{if } 2 \le j \le t \\ \mathrm{Hash}(P_{\alpha^t}) & \text{if } j = t+1; \end{cases}$$
   (b) the output of the opened parties are ACCEPT, *i.e.*
   $$g(\alpha^1, \dots, \alpha^t) \overset{?}{=} 0 \ .$$

Protocol 2: Zero-knowledge protocol: application of the TCitH framework with GGM tree to the general MPC protocol of [FR22].

Protocol routines. Protocol 2 is based on the following three routines that deal with sharings:

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

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

– $\mathsf{GeneratePartyShare}_i$ (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 $\mathsf{GeneratePartyShare}_i((s_T)_{T:i\notin T}, \Delta x)$ outputs

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

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

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

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

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

Soundness and zero-knowledge analysis. Let us analyze the soundness of Protocol 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-ℓ polynomial). This degree of freedom impacts the soundness of the scheme: the false positive probability p is scaled by a factor $\ell(N-\ell)/(\ell+1)$ in the soundness error (see Equation (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 $\binom{N}{\ell}$ seeds $\{\mathsf{seed}_T\}_T$ with an auxiliary value Δw. These values always define a valid Shamir's secret sharing with underlying polynomial

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

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

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

which for $\ell = 1$ matches the soundness error of the MPCitH framework with additive sharing (see Equation (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 1 in the next section (for an extension of the TCitH framework).

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

Performances. Let us analyze the communication cost of Protocol 2. The prover sends

- $t+1$ hash digests $h_1, \dots, h_{t+1}$, which cost $(t+1) \cdot 2\lambda$ bits;
- the sibling path of seed_I in a seed tree with $\binom{N}{\ell}$, which costs $\lambda \cdot \log_2 \binom{N}{\ell}$ bits;
- the auxiliary values $(\Delta x, \{\Delta\beta^j\}_{j\in[1:t]})$ when $I \neq T_0$;
- the plain broadcast values $\alpha^1, \dots, \alpha^t$;
- some commitment digests $\{\mathsf{com}_I^j\}_{j\in[1:t]}$, which cost $t \cdot 2\lambda$ bits when $I = T_0$ and 2λ bits otherwise (since com_I^j is $\emptyset$ for $j > 0$ and $I \neq T_0$).

We obtain a total communication cost of

- when $I \neq T_0$,

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

- when $I = T_0$,

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

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

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

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

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

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

– *Communication cost.* We can assume that

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

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

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

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

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

3.2 Lifting in a Field Extension

As explained previously, the TCitH framework suffers the constraint that the number N of parties should be smaller than the field size: $N \le |\mathbb{F}|$ (or $N \le |\mathbb{F}|+1$ in some cases) as long as $\ell < N-1$ (see Lemma 1 in [FR23b]).[7] This limitation is an issue when dealing with statements defined over small fields (and in particular the binary field $\mathbb{F}_2$). A natural idea to overcome this limitation is to lift the sharing in a field extension.

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

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

but we compute the shares $[\![w]\!]_i$ 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 $\{[\![w]\!]_i\}_i$ live in $\mathbb{K}^{|w|}$ instead of $\mathbb{F}^{|w|}$. Let us stress that the security properties still hold using this tweak:

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

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

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

- The cost of running the plain protocol (Step 3(f) of Protocol 2) remains unchanged, since the plain values still live in $\mathbb{F}$;
- The cost of running the MPC protocol on the ℓ other coefficients of the Shamir's polynomials (Step 3(g) of Protocol 2) is bigger. It is exactly η times bigger as we can decompose this computation into η smaller independent computations living in the base field. Indeed, by denoting $A_0, \ldots, A_j$ the matrices and b the vector underlying the definition of φ^j, which is $\varphi^j : (v_0, \ldots, v_j) \mapsto A_0 \cdot v_0 + \cdots + A_j \cdot v_j + b$ and by denoting $x_{|d}$ the $\mathbb{F}$-coordinate of $x \in \mathbb{K}$ corresponding to the coefficient of the term δ^{d-1} when decomposing $x \in \mathbb{K}$ in the $\mathbb{F}$-basis $(1, \delta, \ldots, \delta^{\eta-1})$, we have:

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

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

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

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

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

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

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

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

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

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

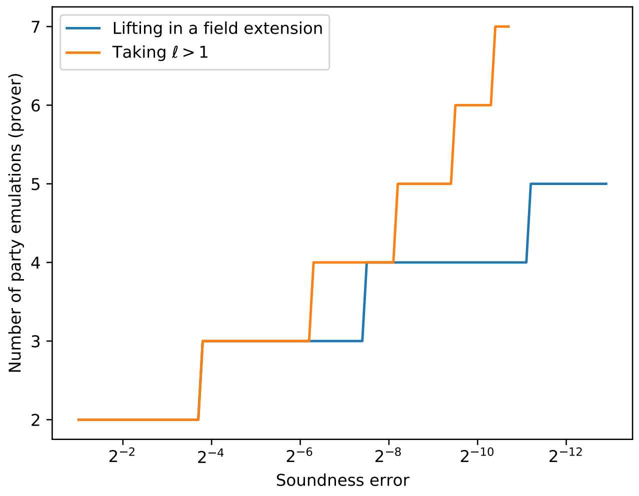


Fig. 1: Emulation cost when working on $\mathbb{F}_{13}$ using lifting and using $\ell > 1$.

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

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

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

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

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

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

Remark 4. Let us remark that the extreme case of $\eta = \log_2 N$ and $N_1 = \ldots = N_\eta = 2$ corresponds to the standard additive-sharing MPCitH framework with hypercube optimization from [AGH+23] (since for $(\ell, N) = (1, 2)$ a Shamir's secret sharing is equivalent to an additive sharing). Whenever the base field is $\mathbb{F}_2$, this gives the best we can hope for with our approach. Whenever the field is larger, our "pseudorandom sharing + lifting" TCitH framework always brings a better trade-off.

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

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

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

3.3 Global Comparison

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

- the traditional additive-sharing MPCitH framework [KKW18, BN20], where the prover emulates all the N parties in their head;
- the additive-sharing MPCitH framework with hypercube optimization [AGH+23], where the prover only emulates $1 + \log_2 N$ parties;
- the original TCitH framework [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.

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

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

We can make the following observations:

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

Remark 6. Let us mention that the framework TCitH with GGM trees is compatible with the principle of MPCitH with rejection proposed in [FMRV22].

3.4 Application to NIST Post-Quantum Signature Candidates

In the recent NIST call for additional post-quantum signature schemes, 7 submissions fit the MPCitH framework:[8] AIMer [KHS$^+$22, CCH$^+$23], Biscuit [BKPV23], MIRA [ABC$^+$23, ABB$^+$23d], MiRitH [ARZV23, ABB$^+$23b], MQOM [FR23a], RYDE [BCF$^+$23, ABB$^+$23c] and SDitH [FJR22, AFG$^+$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.

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

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

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

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

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

4 Extended TCitH Framework and Applications

This section presents our extended TCitH framework. We start by formalizing the MPC model for our extended framework (as an adaptation of the model from [FR23b]), then we describe our extended TCitH proof system in two variants (Merkle tree vs. GGM tree), and finally give several applications and implementation results.

4.1 MPC Model

We consider an MPC protocol that performs its computation on a base finite field $\mathbb{F}$. 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 $\mathbb{F}$. The size of such a tuple v shall be denoted $|v|$ so that $v \in \mathbb{F}^{|v|}$. 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 $[\![v]\!]$ of $v \in \mathbb{F}^{|v|}$ is a vector of packed sharings: $[\![v]\!] = ([\![(v_1, \dots, v_s)]\!], [\![(v_{s+1}, \dots, v_{2s})]\!], \dots) \in (\mathbb{F}^N)^{|v|_s}$ 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

$$f(w, \boldsymbol{\varepsilon}, \boldsymbol{\beta}) = \begin{cases} \text{Accept} & \text{if } g(\boldsymbol{\alpha}) = 0, \\ \text{Reject} & \text{otherwise,} \end{cases} \tag{6}$$

with $\boldsymbol{\varepsilon} = (\varepsilon^1, \dots, \varepsilon^t)$ the random values from a randomness oracle $\mathcal{O}_R$, $\boldsymbol{\beta} = (\beta^1, \dots, \beta^t)$ the hints from a hint oracle $\mathcal{O}_H$, $\boldsymbol{\alpha} = (\alpha^1, \dots, \alpha^t) := \Phi(w, \boldsymbol{\varepsilon}, \boldsymbol{\beta})$ the broadcasted and publicly recomputed values (for some function Φ 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 $v \in \mathbb{F}^s$ 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 $|v|_s > 1$, in which case $[\![v]\!]$ is a $|v|_s$-tuple of packed sharings.
2. The round computation functions $\Phi^1, \ldots, \Phi^t$ (which are used to compute the broadcast values $\alpha^1, \ldots, \alpha^t$) are not restricted to be $\mathbb{F}$-linear but can be polynomial functions of higher degrees.

The latter difference implies that the sharings computed by the parties during the protocol can be of higher degrees than $\ell + s - 1$ (the degree of the input witness sharing). In the following, we denote
$$\deg([\![v]\!]) = \deg(P_v)$$
the degree of the polynomial P_v underlying a Shamir's secret sharing $[\![v]\!]$, also called the degree of $[\![v]\!]$. While the input sharing of the protocol is a fresh degree-$(\ell + s - 1)$ sharing, the computation of non-linear round functions φ 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, ℓ, N)-packed Shamir's secret sharing (*i.e.* a sharing of degree $\ell + s - 1$) $[\![w]\!]$ of the witness w (one share $[\![w]\!]_i$ 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) $\varepsilon \in \mathbb{F}^{|\varepsilon|}$ from a randomness oracle $\mathcal{O}_R$. When calling this oracle, all the parties get the same random value ε. Upon application of the TCitH transform, these random values are provided by the verifier as challenges.

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

– **Computing & broadcasting:** The parties compute $[\![\alpha]\!] := \varphi([\![w]\!], [\![\boldsymbol{\beta}]\!], [\![\theta]\!])$, which means that they locally compute
$$[\![\alpha]\!]_i := \varphi([\![w]\!]_i, [\![\boldsymbol{\beta}]\!]_i, [\![\theta]\!]_i) \ , \ \forall i \in [1:N]$$
where $[\![\boldsymbol{\beta}]\!] = ([\![\boldsymbol{\beta^1}]\!], [\![\boldsymbol{\beta^2}]\!], \ldots)$ is made of the previous outputs of $\mathcal{O}_H$ and $[\![\theta]\!]$ is a fixed (publicly known) sharing. The parties then broadcast $[\![\alpha]\!]$ and publicly recompute α.

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

Example 1. A broadcast value could be computed as
$$[\![\alpha]\!] := [\![w_1]\!] \cdot [\![w_2]\!]$$

with $w_1, w_2 \in \mathbb{F}^s$ two size-s coordinate packs of the witness. Here the function φ is simply the product of these two coordinate packs: α will be equal to $w_1 \circ w_2$, where $\circ$ is the coordinate-wise multiplication. This sharing product is computed sharewisely: $[\![\alpha]\!]_i := [\![w_1]\!]_i \cdot [\![w_2]\!]_i$ for every i. The obtained sharing $[\![\alpha]\!]$ has underlying polynomial $P_\alpha := P_{w_1} \cdot P_{w_2}$, with P_{w_1}, P_{w_2} the polynomials underlying the sharings $[\![w_1]\!], [\![w_2]\!]$. We hence have $\deg([\![\alpha]\!]) = 2(\ell + s - 1)$. Upon application of the TCitH framework, the prover must communicate $2(\ell + s - 1) + 1$ shares of $[\![\alpha]\!]$ to allow the verifier to reconstruct the full sharing.

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

General protocol description. We consider two notions of rounds in our MPC model. The MPC protocol is composed of one or several *outer rounds*. The first outer round starts with the parties receiving the input sharing and possibly a first sharing from the hint oracle. It is then composed of a call to the randomness oracle and one or several *inner rounds* of computing and broadcasting. Then the protocol either finishes with the computation of g, or the parties call the hint oracle. In the latter case, a new outer round begins with a call to the randomness oracle followed by one or several inner rounds. In the MPCitH or TCitH paradigm, a new outer round begins each time the prover needs to commit a new sharing (*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 $[\![\alpha^{j,k}]\!] = \varphi^{j,k}([\![w]\!], [\![\beta^1]\!], \ldots, [\![\beta^j]\!], [\![\theta^{j,k}]\!])$ for $k \in [1 : t_j^{(in)}]$. This enables each function $\varphi^{j,k}$ to depend on previously recomputed values $\alpha^{j,1}, \ldots, \alpha^{j,k-1}$. This notably gives a way to compute or verify non-linear (high degree) functions although the $\varphi^{j,k}$ functions might be linear (or low degree) – see for instance the product-check protocol of [BN20]. We shall denote by Φ^j the global iterative functions obtained from those $t_j^{(in)}$ inner rounds:

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

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

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

False positive probability. The functionality computed by the protocol deterministically depends on the broadcasted values $\boldsymbol{\alpha}$ (through the function g), which in turn deterministically depend on the input witness w, the sampled random values $\boldsymbol{\varepsilon}$, and the hints $\boldsymbol{\beta}$. It is formally given by the function f from Equation (6), with $\boldsymbol{\alpha} = \Phi(w, \boldsymbol{\varepsilon}, \boldsymbol{\beta})$ where Φ is the deterministic function mapping $(w, \boldsymbol{\varepsilon}, \boldsymbol{\beta})$ to $\boldsymbol{\alpha}$ (defined by the coordinate functions $\Phi^1, \ldots, \Phi^t$). This function f aims at checking the validity of a witness w for a statement x with respect to some relation $\mathcal{R}$, namely checking that $(x, w) \in \mathcal{R}$. As in the MPC model of [FR23b], we restrict our model to MPC protocols for which the function f satisfies the following properties:

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

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

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

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

1. The parties take as input an (s, ℓ, N)-packed Shamir's secret sharing $[\![w]\!]$.
2. For $j = 1$ to t (outer rounds):
   (a) For some function ψ^j and some sharing degree d_{β_j}, the parties get a fresh degree-d_{β_j} s-packed sharing $[\![\beta^j]\!]$ from the hint oracle $\mathcal{O}_H$, such that
   $$\beta^j \leftarrow \psi^j(w, \varepsilon^1, \ldots, \varepsilon^{j-1}; r^j)$$
   for a uniform random tape r^j. (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_{β_j}. In case $d_{\beta_j} = s + \ell - 1$ as for the input sharings, the size of the randomness is ℓ and $[\![\beta^j]\!]$ is a fresh (s, ℓ, N)-packed sharing.)

   (b) The parties get a common random ε^j from the oracle $\mathcal{O}_R$.

   (c) Inner rounds: The parties locally compute and broadcast
   $$[\![\alpha^j]\!] := \Phi^j([\![w]\!], [\![\beta^1]\!], \ldots, [\![\beta^j]\!])$$
   and publicly recompute α^j.
   This step is detailed in Protocol 4.
3. The parties finally accept if $g(\alpha^1, \ldots, \alpha^t) = 0$ and reject otherwise.

Protocol 3: General MPC protocol for extended TCitH framework.

(c) For $k = 1$ to $t_j^{(in)}$ (inner rounds):
   – For some $\mathbb{F}$-polynomial function $\varphi^{j,k}$, the parties compute:
   $$[\![\alpha^{j,k}]\!] := \varphi^{j,k}([\![w]\!], [\![\beta^1]\!], \ldots, [\![\beta^j]\!], [\![\theta^{j,k}]\!])$$
   where $[\![\theta^{j,k}]\!]$ is some fixed sharing.
   – The parties broadcast $[\![\alpha^{j,k}]\!]$ and publicly reconstruct α^j.

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

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

Protocol 4: General MPC protocol: inner rounds.

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

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 prositive 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.

4.2 Extended TCitH Framework

We describe hereafter our extended framework of Threshold Computation in the Head (TCitH). The main difference with the original framework is the support of 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 Merkle tree (TCitH-MT) as the original framework, and TCitH with GGM tree (TCitH-GGM) as presented in Section 3.

Tweaking hint commitments. The proof system described in the original TCitH framework [FR23b] 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 $[\![\bar\beta^2]\!], \dots, [\![\bar\beta^t]\!]$ of uniformly random values $\bar\beta^2 \in \mathbb{F}^{|\beta^2|}, \dots, \bar\beta^t \in \mathbb{F}^{|\beta^t|}$. 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^{th} hint β^j, the prover just needs to compute a *hint correction* $\Delta\beta^j$ as $\Delta\beta^j := \beta^j - \bar\beta^j$ and they can then deduce a sharing $[\![\beta^j]\!]$ of β^j using

$$[\![\beta^j]\!] \leftarrow [\![\bar\beta^j]\!] + [\![\Delta\beta^j]\!] \ , \tag{7}$$

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 $i \in [1:s]$.[9]

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 $[\![\bar\beta^2]\!], \dots, [\![\bar\beta^t]\!]$.
- To obtain sound proof in the MPCitH-MT variant, we will need what we call a *degree-enforcing commitment scheme* to make sure that the committed sharings are of the right degrees. This requires an additional challenge-response round for each sharing commitment. Using the above tweak, this additional round is performed a single time (after the initial Merkle tree commitment) instead of t.

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

1. The prover generates and commits the witness sharing $[\![w]\!]$, a first hint $[\![\beta^1]\!]$ and $t-1$ random sharings $[\![\bar\beta^2]\!] \dots, [\![\bar\beta^t]\!]$; In the TCitH-MT variant, an additional degree-enforcement round of challenge and response is performed (see details below);
2. The verifier generates the randomness ε^1 as challenge;
3. The prover runs the inner rounds of computing and broadcasting *in their head* and commits the broadcast shares $[\![\alpha^1]\!]$ to the verifier;
4. For each j from 2 to t:
   (a) The prover generates and commits the hint correction $\Delta\beta^j$;
   (b) The verifier generates the randomness ε^j as challenge;

[9] Here, whenever $|\beta^j| > s$, the sharings $[\![\beta^j]\!]$, $[\![\bar\beta^j]\!]$ and $[\![\Delta\beta^j]\!]$ are vector sharings and $P_{\Delta\beta^j}$ is a vector polynomial in the sense of Definition 1. Then, $P_{\Delta\beta^j}(\omega_i) = \Delta\beta^j_i$ is to be interpreted as the vector composed of the ith coordinates of the packs composing $\Delta\beta^j$.

(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 $\varDelta\beta^2, \ldots, \varDelta\beta^t$), the sharings $[\![\alpha^1]\!]$, …, $[\![\alpha^t]\!]$.
7. The verifier checks:
   - the commitments of the open shares $[\![w]\!]_I, [\![\beta^1]\!]_I, \{[\![\bar{\beta}^j]\!]_I, \varDelta\beta^j\}_{j\geq 2}$ and of the broadcast sharing $[\![\alpha^1]\!]$, …, $[\![\alpha^t]\!]$;
   - the correct computation of the shares $[\![\alpha]\!]_I$ from $[\![w]\!]_I$ (and $[\![\beta^1]\!]_I$, …, $[\![\beta^t]\!]_I$);
   - that $g(\alpha^1, \ldots, \alpha^t) = 0$ (*i.e.* that the protocol accepts).

The generation and commitment of shares in Step 1 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 $[\![\alpha^j]\!]$ is done by hashing the $d_{\alpha_j}+1$ first shares and sending the obtained hash $h_j = \text{Hash}([\![\alpha^j]\!]_{[1:d_{\alpha_j}+1]})$ to the verifier, where $d_{\alpha_j} = \deg([\![\alpha^j]\!])$. Then, in Step 6, the opening of $[\![\alpha^j]\!]$ simply consists in revealing the shares $[\![\alpha^j]\!]_S$ for some set S such that $|S| = d_{\alpha_j} + 1 - \ell$ and $S \cap I = \emptyset$. In Step 7, the verifier recomputes the shares $[\![\alpha^j]\!]_I$ from $[\![w]\!]_I$ (and $[\![\beta^1]\!]_I$, …, $[\![\beta^t]\!]_I$), then reconstructs the shares $[\![\alpha^j]\!]_{[1:d_{\alpha_j}+1]}$ from the shares $[\![\alpha^j]\!]_{I\cup S}$ to finally check the correctness of the hash h_j. This process checks at the same time the correct computation of the shares $[\![\alpha]\!]_I$ and the commitment of the sharing $[\![\alpha^j]\!]$.

Degree-enforcing commitment scheme (TCitH-MT). As explained in Section 3.1 (see the "Analysis" paragraph), one advantage of the TCitH-GGM framework is to enforce the commitment of a valid Shamir's secret sharing (of a possibly incorrect witness), while in the TCitH-MT framework a malicious prover might commit an invalid sharing, *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 hereafter.

We describe hereafter a way to constrain the prover to commit a valid packed Shamir's secret sharing of the witness (*i.e.*, corresponding to a sharing of degree $s+\ell-1$). Our technique uses the idea of the test of interleaved linear codes (a.k.a. proximity test) for Reed-Solomon codes proposed in Ligero [AHIV17]. However, we use different parameters which crucially allows us to ensure a stronger soundness result. In Ligero (improved with subsequent analysis from [BCI+20]), the test ensures that committed codewords have a distance lower than $\delta/2$ from genuine codewords (where δ is the minimum distance of the underlying code) with possibly non-negligible soundness error. In our context, this translates to ensuring that a committed sharing supposed to be of degree d has a distance lower than $(N-d)/2$ (i.e., less than $(N-d)/2$ non-equal evaluations) to some degree-d sharing. Instead, we ensure that the committed sharings are exactly of the expected degree (which can be $d = s+\ell-1$ or larger), with a tunable soundness error (which can be made negligible).

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

$$[\![\xi]\!] = \varGamma \cdot [\![w]\!] + [\![u]\!] \ , \tag{8}$$

namely the sharing defined as $[\![\xi]\!]_i = \varGamma \cdot [\![w]\!]_i + [\![u]\!]_i$ for all $i \in [1:N]$, where $[\![w]\!]_i \in \mathbb{F}^{|w|_s}$ is the vector composed of the ith share of each packed sharing composing $[\![w]\!]$ and $[\![u]\!]_i \in \mathbb{F}^{\eta}$ is the vector composed of the ith share of each packed sharing composing $[\![u]\!]$. The prover commits $[\![\xi]\!]$ to the verifier by sending the hash value of the underlying vector polynomial P_ξ. The sharing $[\![\xi]\!]$ will be later revealed to the verifier which can then check that $[\![\xi]\!]$ is of right degree $s + \ell - 1$ and that the revealed shares well satisfy (8). This ensures that the committed sharings $[\![u]\!], [\![w]\!]$ were of degree $s + \ell - 1$ with high probability. The sharing $[\![u]\!]$ is used to ensure the zero-knowledge property by masking $[\![\xi]\!]$ so that revealing $[\![\xi]\!]$ does not leak any information on w.

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

$$Q(X) := \left(X^{d_\beta - \ell} \cdot P_w(X) \mid X^{d_\beta - d_{\beta_1}} \cdot P_{\beta_1}(X) \mid \cdots \mid X^{d_\beta - d_{\beta_t}} \cdot P_{\beta_t}(X)\right) \tag{9}$$

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

$$P_\xi := \Gamma \cdot Q + P_u \ . \tag{10}$$

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

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

1. The prover generates the sharing of the witness $[\![w]\!]$, the sharing of the random mask $[\![u]\!]$, the sharing of the first hint $[\![\beta^1]\!]$ and sharings $[\![\bar\beta^2]\!], \dots, [\![\bar\beta^t]\!]$ of uniform random vectors $\bar\beta^j \in \mathbb{F}^{|\beta^j|}$ for all $j \in [2 : t]$.
2. The prover commits these sharings using a Merkle tree. Specifically, they compute the leaf commitments $\mathsf{com}_i := \mathrm{Com}([\![w]\!]_i, [\![u]\!]_i, [\![\beta^1]\!]_i, [\![\bar\beta^2]\!]_i, \dots, [\![\bar\beta^t]\!]_i; \rho_i)$ and the root $h_1 := \mathsf{MerkleRoot}(\mathsf{com}_1, \dots, \mathsf{com}_N)$ and send h_1 to the verifier.
3. The verifier samples a random matrix Γ of dimensions $\eta \times |Q|$ where $|Q| = |w|_s + |\beta^1|_s + \cdots + |\beta^t|_s$ and sends it to the prover.
4. The prover computes the degree-enforcing polynomial $P_\xi \in (\mathbb{F}[X])^\eta$ using Equation (10) and sends $h_1' := \mathrm{Hash}(P_\xi)$ to the verifier.

The rest of the protocol runs as overviewed above with the following tweaks. During the opening phase, the prover further reveal $P_\xi(e_i)$ for all $i \in S$ with S some set of cardinality $|S| = d_\beta + 1 - \ell$ and disjoint of I (the set of opened shares). During the final checks, the verifier computes $P_\xi(e_i) = \Gamma \cdot Q(e_i) + P_u(e_i)$ for all $i \in I$ from opened shares $[\![w]\!]_i, [\![u]\!]_i, [\![\beta^1]\!]_i, [\![\bar\beta^2]\!]_i, \dots, [\![\bar\beta^t]\!]_i$ (by definition $Q(e_i)$ and $P_u(e_i)$ are linear functions of these shares). From $\{P_\xi(e_i)\}_{i \in S \cup I}$ the verifier reconstructs P_ξ by interpolation and check the hash $h_1' = \mathrm{Hash}(P_\xi)$.

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

To generate a pseudorandom degree-d 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

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

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

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

where

- for all T, $s_T := (s_T^{(1)}, \dots, s_T^{(d-\ell+1)}) \in \mathbb{F}^{d-\ell+1}$;
- $\Delta x := (\Delta x_1, \dots, \Delta x_s)$ satisfies $\Delta x_k := \sum_{T\in\mathcal{S}_\ell^N} s_T^{(k)}$ for all $k \in [1:s]$;
- for all (T,k), $P_{T,k}$ is the degree-d polynomial satisfying

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

with $\{e_j\}_j$ and $\{e'_j\}_j$ two disjoint sets of distinct field elements with $e'_1 = \omega_1, \dots, e'_s = \omega_s$.

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:

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

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

- $\mathsf{TreePRG}$ the subroutine that expands the seed tree from the root seed,

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

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

   The prover sends h_1 to the verifier.

   In the TCitH-MT variant, the prover and verifier additionally perform the following steps:
   (a) The verifier samples a random matrix Γ from $\mathbb{F}^{\eta \times |w|}$ and sends it to the prover;
   (b) The prover computes $P_\xi := \Gamma \cdot Q + P_u$ where Q is computed from the polynomials of the sharings $(\llbracket w \rrbracket, \llbracket \beta^1 \rrbracket, \llbracket \bar{\beta}^2 \rrbracket, \ldots, \llbracket \bar{\beta}^t \rrbracket)$ using (9) and P_u is the polynomial of $\llbracket u \rrbracket$. The prover sends $h_1' := \mathrm{Hash}(P_\xi)$ to the verifier.
2. The verifier samples at random a challenge ε^1 and sends it to the prover;
3. The prover runs the MPC computation in their head. Specifically, the prover computes the shares

$$\llbracket \alpha^1 \rrbracket_i = \Phi^1(\llbracket w \rrbracket_i, \llbracket \beta^1 \rrbracket_i) \ \ \forall i \in [1 : d_{\alpha_1} + 1] \ .$$

4. For $j = 2$ to t:
   (a) The prover computes the plain hint $\beta^j = \psi^j(w, \varepsilon^1, \ldots, \varepsilon^{j-1}; r^j)$ with $r^j \leftarrow \mathrm{PRG}(\mathsf{rseed})$ and deduce the hint correction $\Delta\beta^j = \beta^j - \bar{\beta}^j$ and the hint sharing $\llbracket \beta^j \rrbracket = \llbracket \bar{\beta}^j \rrbracket + \llbracket \Delta\beta^j \rrbracket$ following Equation (7). The prover then computes the hash $h_j = \mathrm{Hash}(\llbracket \alpha^{j-1} \rrbracket_{[1:d_{\alpha_{j-1}}+1]}, \Delta\beta^j)$ and sends it to the verifier.
   (b) The verifier samples at random a challenge ε^j and sends it to the prover;
   (c) The prover runs the MPC computation in their head. Specifically, the prover computes the shares

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

5. The prover computes $h_{t+1} = \mathrm{Hash}(\llbracket \alpha^t \rrbracket_{[1:d_{\alpha_t}+1]})$ and sends it to the verifier.
6. The verifier samples at random a subset $I \subset [1 : N]$ of ℓ parties (*i.e.* $|I| = \ell$) and sends it to the prover.
7. The prover reveals the views of all the parties in I. Specifically, the prover computes

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

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

   In the TCitH-MT variant, the prover further sends $\{P_\xi(e_i)\}_{i \in S}$ for a set $S \subseteq [1 : N]$ of cardinality $|S| = d_\beta + 1 - \ell$ and such that $S \cap I = \emptyset$.
8. The verifier performs the following checks:
   – *(Shares' commitment)* First, the verifier checks the opened views vs. the commitment h_1. Namely it computes:

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

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

$$(\llbracket w \rrbracket_I, \{\llbracket \beta^j \rrbracket_I\}_j) \leftarrow \mathsf{RetrieveShares}(\mathsf{views}_I, I)$$

   and

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

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

   In the TCitH-MT variant, the verifier further computes $P_\xi(e_i) = \Gamma \cdot Q(e_i) + P_u(e_i)$ for all $i \in I$ from opened shares. From $\{P_\xi(e_i)\}_{i \in S \cup I}$ the verifier reconstructs P_ξ by interpolation and check that $h_1' = \mathrm{Hash}(P_\xi)$. If the check fails, the verifier stops and outputs Reject.
   – *(Protocol outcome)* The verifier recovers the plain broadcast value α from $\llbracket \alpha \rrbracket_{I \cup S}$ and checks that $g(\alpha) = 0$. If one of the checks fails, the verifier stops and outputs Reject.

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

Protocol 5: Zero-knowledge protocol: Application of the extended TCitH framework to the general MPC protocol (Protocol 3).

- GetSiblingPath$_I$ the subroutine which computes the sibling paths of the leaves indexed by I,
- RetriveLeavesFromPath the subroutine which recomputes all the leaves except those indexed by I from the corresponding sibling paths.

To handle the Merkle tree, we denote

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

Security. The completeness, soundness, and zero-knowledge properties of the obtained protocol are stated in the following theorem. The input MPC protocol has the following parameters: the size of the sharings N (a.k.a. the number of parties), the privacy threshold ℓ, the pack size s, the maximal degree of the broadcast sharings d_α, the maximal degree for the hint sharings d_β (wlog. $d_\beta \le d_\alpha$), the false positive probability p. With these parameters, the input sharing is a (s, ℓ, N)-packed Shamir's secret sharing and the MPC protocol is required to be ℓ-private. The theorem proof is provided in Appendix A.

Theorem 1. *Let Π be an MPC protocol of parameters $(N, \ell, s, d_\alpha, d_\beta, p)$ complying to the format of Protocol 3. In particular, Π is ℓ-private in the semi-honest model and of false positive probability p. Then, Protocol 5 built from Π satisfies the three following properties:*

- ***Completeness.*** *A prover $\mathcal{P}$ who knows a witness w such that $(x, w) \in \mathcal{R}$ and who follows the steps of the protocol always succeeds in convincing the verifier $\mathcal{V}$.*
- ***Soundness.*** *Suppose that there is an efficient prover $\tilde{\mathcal{P}}$ that, on input x, convinces the honest verifier $\mathcal{V}$ to accept with probability*
$$\tilde{\epsilon} := \Pr[\langle \tilde{\mathcal{P}}, \mathcal{V} \rangle(x) \to 1] \;>\; \epsilon$$
where the soundness error ϵ is defined as
$$\epsilon := \begin{cases} p + (1-p) \cdot \frac{\binom{d_\alpha}{\ell}}{\binom{N}{\ell}} & \textit{for the TCitH-GGM variant,} \\ p + (1-p) \cdot \frac{\binom{d_\alpha}{\ell}}{\binom{N}{\ell}} + \frac{\binom{N}{d_\beta+1}^2}{|\mathbb{F}|^\eta} & \textit{for the TCitH-MT variant.} \end{cases} \tag{11}$$
Then, there exists a probabilistic extraction algorithm $\mathcal{E}$ with time complexity in $\mathsf{poly}(\lambda, (\tilde{\epsilon} - \epsilon)^{-1})$ that, given rewindable black-box access to $\tilde{\mathcal{P}}$, outputs either a witness w satisfying $(x, w) \in \mathcal{R}$, a hash collision, or a commitment collision.
- ***Honest-Verifier Zero-Knowledge.*** *Let the pseudorandom generator* PRG *used in Protocol 2 be $(t, \epsilon_{\mathrm{PRG}})$-secure and the commitment scheme* Com *be $(t, \epsilon_{\mathrm{COM}})$-hiding. There exists an efficient simulator $\mathcal{S}$ which, given random challenge I outputs a transcript which is $(t, \epsilon_{\mathrm{PRG}} + \epsilon_{\mathrm{COM}})$-indistinguishable from a real transcript of Protocol 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 $\mathbb{F}$. It can be an issue when working with statements based on a small field $\mathbb{F}_0$ as the binary field. To handle that, we should share the secret in an larger field extension $\mathbb{F}$ 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 ($[\mathbb{F} : \mathbb{F}_0]$ times more expensive, where $[\mathbb{F} : \mathbb{F}_0]$ 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 $\mathbb{F}_0$ (for example, by checking that a random $\mathbb{F}_0$-linear combination of the shared values is in $\mathbb{F}_0$).

TCitH-GGM

```
$\underline{\mathsf{GenerateAndCommitShares}}(w, \beta^1, \mathsf{rseed})$:
    $\{\mathsf{seed}_T\}_{T \in \mathcal{S}_\ell^N} \leftarrow \mathsf{TreePRG}(\mathsf{rseed})$
    For all $T$,
        $s_T^0 := (s_{T,a}^0, s_{T,b}^0) \leftarrow \mathrm{PRG}(\mathsf{seed}_T, 0)$
        $s_T^1 := (s_{T,a}^1, s_{T,b}^1) \leftarrow \mathrm{PRG}(\mathsf{seed}_T, 1)$
    $\Delta w \leftarrow w - \sum_T s_{T,a}^0$
    $\Delta\beta^1 \leftarrow \beta^1 - \sum_T s_{T,a}^1$
    $\llbracket w \rrbracket \leftarrow \mathsf{GenerateSharing}(\Delta w, \{s_T^0\}_T, \ell)$
    $\llbracket \beta^1 \rrbracket \leftarrow \mathsf{GenerateSharing}(\Delta\beta^1, \{s_T^1\}_T, d_{\beta_1})$
    For all $j \in [2:t]$,
        $s_T^j := (s_{T,a}^j, s_{T,b}^j) \leftarrow \mathrm{PRG}(\mathsf{seed}_T, j)$
        $\llbracket \bar\beta^j \rrbracket \leftarrow \mathsf{GenerateSharing}(0, \{s_T^j\}_T, d_{\beta_j})$
    For all $T \in \mathcal{S}_\ell^N$:
        If $T \neq T_0$,
            $\mathsf{com}_T \leftarrow \mathrm{Com}(\mathsf{seed}_T; \rho_T)$
        Else
            $\mathsf{com}_T \leftarrow \mathrm{Com}(\mathsf{seed}_T, \Delta w, \Delta\beta^1; \rho_T)$
    $h_1 \leftarrow \mathrm{Hash}(\{\mathsf{com}_T\}_T)$
    Return $(\llbracket w \rrbracket, \emptyset, \llbracket \beta^1 \rrbracket, \llbracket \bar\beta^2 \rrbracket, \ldots, \llbracket \bar\beta^t \rrbracket, h_1)$
```

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

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

```
$\underline{\mathsf{RetrieveShares}}(\mathsf{views}_I, I)$:
    $\mathsf{path}_I \parallel (\Delta w, \Delta\beta^1, \ldots, \Delta\beta^t) \leftarrow \mathsf{views}_I$
    $\{\mathsf{seed}_T\}_{T \neq I} \leftarrow \mathsf{RetriveLeavesFromPath}(\mathsf{path}_I)$
    For all $T \neq I$,
        $s_T^j \leftarrow \mathrm{PRG}(\mathsf{seed}_T, j)$ for $j \in \{0, \ldots, t\}$
    For all $i \in I$,
        $\llbracket w \rrbracket_i \leftarrow \mathsf{GeneratePartyShare}_i((s_T^0)_{T: i \notin T}, \Delta w, \ell)$
        For all $j \in [1:t]$:
            $\llbracket \beta^j \rrbracket_i \leftarrow \mathsf{GeneratePartyShare}_i((s_T^j)_{T: i \notin T}, \Delta\beta^j, d_{\beta_j})$
    Return $(\llbracket w \rrbracket_I, \llbracket \beta^1 \rrbracket_I, \ldots, \llbracket \beta^t \rrbracket_I)$
```

TCitH-MT

```
$\underline{\mathsf{GenerateAndCommitShares}}(w, \beta^1, \mathsf{rseed})$:
    $\{r_k^0\}_{k \in [0:\ell-1]} \leftarrow \mathrm{PRG}(\mathsf{rseed}, 0)$
    $\{r_k^1\}_{k \in [0:d_{\beta_1}-s]} \leftarrow \mathrm{PRG}(\mathsf{rseed}, 1)$
    $u, \{r_k^u\}_{k \in [0:d_\beta-s]} \leftarrow \mathrm{PRG}(\mathsf{rseed}, -1)$
    For all $i \in [1:N]$:
        $P_w(X) = I_w(X) + F(X) \cdot \sum_{k=0}^{\ell-1} r_k^0 \cdot X^k$
        $P_{\beta^1}(X) = I_{\beta^1}(X) + F(X) \cdot \sum_{k=0}^{d_{\beta_1}-s} r_k^1 \cdot X^k$
        $P_u(X) = I_u(X) + F(X) \cdot \sum_{k=0}^{d_\beta-s} r_k^u \cdot X^k$
    For all $j \in [2:t]$,
        $\bar\beta^j, \{r_k^j\}_{k \in [0:d_{\beta_j}-s]} \leftarrow \mathrm{PRG}(\mathsf{rseed}, j)$
        For all $i \in [1:N]$:
            $P_{\bar\beta^j}(X) := I_{\bar\beta^j} + F(X) \cdot \sum_{k=0}^{d_{\beta_j}-s} r_k^j \cdot X^k$
    For all $i \in [1:N]$
        $\llbracket w \rrbracket_i \leftarrow P_w(e_i)$
        $\llbracket \beta^1 \rrbracket_i \leftarrow P_{\beta^1}(e_i)$
        $\llbracket \bar\beta^j \rrbracket_i \leftarrow P_{\bar\beta^j}(e_i)$ for all $j \geq 2$
        $\mathsf{com}_i := \mathrm{Com}(\llbracket w \rrbracket_i \parallel \llbracket \beta^1 \rrbracket_i \parallel \llbracket \bar\beta^2 \rrbracket_i \parallel \ldots \parallel \llbracket \bar\beta^t \rrbracket_i, \rho_i)$
    $h_1 := \mathsf{MerkleRoot}(\mathsf{com}_1, \ldots, \mathsf{com}_N)$
    Return $(\llbracket w \rrbracket, \llbracket u \rrbracket, \llbracket \beta^1 \rrbracket, \llbracket \bar\beta^2 \rrbracket, \ldots, \llbracket \bar\beta^t \rrbracket, h_1)$
```

```
$\underline{\mathsf{OpenShares}}(w, \mathsf{rseed}, \beta^1, \{\Delta\beta^j\}_{j \geq 2}, I)$:
    $(w, \mathsf{rseed}, \beta^1) \mapsto (\llbracket w \rrbracket, \llbracket \beta^1 \rrbracket, \llbracket \bar\beta^2 \rrbracket, \ldots, \llbracket \bar\beta^t \rrbracket, \{\mathsf{com}_i\}_i)$
    $\mathsf{views}_I \leftarrow (\llbracket w \rrbracket_I, \llbracket \beta^1 \rrbracket_I, \llbracket \bar\beta^2 \rrbracket_I, \ldots, \llbracket \bar\beta^t \rrbracket_I, \Delta\beta^2, \ldots, \Delta\beta^t)$
    $\mathsf{decom}_I \leftarrow \mathsf{GetAuthPath}(\{\mathsf{com}_i\}_i, I)$
    Return $(\mathsf{views}_I, \mathsf{decom}_I)$
```

```
$\underline{\mathsf{VerifyDecommitment}}(\mathsf{views}_I, \mathsf{decom}_I, I)$:
    $(\llbracket w \rrbracket_I, \llbracket \beta^1 \rrbracket_I, \llbracket \bar\beta^2 \rrbracket_I, \ldots, \llbracket \bar\beta^t \rrbracket_I, \Delta\beta^2, \ldots, \Delta\beta^t) \leftarrow \mathsf{views}_I$

    For all $i \in I$:
        $\mathsf{com}_i := \mathrm{Com}(\llbracket w \rrbracket_i \parallel \llbracket \beta^1 \rrbracket_i \parallel \llbracket \bar\beta^2 \rrbracket_i \parallel \ldots \parallel \llbracket \bar\beta^t \rrbracket_i, \rho_i)$
    $h_1 \leftarrow \mathsf{RetrieveRootFromPath}(\mathsf{decom}_I, \{\mathsf{com}_i\}_{i \in I})$
    Return $h_1$
```

```
$\underline{\mathsf{RetrieveShares}}(\mathsf{views}_I, I)$:
    $\mathsf{shares}_I \parallel (\Delta\beta^2, \ldots, \Delta\beta^t) \leftarrow \mathsf{views}_I$
    $(\llbracket w \rrbracket_i, \llbracket \beta^1 \rrbracket_i, \llbracket \bar\beta^2 \rrbracket_i, \ldots, \llbracket \bar\beta^t \rrbracket_i)_{i \in I} \leftarrow \mathsf{shares}_I$
    For all $i \in I$,
        If $e_i \neq \infty$,
            $\llbracket \beta^j \rrbracket_i \leftarrow \Delta\beta^j + \llbracket \bar\beta^j \rrbracket_i$ for all $j \geq 2$
        Else,
            $\llbracket \beta^j \rrbracket_i \leftarrow \llbracket \bar\beta^j \rrbracket_i$ for all $j \geq 2$
    Return $(\llbracket w \rrbracket_I, \llbracket \beta^1 \rrbracket_I, \ldots, \llbracket \beta^t \rrbracket_I)$
```

Fig. 2: Sharing generation and commitment routines, where the polynomial $F(X)$ is defined as $\prod_{k=1}^{s}(X - \omega_k)$ and I_v is the polynomial defined by interpolation such that $\forall i \in [1:s], I(\omega_i) = v_i$.

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 $s_T \in \mathbb{F}_0^{|w|}$ for all $T \in \mathcal{S}_\ell^N$, we compute the auxiliary value $\Delta w \in \mathbb{F}_0^{|w|}$ as

$$\Delta w := w - \sum_{T \in \mathcal{S}_\ell^N} s_T \in \mathbb{F}_0^{|w|},$$

but we compute the shares $[\![w]\!]_i$ using Equation (4) with parties' evaluation points $\{e_j\}$ living in the field extension $\mathbb{F}$ (instead of $\mathbb{F}_0$). As consequence, the shares $\{[\![w]\!]_i\}_i$ live in $\mathbb{F}^{|w|}$ instead of $\mathbb{F}_0^{|w|}$. Since the proof transcript contains the auxiliary value Δw 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 $[\![\beta^1]\!], \dots, [\![\beta^t]\!]$. However, the lifting impacts the communication cost due to the broadcasted sharings $[\![\alpha^1]\!], \dots, [\![\alpha^t]\!]$.

5 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 5.1). We introduce two general zero-knowledge proof systems obtained by applying the TCitH framework to simple MPC protocols: the first one, TCitH-Π_{PC}, checks polynomial constraints on the witness in the non-packed setting (Section 5.2) while the second one, TCitH-Π_{LPPC}, checks global linear constraints and parallel polynomial constraints on the witness in the packed setting (Section 5.3). We show how these proof systems results in competitive zero-knowledge arguments for (low-degree) arithmetic circuits (Section 5.4). We further described improved post-quantum (ring) signatures from TCitH-Π_{PC} (Sections 5.5 and 5.6) as well as zero-knowledge arguments for lattices from TCitH-Π_{LPPC} (Section 5.7).

5.1 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 $\mathcal{O}_H$. 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 $\Pi_\$$ described below (see Protocol 6). We further introduce variants of this protocol to generate random sharings of zero and random sharings summing to zero.

Generation of a Random Sharing ($\Pi_\$$). The idea behind our approach is to define the polynomial P_v underlying the random sharing $[\![v]\!]$ from lower degree polynomials $P_{u_1}, \dots, P_{u_n}$ 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 $P_{u_1}, \dots, P_{u_{n-1}}$ of degree d_β and P_{u_n} of degree $d' - (n-1)\delta \le d_\beta$ we get that P_v is of degree d'. Based on this definition of P_v, $\Pi_\$$ is formally defined in Protocol 6.

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 ℓ 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 $|J| \le d' + 1 - \ell$ s.t. $I \cap J = \emptyset$, $[\![v]\!]_J$ is a uniform random tuple of $\mathbb{F}^{|J|}$ mutually independent of the views.*

1. The parties get $n-1$ random sharings $[\![u_1]\!]$, …, $[\![u_{n-1}]\!]$ of degree d_β from the hint oracle $\mathcal{O}_H$,
2. The parties get a random sharing $[\![u_n]\!]$ of degree $d' - (n-1)\delta$ from the hint oracle $\mathcal{O}_H$,
3. The parties locally computes
$$[\![v]\!]_i = \sum_{j=1}^{n} [\![u_j]\!]_i \cdot (e_i)^{(j-1)\delta} .$$
4. The parties now hold a random degree-d' sharing $[\![v]\!]$.

Protocol 6: $\mathit{\Pi}_\$$ – Generation of a random degree-d' sharing.

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

$$\underbrace{P_{u_j}(X)}_{\substack{\text{of degree} \\ d_\beta \text{ when } j<n \\ d'-(n-1)\delta \text{ when } j=n}} = \underbrace{I_{u_j,I}(X)}_{\text{of degree } \ell-1} + \underbrace{\prod_{i\in I}(X-e_i)}_{\text{of degree } \ell} \cdot \underbrace{R_{u_j,I}(X)}_{\substack{\text{of degree} \\ d_\beta-\ell \text{ when } j<n \\ d'-(n-1)\delta-\ell \text{ when } j=n}} ,$$

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_j} is a uniformly-random polynomial satisfying $P_{u_j}(e_i) = [\![u_j]\!]_i$ for all $i \in I$.

So, we have

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

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

For a target degree d' and a target maximal degree d_β for the hints, Lemma 1 gives the parameters n and δ of the generation protocol $\mathit{\Pi}_\$$. For the TCitH-MT variant, the parameter d_β can then be chosen to minimize the communication by balancing its impact on the degree enforcement parameter η and the number n of hints $[\![u_i]\!]$ (both impacting the proof size).

Generation of Sharings of Zero ($\mathit{\Pi}_0$). Let us now consider the case of the generation of a uniformly-random degree-d' sharing of $(0,\dots,0) \in \mathbb{F}^s$. This can be achieved by using the protocol $\mathit{\Pi}_\$$ to generate a uniformly-random degree-$(d'-s)$ sharing which is then locally multiplied by the constant sharing $[\![\mathbf{0}]\!]$ corresponding to the degree-s polynomial $P_0(X) = \prod_{i=1}^{s}(X-\omega_i)$ (where we recall that the ω_i's are the evaluation points for the plain coordinates). Formally, the protocol $\mathit{\Pi}_0$ for the generation of a uniformly-random degree-d' sharing of $(0,\dots,0) \in \mathbb{F}^s$ performs the following steps:

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

In terms of privacy, Lemma 1 implies that given the views of any ℓ parties of indices in J, the generated sharing $[\![z]\!]$ is uniformly-random conditioned to $(P_z(\omega_1),\dots,P_z(\omega_s)) = (0,\dots,0)$ 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 $\Pi_\$$. In this tweaked version, the sharings $[\![u_1]\!]$, ..., $[\![u_n]\!]$ returned by the hint oracle $\mathcal{O}_H$ are not fully random: the constant term of the polynomial P_{u_1} is defined w.r.t. the random polynomials $\{P_{u_i}\}_{i \geq 2}$ 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 $\Pi_\$$ 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 ℓ parties of indices in J, the generated sharing $[\![v]\!]$ is uniformly-random conditioned to $[\![v]\!]_J$ consistent with the views and to $\sum_{i=1}^{s} v_i = 0$.

5.2 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).

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 Π_{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 Π_0 (Section 5.1).
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 α.
6. The parties output Accept if and only if $\alpha = 0$.

Protocol 7: Π_{PC} – Verification of polynomial constraints.

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

Proof. It is easy to check that Π_{PC} outputs 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, α is uniformly random in $\mathbb{F}$ since $\gamma_{j'}$ has been chosen uniformly at random in $\mathbb{F}$. We get that Π_{PC} shall output Accept with probability $\frac{1}{|\mathbb{F}|}$, which corresponds to its false positive probability. Finally, the ℓ-privacy of the protocol holds for the following reason. In our MPC model, the protocol is ℓ-private as long as the broadcast sharings do not leak information on the witness. In the present case, we have that $[\![v]\!]$ 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 Π_{PC} is thus ℓ-private. $\square$

[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 $i \in [2:s]$.

Parallel repetitions. The false positive probability can be made arbitrarily small by performing several repetitions of Π_{PC} in parallel. The obtained protocol $\Pi_{\text{PC}}^{(\rho)}$ is similar to Π_{PC} with the following differences:

- ρ sharings $[\![v_1]\!], \ldots, [\![v_\rho]\!]$ are generated using Π_0 in Step 2,
- ρ batches of m random values $(\gamma_{1,i})_i, \ldots, (\gamma_{\rho,i})_i$ are requested from $\mathcal{O}_R$ in Step 3,
- ρ 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 accept iff the ρ 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-Π_{pc} proof system. By Theorem 1, when applying the TCitH framework to the protocol $\Pi_{\text{PC}}^{(\rho)}$ we obtain a complete, sound, and HVZK proof system of soundness error

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

To obtain a zero-knowledge non-interactive argument of knowledge, we perform τ 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 [KZ20a]. To achieve λ bits of security in the non-interactive setting, we propose to select

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

Here, N, ℓ and d_β are flexible parameters which can be chosen to optimize the proof size. The proof transcript includes:

- The opened shares $[\![w]\!]_I$ of the witness $w \in \mathbb{F}^{|w|}$, for each of the τ PoK repetitions.
- The opened shares of the hints used to build the sharing $[\![v]\!]$, for each of the ρ MPC repetitions of the τ PoK repetitions. With the variant TCitH-GGM, these shares are communication-free.
- The degree-$(d\ell)$ sharing $[\![\alpha]\!]$, for each of the ρ MPC repetitions of the τ PoK repetitions. Since $[\![\alpha]\!]_I$ can be recomputed by the verifier and since the α 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 $[\![u]\!]_I$ used to mask the output of the sharing-degree test (where $u \in \mathbb{F}^{\eta\cdot s}$) together with $d_\beta + 1 - \ell$ additional shares of $[\![\xi]\!]$ (where $\xi \in \mathbb{F}^{\eta\cdot s}$), for each of the τ PoK repetitions.

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

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

and when applying TCitH-MT:

$$\textsc{Size}_{\mathrm{MT}} = 6\lambda + \tau \cdot \Bigg(\underbrace{\ell \cdot (|w| + n_{\mathrm{hints}} \cdot \rho) \cdot \log_2 |\mathbb{F}|}_{[\![w]\!]_I, [\![v]\!]_I} + \underbrace{(d-1)\ell \cdot \rho \cdot \log_2 |\mathbb{F}|}_{[\![\alpha]\!]} + \underbrace{2\lambda \cdot \ell \cdot \log_2 \frac{N}{\ell}}_{\text{Merkle tree}} + \underbrace{(d_\beta + 1) \cdot \eta \cdot \log_2 |\mathbb{F}|}_{\text{degree enforcing}} \Bigg)$$

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

5.3 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 $w = (w_1, \ldots, w_{ns}) \in \mathbb{F}^{ns}$ arranged in n packs:

$$w = \begin{pmatrix} w_1, & \ldots, & w_s, \\ w_{s+1}, & \ldots, & w_{2s}, \\ \vdots & & \vdots \end{pmatrix} \quad \text{s.t.} \quad [\![w]\!] = \begin{pmatrix} [\![(w_1, \quad \ldots, \quad w_s)]\!] \\ [\![(w_{s+1}, \ldots, w_{2s})]\!] \\ \vdots \end{pmatrix}$$

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 m_1 the number of (parallel) polynomial constraints and m_2 the number of (global) linear constraints, the protocol Π_{LPPC} checks that w verifies:

1. *(parallel polynomial constraints)* the $w^{(k)}$'s satisfy some polynomial relations:

$$\forall j \in [1:m_1],\ \forall k \in [1:s],\ f_j(w^{(k)}) = 0$$

   where $f_1, \ldots, f_{m_1}$ are polynomials from $\mathbb{F}[X_1, \ldots, X_n]$ 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 $\mathbb{F}^{m_2 \times (ns)}$ and t is a vector from $\mathbb{F}^{m_2}$.

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 $(0, \ldots, 0) \in \mathbb{F}^s$ in the same way as protocol Π_{PC}. To check the global linear constraints, we use a similar approach as Ligero [AHIV17]. Each row $A_j = (A_{j,1}, \ldots, A_{j,ns})$ of the matrix gives rise to a constant sharing $[\![A_j]\!]$ defined as:

$$[\![A_j]\!] = \begin{pmatrix} [\![(A_{j,1}, \quad \ldots, \quad A_{j,s})]\!] \\ [\![(A_{j,s+1}, \ldots, A_{j,2s})]\!] \\ \vdots \end{pmatrix}$$

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 $[\![b_j]\!] := \langle [\![A_j]\!], [\![w]\!] \rangle$, we have that the jth linear constraint $\langle A_j, w \rangle = t_j$ is satisfied if and only if $[\![b_j]\!]$ shares a tuple $b_j = (b_{j,1}, \ldots, b_{j,s}) \in \mathbb{F}^s$ satisfying $\sum_{i=1}^{s} b_{j,i} = t_j$. The m_2 linear constraints are batched by computing $[\![\alpha']\!] = [\![v']\!] + \sum_{j=1}^{m_2} \gamma'_j [\![b_j]\!]$, for $[\![v']\!]$ a random degree-$(2s + \ell - 2)$ sharing of a random tuple $v' = (v'_1, \ldots, v'_s) \in \mathbb{F}^s$ satisfying $\sum_{i=1}^{s} v'_i = 0$. We then have $A \cdot w = t$ if $[\![\alpha']\!]$ is such that $\sum_{i=1}^{s} \alpha'_i = \sum_{j=1}^{m_2} \gamma'_j t_j$ with high probability (specifically with false positive probability $1/|\mathbb{F}|$). The obtained protocol Π_{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 $v = (0,\ldots,0) \in \mathbb{F}^s$ using Π_0 (Section 5.1).
3. The parties get a uniformly-random degree-$(2s+\ell-2)$ sharing $[\![v']\!]$ of a random tuple $v' = (v'_1,\ldots,v'_s) \in \mathbb{F}^s$ satisfying $\sum_{i=1}^{s} v'_i = 0$ using $\Pi_{\Sigma 0}$ (Section 5.1).
4. The parties receive random values $\gamma_1,\ldots,\gamma_{m_1} \in \mathbb{F}$ and $\gamma'_1,\ldots,\gamma'_{m_2} \in \mathbb{F}$ from $\mathcal{O}_R$.
5. The parties locally compute
$$[\![\alpha]\!] = [\![v]\!] + \sum_{j=1}^{m_1} \gamma_j \cdot f_j([\![w]\!])$$
$$[\![\alpha']\!] = [\![v']\!] + \sum_{j=1}^{m_2} \gamma'_j \cdot [\![b_j]\!] \quad \text{with} \quad [\![b_j]\!] = \langle [\![A_j]\!], [\![w]\!] \rangle$$
6. The parties open $[\![\alpha]\!]$ and $[\![\alpha']\!]$ to publicly recompute $\alpha, \alpha' \in \mathbb{F}^s$.
7. The parties output Accept if and only if $\alpha = (0,\ldots,0)$ and $\sum_{i=1}^{s} \alpha'_i = \sum_{j=1}^{m_2} \gamma'_j t_j$.

Protocol 8: Π_{LPPC} – Verification of linear & parallel polynomial constraints.

Lemma 3. *The protocol Π_{LPPC} is correct, namely it always outputs* Accept *when w satisfies the parallel polynomial constraints and the global linear constraints described above. The protocol Π_{LPPC} is also sound with false positive probability $\frac{1}{|\mathbb{F}|}$ and ℓ-private.*

Proof. It is easy to check that Π_{LPPC} outputs Accept when the witnesses satisfy the desired properties and when the hints are well-formed, since

$$\forall i \in [1:s],\ \alpha_i = v_i + \sum_{j=1}^{m_1} \gamma_j f_j(w^{(i)}) = 0 + \sum_{j=1}^{m} \gamma_i \cdot 0 = 0\ ,$$
$$\sum_{i=1}^{s} \alpha'_i = \sum_{i=1}^{s} v'_i + \sum_{i=1}^{s}\sum_{j=1}^{m_2} \gamma'_j b_{j,i} = 0 + \sum_{j=1}^{m_2} \gamma'_j \Big(\sum_{i=1}^{s} b_{j,i}\Big) = \sum_{j=1}^{m_2} \gamma'_j t_j .$$

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 $\gamma_{j'}$ has been chosen uniformly at random in $\mathbb{F}$.
– Or there exists $j' \in [1:m_2]$ such that $\langle A_j, w\rangle \neq t_j$. 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_{LPPC} shall output Accept with probability $1/|\mathbb{F}|$, which corresponds to its false positive probability. In our MPC model, the protocol is ℓ-private as long as the broadcast sharings do not leak information on the witness. In the present case, we have that $[\![v]\!]$ is a uniformly-random degree-$(d\cdot(s+\ell-1))$ sharing of $(0,\ldots,0)\in\mathbb{F}^s$, which implies that $[\![\alpha]\!]$ is a uniformly-random degree-$(d\cdot(s+\ell-1))$ sharing of $\big(\sum_{j=1}^{m} \gamma_j \cdot f_j(w^{(k)})\big)_k = (0,\ldots,0)$. Moreover, we have that $[\![v']\!]$ is a uniformly-random degree-$(2s+\ell-2)$ sharing of $v' \in \mathbb{K}^s$ satisfying $\sum_{j=1}^{s} v'_j = 0$, which implies that $[\![\alpha']\!]$ is a uniformly-random degree-$(2s+\ell-2)$ sharing of $\alpha' = v' + \sum_{j=1}^{m_2} \gamma'_j b_j$. Thanks to the randomness of v', α' is a uniformly-random tuple of $\mathbb{F}^s$ conditioned to $\sum_{i=1}^{s} \alpha'_i = \sum_{j=1}^{m_2} \gamma'_j \cdot t_j$. The protocol Π_{LPPC} is thus ℓ-private. □

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

The TCitH-$\Pi_{\mathbf{lppc}}$ proof system. By Theorem 1, when applying the TCitH framework to $\Pi_{\text{LPPC}}^{(\rho)}$ we obtain a complete, sound, and HVZK proof system of soundness error

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

To obtain a zero-knowledge non-interactive argument of knowledge, we perform τ 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 [KZ20a]. To achieve λ bits of security in the non-interactive setting, we propose to select

- the parameter η of the degree-enforcing commitment in TCitH-MT such that $\binom{N}{s+\ell}^2 / (2|\mathbb{F}|^\eta) \le 2^{-\lambda}$,
- the number ρ of MPC repetitions in the protocol $\Pi_{\text{LPPC}}^{(\rho)}$ such that $1/|\mathbb{F}|^\rho \le 2^{-\lambda}$,
- the number τ of PoK repetitions such that $\left(\frac{\binom{d\cdot(s+\ell-1)}{\ell}}{\binom{N}{\ell}}\right)^\tau \le 2^{-\lambda}$.

Here, N, ℓ and d_β are flexible parameters which can be chosen to optimize the proof size. The proof transcript includes:

- The opened shares $[\![w]\!]_I$ of the witness $w \in \mathbb{F}^{ns}$, for each of the τ PoK repetitions.
- The opened shares of the hints used to build the sharings $[\![v]\!]$ and $[\![v']\!]$, for each of the ρ MPC repetitions of the τ PoK repetitions. With the variant TCitH-GGM, these shares are communication-free except for the correction term $\Delta u_1 \in \mathbb{F}$ arising in the generation of $[\![v']\!]$ (see Section 5.1).
- The degree-$(d \cdot (s + \ell - 1))$ sharing $[\![\alpha]\!]$ and the degree-$(2s + \ell - 2)$ sharing $[\![\alpha']\!]$, for each of the ρ MPC repetitions of the τ PoK repetitions. Since $[\![\alpha]\!]_I$ and $[\![\alpha']\!]_I$ can be recomputed by the verifier, since the $\alpha \in \mathbb{F}^s$ should be zero, and since the sum of the coordinates of $\alpha' \in \mathbb{F}$ should be equal to a public value (which is $\sum_j \gamma'_j t_j$), the proof just needs to include $(d \cdot (s + \ell - 1) + 1) - \ell - s$ shares for $[\![\alpha]\!]$ and $((2s + \ell - 2) + 1) - \ell - 1$ shares for $[\![\alpha']\!]$.
- 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 $[\![u]\!]_I$ used to mask the output of the sharing-degree test (where $u \in \mathbb{F}^{\eta\cdot s}$) together with $d_\beta + 1 - \ell$ additional shares of $[\![\xi]\!]$ (where $\xi \in \mathbb{F}^{\eta\cdot s}$), for each of the τ PoK repetitions.

We obtain the following communication cost (in bits):

$$\textsc{Size}_{\text{GGM}} = 4\lambda + \tau \cdot \Bigg(\underbrace{\ell \cdot (ns + \rho) \cdot \log_2 |\mathbb{F}|}_{\Delta w, \Delta u_1} + \Bigg[\underbrace{(d-1)\cdot(s+\ell-1)}_{[\![\alpha]\!]} + \underbrace{(2s-2)}_{[\![\alpha']\!]} \Bigg] \cdot \rho \cdot \log_2 |\mathbb{F}| + \underbrace{\lambda \cdot \log_2 N}_{\text{GGM tree}} + 2\lambda \Bigg)$$

for TCitH-GGM, and

$$\textsc{Size}_{\text{MT}} = 6\lambda + \tau \cdot \left(\underbrace{(d_\beta + 1) \cdot \eta \cdot \log_2 |\mathbb{F}|}_{\text{degree enforcing}} + \ell \cdot \underbrace{(n + n_{\text{hints}} \cdot \rho) \cdot \log_2 |\mathbb{F}|}_{[\![w]\!]_I, [\![v]\!]_I, [\![v']\!]_I} \right.$$
$$\left. + \left[\underbrace{(d-1) \cdot (s + \ell - 1)}_{[\![\alpha]\!]} + \underbrace{(2s - 2)}_{[\![\alpha']\!]} \right] \cdot \rho \cdot \log_2 |\mathbb{F}| + \underbrace{2\lambda \cdot \ell \cdot \log_2 \frac{N}{\ell}}_{\text{Merkle tree}} \right) \tag{12}$$

for TCitH-MT, where $n_{\text{hints}} := \left\lceil \frac{(d-1)\cdot(s+\ell-1)}{d_\beta - \ell + 1} \right\rceil + \left\lceil \frac{2s-1}{d_\beta - \ell + 1} \right\rceil$.

Remark 8. One could try to optimize the above protocol by batching $[\![\alpha]\!]$ and $[\![\alpha']\!]$ into a single broadcasted sharing. But to keep the soundness of the protocol in doing so, the unique broadcast would be of higher degree $d_\alpha = d \cdot (s + \ell - 1) + (s - 1)$ which, one one hand, would not reduce the number of field elements in the proof transcript, and, on the other hand, would increase the soundness error of a single repetition (see Equation 11). Therefore, such an optimization is not interesting here.

Remark 9. The protocol Π_{LPPC} can be seen as a generalization of the protocol Π_{PC}. Indeed, Π_{LPPC} with $s = 1$ is equivalent to Π_{PC}. To see this, observe that the broadcast sharing of α' 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 $s = 1$).

5.4 Zero-Knowledge Arguments for Arithmetic Circuits

In this section, we apply the previous zero-knowledge arguments TCitH-Π_{PC} and TCitH-Π_{LPPC} to general (low-degree) arithmetic circuits.

Arguments for Low-Degree Arithmetic Circuits. TCitH-Π_{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 jth coordinate of the output y and C_j denotes the subcircuit computing this output coordinate. From the analysis of Section 5.2, 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-Π_{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.[11]

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

[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.

$m_2 = 2|C| + |y|$ global linear constraints to verify (for the computation of a, b and y). For this particular LPPC instance, the MPC protocol Π_{LPPC} is a slightly optimized version of the Ligero MPC protocol, which we therefore denote Π_{Ligero} here.[12] The zero-knowledge argument obtained when applying our framework is denoted TCitH-Π_{Ligero}.

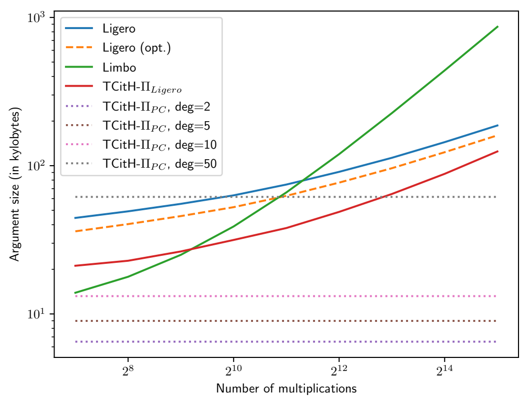
Remark 10. Let us mention that the TCitH-Π_{LPPC} proof system can deal with circuits with higher-degree gates instead of just considering multiplication gates. This could provide significant gain for some applications but highly depends on the topology of the underlying circuit.

Comparison. Figures 3a and 3b compare our two schemes TCitH-Π_{PC} and TCitH-Π_{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.)") 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).

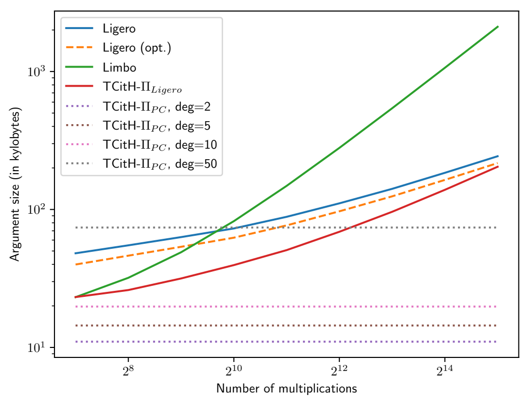
From those figures, we observe that TCitH-Π_{PC} achieves very competitive cost when the circuit degree is small (below 50). For general circuits, both Ligero and our scheme quickly outperform Limbo when the circuit size increases, and our scheme performs better than Ligero for small-to-medium size circuits, specifically circuits with $\leq 2^{16}$ multiplication gates. A qualitative comparison between Ligero and our scheme is given in Section 6.2.

The following table further provides concrete parameters for Ligero vs. TCitH-Π_{Ligero} for a circuit with 256 multiplications:

Scheme	$\lvert\mathbb{F}\rvert$	$\lvert x\rvert$	$\lvert C\rvert$	τ	N	ℓ	s	η	ρ	Size
Ligero	8192	100	256	1	975	219	106	–	10	49.2 kB
TCitH-Π_{Ligero}				1	924	50	28	69	10	22.9 kB
Ligero	2^{32}	100	256	1	975	219	106	–	4	54.9 kB
TCitH-Π_{Ligero}				1	924	50	28	28	4	26.1 kB



(a) Over the field $\mathbb{F}_{8192}$



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

Fig. 3: Comparison of TCitH-Π_{PC}, TCitH-Π_{Ligero}, Ligero [AHIV17] and Limbo [DOT21] for arithmetic circuits (with input $x \in \mathbb{F}^{100}$).

[12] Compared to the original Ligero MPC protocol, Π_{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.

5.5 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_{PC} (Section 5.2) 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_{PC} 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 $A_1, \ldots, A_m \in \mathbb{F}_q^{n\times n}$, vectors $b_1, \ldots, b_m \in \mathbb{F}_q^n$ and scalars $y_1, \ldots, y_m \in \mathbb{F}_q$, the MQ problem consists in finding a vector x such that, for all j, $x^T A_j x + b_j^T x = y_j$. Applying the proof system to this problem is straightforward since it is naturally expressed as degree-2 polynomials. We just need to define the polynomials $f_1, \ldots, f_m$ as

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

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

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

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

		TCitH-GGM			TCitH-MT		
		Size	Signing	Verif	Size	Signing	Verif
MQ over $\mathbb{F}_{251}$	MQOM	6 575 B	5.97 ms	5.57 ms	≈ 13000 B	-	-
$(m = n = 43)$	Our scheme	4 257 B	5.23 ms	4.77 ms	7 177 B	3.55 ms	0.63 ms
SD over $\mathbb{F}_{251}$	SDitH	8 241 B	6.44 ms	6.11 ms	10 117 B	1.55 ms	0.17 ms
$(n = 230, k = 126, k = 79)$	Our scheme	7 335 B	6.73 ms	6.45 ms	10 255 B	4.85 ms	0.30 ms

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

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.

Hardness Assumption	n	m	k	w	Former size	TCitH-GGM	Saving
MQ over $\mathbb{F}_4$	88	88	-	-	8 609 B [Wan22]	3 858 B	−55%
SD over $\mathbb{F}_2$	1280	-	640	132	11 160 B [FJR22]	7 354 B	−34%
6-split SD over $\mathbb{F}_2$	1536	-	888	120	12 066 B [FJR22]	6 974 B	−42%

Table 4: Signature sizes using alternative MQ and SD problems.

Application to other NIST post-quantum signature candidates. As explained in Section 3.4, several MPCitH-based schemes relying on different hardness assumptions have been proposed in the new NIST call for additional post-quantum signatures. We already deal with the case of MQOM and SDitH above (the $\mathbb{F}_{251}$ instance in both cases) but our proof system can also be applied to the other schemes. We provide in Table 5 a list of NIST candidates for which an application of our TCitH-Π_{LC} proof system (GGM variant) provides an improvement of the signature size, namely all the MPCitH-based candidates except PERK [ABB$^+$23a] (which is based on the shared-permutation technique [FJR23] and does not fit our framework) and AIMer [CCH$^+$23].

As it was the case for the non-structured MQ problem, adapting the NIST candidate Biscuit [BKPV23] is straightforward since it relies on a structured variant of the MQ problem, called the PowAff2 problem, which is directly expressed as degree-2 polynomial constraints on the witness. MiRitH [ABB$^+$23b] relies on the MinRank problem which consists, given $k+1$ matrices $M_0, M_1, \ldots, M_k$, in finding x such that the rank of $E := M_0 + \sum_{j=1}^{k} x_j \cdot M_j \in \mathbb{F}_q^{m\times n}$ is smaller than a public bound r. The idea of MiRitH is to show that the $n-r$ last columns E_R of $E := (E_L \mid E_R)$ can be expressed as a linear combination of the r first columns E_L, namely there exists a matrix $A \in \mathbb{F}_q^{r\times(n-r)}$ such that $E_R = E_L \cdot A$. The latter relation provides us the degree-2 polynomial constraints we can use, assuming A is part of the witness (together with x). MIRA [ABB$^+$23d] is another NIST candidate that relies on the MinRank problem, but using another verification circuit (based on q-polynomials). The idea of MIRA is to show that there exists $\beta \in \mathbb{F}_{q^m}^r$ such that $x_j^{q^r} + \sum_{i=0}^{r-1} \beta_i \cdot x_j^{q^i} = 0$ 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 $\cdot \mapsto \cdot^{q^i}$ 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 $x \in \mathbb{F}_q^k$ and of the matrix $A \in \mathbb{F}_q^{r\times(n-r)}$. In MIRA, the witness is composed of $x \in \mathbb{F}_q^k$ and of a vector of $\mathbb{F}_{q^n}^r$ (which represents a monic degree-q^r q-polynomial of $\mathbb{F}_{q^n}[X]$). We thus have that adapting MIRA will lead to larger signature sizes than adapting MiRitH (for the same MinRank parameters). Finally, RYDE [ABB$^+$23c] relies on the syndrome decoding problem in the rank metric which consists, given a matrix $H \in \mathbb{F}_{q^m}^{(n-k)\times n}$ and a vector $y \in \mathbb{F}_{q^m}^{n-k}$, in finding $x \in \mathbb{F}_{q^m}^n$ 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 $\beta \in \mathbb{F}_{q^m}^r$ such that $x_j^{q^r} + \sum_{i=0}^{r-1} \beta_i \cdot x_j^{q^i} = 0$ for all j.

5.6 Short Post-Quantum Ring Signatures

As another application of TCitH-Π_{PC} (Section 5.2), 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

	Original size	Our variant	Saving
Biscuit [BKPV23]	4 758 B	4 048 B	−15%
MIRA [ABB+23d]	5 640 B	5 340 B	−5%
MiRitH-Ia [ABB+23b]	5 665 B	4 694 B	−17%
MiRitH-Ib [ABB+23b]	6 298 B	5 245 B	−17%
MQOM (over $\mathbb{F}_{251}$) [FR23a]	6 575 B	4 257 B	−35%
MQOM (over $\mathbb{F}_{31}$) [FR23a]	6 348 B	4 027 B	−37%
RYDE [ABB+23c]	5 956 B	5 281 B	−11%
SDitH (over $\mathbb{F}_{251}$ & $\mathbb{F}_{256}$) [FJR22, AFG+23]	8 241 B	7 335 B	−11%

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

ring member). The important security property of the scheme (beyond the unforgeability) is the anonymity of the signer. Let us denote j^* the secret index of the signer within the ring. To build a ring signature scheme with our framework, we need to build an MPC protocol that checks that the input sharing $[\![w]\!]$ is the private key which corresponds to one public key of the ring, namely y_{j^*}, while keeping j^* private. A way to proceed is to input to the protocol (in addition to the witness sharing $[\![w]\!]$) a sharing $[\![s]\!]$ of a one-hot encoding $s \in \{0,1\}^r$ of j^*. That is $\forall j \in [1:r]$, $s_j = 1$ if $j = j^*$ and $s_j = 0$ otherwise. Then the MPC protocol starts by securely computing a sharing of the right public key as

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

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

$$s^{(1)}_{j_1^*} = s^{(2)}_{j_2^*} = \cdots = s^{(d)}_{j_d^*} = 1$$

while the other coordinates of the $s^{(j)}$'s equal zero, where $(j_1^*, \ldots, j_d^*)$ is the base-$\sqrt[d]{r}$ decomposition of j^* "shifted by one" (which means that $(j_1^* - 1, \ldots, j_d^* - 1)$ is the standard base-$\sqrt[d]{r}$ decomposition of $j^* - 1$; we use this translation because vector indices start from 1). Here d is a parameter of the scheme that we can tune to optimize the signature size. Then we can compute a sharing of the standard one-hot encoding $[\![s]\!]$ from the sharings $[\![s^{(1)}]\!], \ldots, [\![s^{(d)}]\!]$ by

$$\forall j,\ [\![s_j]\!] = [\![s^{(1)}_{j_1}]\!] \times \ldots \times [\![s^{(d)}_{j_d}]\!]\ , \tag{13}$$

with $(j_1, \ldots, j_d)$ the base-$\sqrt[d]{r}$ decomposition of j "shifted by one". Computing $[\![s]\!]$ with the above method involves a large number of multiplications, but fortunately, using of our proof system of Section 5.2, 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 $\sqrt[d]{r} - 1$ zeros and 1 one. To this purpose, we further extend the input of the protocol with secret values $\{p_j\}_j$ encoding the position of the non-zero coordinates in the vectors $\{s^{(j)}\}_j$:

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

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

$$\forall j \in [1:d],\ \forall k \in [1:\sqrt[d]{r}],\ s^{(j)}_k \cdot (\gamma_k - p_j) = 0, \tag{14}$$

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

$$\sum_{j=1}^{r} s_j = 1\ , \tag{15}$$

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 (13);
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 (14) and (15).

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 5.5 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

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

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

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

All these polynomials have degrees at most $\max\{d, 2\}$, and we can apply the proof system of Section 5.2 (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:

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

All these polynomials have degrees at most $d + 1$, and we can apply the proof system of Section 5.2 (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 (splitted 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 $\mathsf{pk} := (x, \mathrm{AES}_{\mathsf{sk}}(x))$, we also include the signer's public key in the witness. The polynomial constraints are split in two categories: ones 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 $\mathsf{pk} := (x, \mathrm{AES}_x(\mathsf{sk}) \oplus \mathsf{sk})$, we shall use the same AES key x for all the users of the ring so that only $\mathrm{AES}_x(\mathsf{sk}) \oplus \mathsf{sk}$ shall differ between the different public keys. With this variant, we only include the block-cipher output $\mathrm{AES}_x(\mathsf{sk}) \oplus \mathsf{sk}$ 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 11. 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 againts 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 4 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:

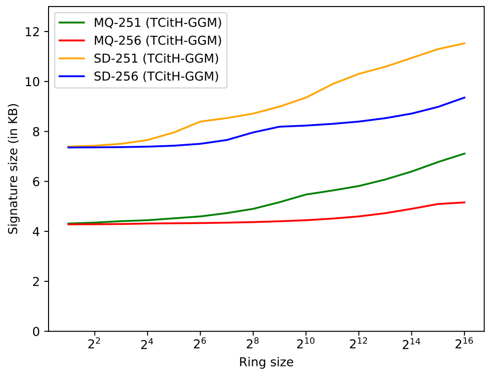
- when using $\mathbb{F}_{251}$ (no subfield), we need to share the coordinates of $s^{(1)}, \dots, s^{(d)}$ over $\mathbb{F}_{251}$. Sending such a share costs $\log_2(251)$ bits per coordinate.
- when using $\mathbb{F}_{256}$ (for which the smallest subfield is $\mathbb{F}_2$), we can share the coordinates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

5.7 Exact Zero-Knowledge Arguments for Lattices

We now present an application of our generic proof system TCitH-Π_{LPPC} (Section 5.3) to obtain short and exact zero-knowledge arguments for lattice problems. Previous tries using the MPC-in-the-Head paradigm



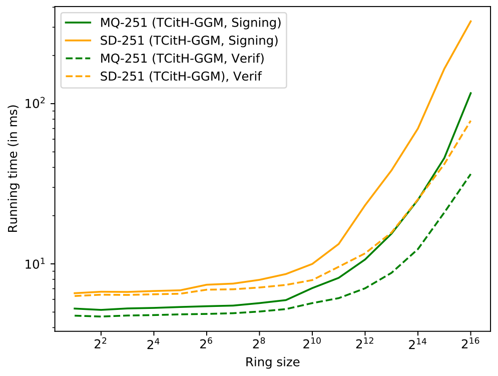


Fig. 4: Benchmark of the proposed ring signature schemes

Table 6: Signature size (KB) for different post-quantum ring signature schemes.

#users		2^3	2^6	2^8	2^{10}	2^{12}	2^{20}	Assumption	Security
Our scheme	2023	4.41	4.60	4.90	5.48	5.82	8.19	MQ over $\mathbb{F}_{251}$	NIST I
Our scheme	2023	4.30	4.33	4.37	4.45	4.60	5.62	MQ over $\mathbb{F}_{256}$	NIST I
Our scheme	2023	7.51	8.40	8.72	9.36	10.30	12.81	SD over $\mathbb{F}_{251}$	NIST I
Our scheme	2023	7.37	7.51	7.96	8.24	8.40	10.09	SD over $\mathbb{F}_{256}$	NIST I
Our scheme	2023	7.87	7.90	7.94	8.02	8.18	9.39	AES128	NIST I
Our scheme	2023	6.81	6.84	6.88	6.96	7.12	8.27	AES128-EM	NIST I
KKW [KKW18]	2018	-	250	-	-	456	-	LowMC	NIST V
GGHK [GGHAK22]	2021	-	-	-	56	-	-	LowMC	NIST V
Raptor [LAZ19]	2019	10	81	333	1290	5161	-	MSIS / MLWE	100 bit
EZSLL [EZS+19]	2019	19	31	-	-	148	-	MSIS / MLWE	NIST II
Falafl [BKP20]	2020	30	32	-	-	35	-	MSIS / MLWE	NIST I
Calamari [BKP20]	2020	5	8	-	-	14	-	CSIDH	128 bit
LESS [BBN+22]	2022	11	14	-	-	20	-	Code Equiv.	128 bit
MRr-DSS [BESV22]	2022	27	36	64	145	422	-	MinRank	NIST I

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-Π_{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 $(e^{(0)}, \dots, e^{(k-1)})$ of $\{0,1\}^n$ such that

$$e = \sum_{i=0}^{k-2} 2^i e^{(i)} + (2\beta - 2^{k-1} + 1)e^{(k-1)} - \boldsymbol{\beta} \tag{16}$$

where $\boldsymbol{\beta} = (\beta, \dots, \beta) \in \mathbb{F}_q^n$. 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 (16). This can be expressed as an instance of the Π_{LPPC} protocol (Section 5.3) with the following global linear constraints:

$$\underbrace{A \cdot \big(I_n \mid 2I_n \mid \cdots \mid 2^{k-2}I_n \mid (2\beta - 2^{k-1} + 1)I_n\big) \cdot \begin{pmatrix} e^{(0)} \\ \vdots \\ e^{(k-1)} \end{pmatrix}}_{e+\boldsymbol{\beta}} = t + A \cdot \boldsymbol{\beta}$$

and the following parallel polynomial constraints:

$$\forall j \in \{0, \dots, k-1\},\ e^{(j)} \circ (e^{(j)} - \mathbf{1}) = \mathbf{0}\ ,$$

where $\mathbf{0} = (0, \dots, 0) \in \mathbb{F}_q^n$, $\mathbf{1} = (1, \dots, 1) \in \mathbb{F}_q^n$ and $\circ$ is the coordinate-wise multiplication. By applying the TCitH-Π_{LPPC} proof system of Section 5.2, with the variant TCitH-MT, the size of the proof transcript is given by Equation (12), where the degree d is 2 and the packed witness size is n/s.

We apply this proof system on toy ISIS instances with $q \approx 2^{32}$ and $q \approx 2^{61}$ from the lattice zero-knowledge literature as well as on Kyber [ABD+21] and Dilithium [BDK+21a] 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 than [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 and 15 KB *vs.* 41 KB) but either rely on NTT-friendly fields [LNS21] or prove the L_2 norm [LNP22] (instead of proving the L_∞ norm as the other schemes). Regarding the Kyber512 instance, our size (23 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_∞ in our case).

ISIS/LWE Instance	q	n	m	β	N	τ	ℓ	s	η	ρ	Proof Size
Toy Example 1	$\approx 2^{32}$	2048	1024	1	1024	1	43	32	28	4	41 386 B
Toy Example 2	$\approx 2^{61}$	4096	512	1/2	1024	1	47	41	17	3	62 139 B
Kyber512	3329	$(2+2) \times 256$	2×256	3	1024	1	37	19	64	11	23 325 B
Kyber768	3329	$(3+3) \times 256$	3×256	2	1024	1	39	23	68	11	27 517 B
Kyber1024	3329	$(4+4) \times 256$	4×256	2	1024	1	41	27	72	11	31 250 B
Dilithium2	8380417	$(4+4) \times 256$	4×256	2	1024	1	43	32	39	6	43 578 B
Dilithium3	8380417	$(6+5) \times 256$	6×256	4	1024	1	45	37	41	6	61 537 B
Dilithium5	8380417	$(8+7) \times 256$	8×256	2	1024	1	45	37	41	6	62 024 B

Table 7: Proof sizes when applied on some SIS/LWE instances.

Remark 12. 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 a 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 prameters for $q = 251$ and $q = 1021$ ($\beta = 1/2$ means that the small vectors are binary). For the latter, we achieve argument size below 17 KB.

Scheme	Year	Any q	Toy Example 1		Toy Example 2	
			Proof Size	Rej. Rate	Proof Size	Rej. Rate
[LNSW13]	2013	✓	3600 KB	0	8988 KB	0
Ligero [AHIV17]	2017	q prime + NTT	157 KB	0	-	-
Aurora [BCR+19]	2019	q prime + NTT	71 KB	0	-	-
[BLS19]	2019	q prime + NTT	384 KB	0.92	-	-
[BN20]	2020	q prime	-	-	4077 KB	0
[Beu20]	2020	q prime	233 KB	0	444 KB	0
[ENS20]	2020	q prime + NTT	47 KB	0.95	-	-
[LNS21]	2021	q prime + NTT	33.3 KB	0.85	-	-
[FMRV22] (batching)	2022	✓	291 KB	0.04	291 KB	0.04
[FMRV22] (C&C)	2022	✓	184 KB	0.05	184 KB	0.05
[LNP22]*	2022	q prime	14.7 KB	0.86	-	-
Our scheme	2023	q prime	41 KB	0	62 KB	0

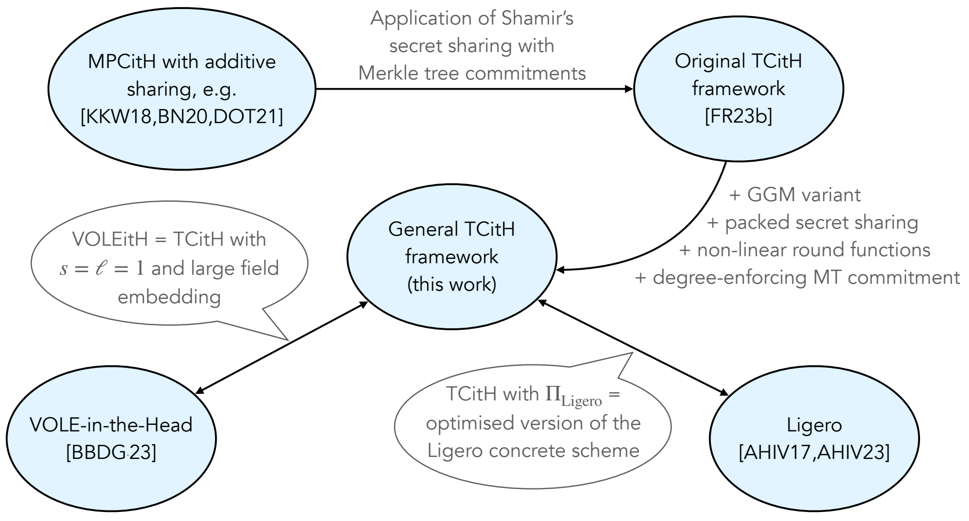
Table 8: Comparison with the existing exact protocols which prove the knowledge of the solution of a ISIS instance. ($\star$): All the schemes prove the infinity norm, except [LNP22] that proves the L_2-norm.

LWE Instance	q	n	m	β	N	τ	ℓ	s	d_β	η	ρ	Proof Size
LWE over $\mathbb{F}_{1021}$	1021	561	561	1/2	1000	1	34	11	44	65	13	16 944 B
LWE over $\mathbb{F}_{251}$	251	473	473	1/2	240	4	10	6	17	39	17	18 583 B

Table 9: Proof sizes for LWE instances.

6 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 instanciations and these proof systems. Those connections are illustrated in Figure 5.



MPCitH with additive sharing, e.g. [KKW18,BN20,DOT21]
Application of Shamir's secret sharing with Merkle tree commitments
Original TCitH framework [FR23b]
+ GGM variant
+ packed secret sharing
+ non-linear round functions
+ degree-enforcing MT commitment
VOLEitH = TCitH with $s = \ell = 1$ and large field embedding
General TCitH framework (this work)
VOLE-in-the-Head [BBDG 23]
TCitH with Π_{Ligero} = optimised version of the Ligero concrete scheme
Ligero [AHIV17,AHIV23]

Fig. 5: The TCitH framework in the landscape of MPCitH-like proof systems.

6.1 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 $u \in \mathbb{F}_p^\ell, v \in \mathbb{F}_{p^k}^\ell$, while the verifier learns a random $\Delta \in \mathbb{F}_{p^k}$ and $q = u \cdot \Delta + v \in \mathbb{F}_{p^k}^\ell$. The key idea of the VOLEitH construction consists in proposing a method to commit a random vector u through a VOLE for which the verifier will be enabled to get a VOLE correlation $(\Delta, q = u \cdot \Delta + v)$. The used technique consists in transforming a vector commitment with all-but-one opening into a VOLEitH correlation. We can observe that a VOLE gadget can be seen as a (non-packed) $(\ell = 2, N)$-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,

- 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 ith 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-ℓ polynomial such that

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

In that case, if we restrict Equation (4) with $\ell = 1$, we obtain (up to the auxiliary value)

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

which exactly corresponds to the formulae from [BBdSG$^+$23] (see Equation 2 in [BBdSG$^+$23] for example). Once we made this observation (committing a VOLE gadget is equivalent to committing a $(2, N)$-threshold Shamir's secret sharing), we may wonder what would be the equivalent MPC protocol used in [BBdSG$^+$23]. In fact, [BBdSG$^+$23] relies on a variant of the QuickSilver VOLE-based protocol [YSWW21], and it can be equivalent to the MPC protocol Π_{PC} of Section 5.2 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 Π_{PC}.

Large field embedding. Compared to the TCitH-GGM-Π_{PC} 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 be also applied to the TCitH framework. Let us assume that we have τ sharings

$[\![u]\!]^{(1)}, \ldots, [\![u]\!]^{(\tau)}$ of the same value $u \in \mathbb{F}$. We denote $P_u^{(1)}(X) := u \cdot X + v^{(1)}, \ldots, P_u^{(\tau)}(X) := u \cdot X + v^{(\tau)}$ the corresponding Shamir's polynomials (while assuming the shared value is stored to the infinity point). Let us consider an isomorphism ϕ between $\mathbb{F}_q^\tau$ and $\mathbb{F}_{q^\tau}$. Then, the N^τ-sharing $[\![u]\!]^{(\phi)}$ defined as

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

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

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

Indeed, we have

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

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

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. 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 [BBD+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 $\mathbb{F}_{251^2}$ while VOLEitH run the MPC protocol only once, but in the field extension $\mathbb{F}_{251^{24}}$. 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 5.6.

Let us stress that the large field embedding optimization of VOLEitH only support Shamir's secret sharings with $\ell = s = 1$ in the TCitH framework. It cannot be used with $\ell > 1$ or with packed sharings. However, while this optimization is presented in a context relying on GGM trees, it could also be used in the TCitH-MT variant (with $\ell = 1$ and $s = 1$), thus providing an additional trade-off. In a non-packed context, with $\ell = s = 1$, the TCitH-MT variant comes with a penalty in the proof size compared to the TCitH-GGM variant but with the benefit of a fast verification. While this trade-off seems in conflict with the trade-off offered large field embedding optimization, there could exist particular applications for which considering the optimization with TCitH-MT would be of interest (which we leave as an open question).

6.2 Connections to Ligero

As in Ligero, our framework with packed secret sharing, 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 / 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.

	TCitH-GGM		VOLEitH	
	Size	Comput. Field	Size	Computat. Field
AImer [CCH+23]	4 352 B	19× $GF(2^8)$	3 938 B	$GF(2^{128})$
Biscuit [BKPV23]	4 048 B	19× $GF(16^2)$	3 682 B	$GF(16^{2\times 16})$
MIRA [ABB+23d]	5 340 B	19× $GF(16^2)$	4 770 B	$GF(16^{2\times 16})$
MiRitH-Ia [ABB+23b]	4 694 B	19× $GF(16^2)$	4 226 B	$GF(16^{2\times 16})$
MiRitH-Ib [ABB+23b]	5 245 B	19× $GF(16^2)$	4 690 B	$GF(16^{2\times 16})$
MQOM (over $\mathbb{F}_{251}$) [FR23a]	4 257 B	19× $GF(251)$	3 858 B	$GF(251^{16})$
MQOM (over $\mathbb{F}_{31}$) [FR23a]	4 027 B	19× $GF(31^2)$	3 660 B	$GF(31^{2\times 16})$
RYDE [ABB+23c]	5 281 B	19× $GF(2^8)$	4 720 B	$GF(2^{128})$
		19× $GF(2^{31})$		
SDitH (over $\mathbb{F}_{251}$) [AFG+23]	7 335 B	19× $GF(251)$	6 450 B	$GF(251^{16})$
SDitH (over $\mathbb{F}_{256}$) [AFG+23]	7 335 B	19× $GF(256)$	6 450 B	$GF(256^{16})$

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

	TCitH-GGM		VOLEitH	
	Size	Comput. Field	Size	Computat. Field
AIMer [CCH+23]	3 639 B	13× $GF(2^{11})$	3 546 B	$GF(2^{128})$
Biscuit [BKPV23]	3 431 B	13× $GF(16^3)$	3 354 B	$GF(16^{3\times 12})$
MIRA [ABB+23d]	4 314 B	13× $GF(16^3)$	4 170 B	$GF(16^{3\times 12})$
MiRitH-Ia [ABB+23b]	3 873 B	13× $GF(16^3)$	3 762 B	$GF(16^{3\times 12})$
MiRitH-Ib [ABB+23b]	4 250 B	13× $GF(16^3)$	4 110 B	$GF(16^{3\times 12})$
MQOM (over $\mathbb{F}_{251}$) [FR23a]	3 567 B	13× $GF(251^2)$	3 486 B	$GF(251^{2\times 12})$
MQOM (over $\mathbb{F}_{31}$) [FR23a]	3 418 B	13× $GF(31^3)$	3 338 B	$GF(31^{3\times 12})$
RYDE [ABB+23c]	4 274 B	13× $GF(2^{11})$	4 133 B	$GF(2^{128})$
		13× $GF(2^{31})$		
SDitH (over $\mathbb{F}_{251}$) [AFG+23]	5 673 B	13× $GF(251^2)$	5 430 B	$GF(251^{2\times 12})$
SDitH (over $\mathbb{F}_{256}$) [AFG+23]	5 673 B	13× $GF(256^2)$	5 430 B	$GF(256^{2\times 12})$

Table 11: Comparison between TCitH-GGM and VOLEitH using GGM trees of 2048 leaves.

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 the 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 $\binom{d_\alpha}{\ell}/\binom{N}{\ell}$, where d_α 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_α and t_r in this regime.

Then, Ligero further provides a concrete zero-knowledge proof system for arithmetic circuits. As shown in Section 5.4, our framework can be applied to the MPC protocol Π_{Ligero} which is an optimized version of the Ligero protocol to check an arithmetic circuit. The obtained specific instantiation of our framework, TCitH-Π_{Ligero}, is close to this concrete proof system but achieves smaller proof sizes (see Figures 3a and 3b) for circuits with $\leq 2^{16}$ multiplication gates.

One key ingredient to this improvement is our degree-enforcing sharing commitment scheme described in Section 4.2 and the related soundness analysis (see the proof of Theorem 1). 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 guaranty leaves much 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 sharing commitment 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 instanciations).

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

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

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

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

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

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

ABD$^+$21. Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crypstals-kyber – algorithm specifications and supporting documentation. Version 3.02 – August 4, 2021, 2021. `https://pq-crystals.org/kyber/data/kyber-specification-round3-20210804.pdf`.

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

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

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

AHIV23. Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: lightweight sublinear arguments without a trusted setup. *DCC*, 91(11):3379–3424, 2023.

APS15. Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.

ARZV23. Gora Adj, Luis Rivera-Zamarripa, and Javier A. Verbel. MinRank in the head - short signatures from zero-knowledge proofs. In Nadia El Mrabet, Luca De Feo, and Sylvain Duquesne, editors, *AFRICACRYPT 23*, volume 14064 of *LNCS*, pages 3–27. Springer Nature, July 2023.

BBD$^+$23. Carsten Baum, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß, Christian Majenz, Shibam Mukherjee, Emmanuela Orsini, Sebastian Ramacher, Christian Rechberger, Lawrence Roy, and Peter Scholl. FAEST: Algorithm Specifications – Version 1.1, 2023. `https://faest.info/faest-spec-v1.1.pdf`.

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

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

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

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

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

BCR$^+$19. Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 103–128. Springer, Heidelberg, May 2019.

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

BDK$^+$21a. Shi Bai, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crypstals-dilithium – algorithm specifications and supporting documentation. Version 3.1 – February 8, 2021, 2021. `https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf`.

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

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

Beu20. Ward Beullens. Sigma protocols for MQ, PKP and SIS, and Fishy signature schemes. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 183–211. Springer, Heidelberg, May 2020.

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

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

BLS19. Jonathan Bootle, Vadim Lyubashevsky, and Gregor Seiler. Algebraic techniques for short(er) exact lattice-based zero-knowledge proofs. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 176–202. Springer, Heidelberg, August 2019.

BN20. Carsten Baum and Ariel Nof. Concretely-efficient zero-knowledge arguments for arithmetic circuits and their application to lattice-based cryptography. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 495–526. Springer, Heidelberg, May 2020.

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

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

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

ENS20. Muhammed F. Esgin, Ngoc Khanh Nguyen, and Gregor Seiler. Practical exact proofs from lattices: New techniques to exploit fully-splitting rings. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 259–288. Springer, Heidelberg, December 2020.

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

FJR22. Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 541–572. Springer, Heidelberg, August 2022.

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

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

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

FR23a. Thibauld Feneuil and Matthieu Rivain. MQOM: MQ on my Mind – Algorithm Specifications and Supporting Documentation. Version 1.0 – 31st May 2023, 2023. `https://mqom.org/docs/mqom-v1.0.pdf`.

FR23b. Thibauld Feneuil and Matthieu Rivain. Threshold Linear Secret Sharing to the Rescue of MPC-in-the-Head. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings, Part I*, volume 14438 of *Lecture Notes in Computer Science*, pages 441–473. Springer, 2023.

FY92. Matthew K. Franklin and Moti Yung. Communication complexity of secure computation (extended abstract). In *24th ACM STOC*, pages 699–710. ACM Press, May 1992.

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

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

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

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

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

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

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

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

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

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

LNP22. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plançon. Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 71–101. Springer, Heidelberg, August 2022.

LNS21. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Shorter lattice-based zero-knowledge proofs via one-time commitments. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 215–241. Springer, Heidelberg, May 2021.

LNSW13. San Ling, Khoa Nguyen, Damien Stehlé, and Huaxiong Wang. Improved zero-knowledge proofs of knowledge for the ISIS problem, and applications. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 107–124. Springer, Heidelberg, February / March 2013.

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

Sha79. Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.

Wan22. William Wang. Shorter signatures from MQ. Cryptology ePrint Archive, Report 2022/344, 2022. `https://eprint.iacr.org/2022/344`.

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

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

A Proof of Theorem 1

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

Theorem 1. *Let Π be an MPC protocol of parameters $(N, \ell, s, d_\alpha, d_\beta, p)$ complying to the format of Protocol 3. In particular, Π is ℓ-private in the semi-honest model and of false positive probability p. Then, Protocol 5 built from Π satisfies the three following properties:*

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

– ***Soundness.*** *Suppose that there is an efficient prover $\tilde{\mathcal{P}}$ that, on input x, convinces the honest verifier $\mathcal{V}$ to accept with probability*

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

where the soundness error ϵ is defined as

$$\epsilon := \begin{cases} p + (1-p) \cdot \frac{\binom{d_\alpha}{\ell}}{\binom{N}{\ell}} & \text{for the TCitH-GGM variant,} \\ p + (1-p) \cdot \frac{\binom{d_\alpha}{\ell}}{\binom{N}{\ell}} + \frac{\binom{N}{d_\beta+1}^2}{|\mathbb{F}|^\eta} & \text{for the TCitH-MT variant.} \end{cases} \tag{17}$$

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

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

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

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

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

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

– A false positive event occurs: the MPC randomness $\varepsilon^1, \ldots, \varepsilon^t$ sampled by the verifier is such that an honest execution of the MPC protocol produces the output Accept. By definition of the considered MPC protocol Π, this case occurs with probability p.

– No false positive event occurs. In that case, we show below that the last challenge-response round of the zero-knowledge protocol can be seen as a $\left(\binom{d_\alpha}{\ell} + 1\right)$-special-sound protocol. The probability that the verifier is convinced after the last round is thus $\binom{d_\alpha}{\ell}/\binom{N}{\ell}$, since $\binom{N}{\ell}$ is the size of the challenge space.

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

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

We now show how to upper bound $\Pr[\mathsf{Succ} \mid \neg\mathsf{FP}]$ 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 $\big(\binom{d_\alpha}{\ell}+1\big)$-special-sound protocol. Namely, we will show that if $\binom{d_\alpha}{\ell}+1$ different accepting transcripts can be obtained from the malicious prover which only differ in their last round of challenge-response (opening of the parties), then a valid witness w can be extracted from these transcripts, leading to a contradiction. This shall imply that at most $\binom{d_\alpha}{\ell}$ valid transcripts can be obtained from the malicious prover if no false positive event occurs, implying $\Pr[\mathsf{Succ} \mid \neg\mathsf{FP}] \leq \binom{d_\alpha}{\ell}/\binom{N}{\ell}$.

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

$$\mathcal{T}_k := (\underbrace{h_1, \varepsilon^1, \ldots, h_t, \varepsilon^t, h_{t+1}}_{\text{Same for all transcripts}}, I_k, \underbrace{\sigma_k := (\mathsf{views}_{I_k}, \mathsf{decom}_{I_k}, \{[\![\alpha^j]\!]_{S_k^j}\}_{i\in[1:t]})}_{\text{Prover's last response}})$$

where $\mathsf{views}_{I_k} = (\mathsf{path}_{I_k}, \Delta w, \Delta\beta^1, \ldots, \Delta\beta^t)$ (the Δ's being omitted if $I_k = T_0$), $\mathsf{decom}_{I_k} = \mathsf{com}_{I_k} = \mathrm{Com}(\mathsf{seed}_{I_k}; \rho_{I_k})$ and S_k^j denotes some set of cardinality $|S_k^j| = d_{\alpha[}j] + 1 - \ell$ disjoint of I_k (as defined in Step 7 of Protocol 5). From the revealed $\{[\![\alpha^j]\!]_{S_k^j}\}_j$ 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

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

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

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

(In the above equation Φ^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 Accept on the witness w. Since we assumed that there were no false positive events, we get that w must be a valid witness, which concludes the proof.

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

Let $J \subseteq [1:N]$ such that $|J| = d_\beta + 1$, where $d_\beta = \max(s+\ell-1, d_1, \dots, d_t)$. We shall denote $Q^{(J)}$, resp. $P_u^{(J)}$, the polynomial obtained by interpolation of $\{Q(e_i)\}_{i\in J}$, resp. $\{P_u(e_i)\}_{i\in J}$, for Q defined in Equation (9) (where both $Q(e_i)$ and $P_u(e_i)$ are defined with respect to the shares $[\![w]\!]_i, [\![u]\!]_i, [\![\beta^1]\!]_i, [\![\bar\beta^2]\!]_i, \dots, [\![\bar\beta^t]\!]_i$). We stress that if the committed sharings are valid Shamir's secret sharings of their respective degrees (meaning that the polynomials P_w, P_u, P_{β_1}, $\dots$, P_{β_t} are of degrees $s+\ell-1$, d_β, d_1, $\dots$, d_t respectively), all the polynomials $Q^{(J)}$ are equal to Q. But by committing invalid sharings (sharings of higher degrees), one ends up with different polynomials $Q^{(J)}$. We shall further denote $P_\xi^{(J)} = \Gamma \cdot Q^{(J)} + P_u^{(J)}$.

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

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

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

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

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

for Coll the event defined as

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

The soundness error of the protocol then satisfies:

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

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

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

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

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

We get that the soundness error satisfies:

$$\begin{aligned}\epsilon \;&\le\; \Pr[\mathsf{FP}] + \Pr[\neg\mathsf{FP}]\cdot\Pr[\mathsf{Succ} \mid (\neg\mathsf{FP})\wedge(\neg\mathsf{Coll})] + \Pr[\mathsf{Coll}] \\ &\le\; p + (1-p)\cdot\frac{\binom{d_\alpha}{\ell}}{\binom{N}{\ell}} + \frac{\binom{N}{d_\beta+1}^2}{2|\mathbb{F}|^\eta}\ . \qquad (18)\end{aligned}$$

We now show how to upper bound $\Pr[\mathsf{Succ} \mid (\neg\mathsf{FP})\wedge(\neg\mathsf{Coll})]$ 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 $\left(\binom{d_\alpha}{\ell}+1\right)$-special-sound protocol. Namely, we will show that if $\binom{d_\alpha}{\ell}+1$ different accepting transcripts can be obtained from the malicious prover which only differ in their last round of challenge-response (opening of the parties), then a valid witness w can be extracted from these transcripts, leading to a contradiction. This shall imply that at most $\binom{d_\alpha}{\ell}$ valid transcripts can be obtained from the malicious prover if no false positive event occurs, implying $\Pr[\mathsf{Succ} \mid (\neg\mathsf{FP})\wedge(\neg\mathsf{Coll})] \le \binom{d_\alpha}{\ell}/\binom{N}{\ell}$.

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

$$\mathcal{T}_k := \big(\underbrace{h_1, \Gamma, h_1', \varepsilon^1, \dots, h_t, \varepsilon^t, h_{t+1}}_{\text{Same for all transcripts}}, I_k, \underbrace{\sigma_k := (\mathsf{views}_{I_k}, \mathsf{decom}_{I_k}, \{[\![\alpha^j]\!]_{S_k^j}\}_{i\in[1:t]})}_{\text{Prover's last response}}\big)$$

where $\mathsf{views}_{I_k} = \left(\{[\![w]\!]_i, [\![\beta^1]\!]_i, [\![\bar\beta^2]\!]_i, \dots, [\![\bar\beta^t]\!]_i\}_{i\in I_k}, \Delta\beta^2, \dots, \Delta\beta^t\right)$, $\mathsf{decom}_{I_k} = \mathsf{path}_{I_k}$ and S_k^j denotes some set of cardinality $|S_k^j| = d_{\alpha[}j] + 1 - \ell$ disjoint of I_k (as defined in Step 7 of Protocol 5). From these transcript, since $I_k \ne I_{k'}$ for all $k \ne k'$, we can retrieve at least $d_\alpha + 1 \ge d_\beta + 1$ shares of the committed sharings $[\![w]\!]$, $[\![u]\!]$, $[\![\beta^1]\!]$, $[\![\bar\beta^2]\!]$, …, $[\![\bar\beta^t]\!]$ (which pass the degree-enforcement test, *i.e.* which are consistent with P_ξ). From these shares, we can build the Shamir's polynomials $P_w, P_u, P_{\beta^1}, \dots, P_{\beta^t}$ by interpolation. The rest of the proof is similar to the GGM case. Since $\{\mathcal{T}_k\}_k$ are valid transcripts, it means that, for all $i \in \bigcup_k I_k$ and $j \in [1:t]$, we have

$$P_{\alpha^j}(e_i) = \Phi^j\left(P_w(e_i), (P_{\beta^k}(e_i))_{k\le j}\right)$$

since $[\![\alpha^j]\!]_i = P_{\alpha^j}(e_i)$, $[\![w]\!]_i = P_w(e_i)$ and $[\![\beta^k]\!]_i = P_{\beta^k}(e_i)$. Since the polynomials in the above relation are of degree at most d_α and since the relations hold for at least $d_\alpha + 1$ evaluation points $\{e_i\}_{\bigcup_k I_k}$, we have that the relations hold directly on the polynomials: $P_{\alpha^j} = \Phi^j\left(P_w, (P_{\beta^k})_{k\le j}\right)$ for all j. In particular, the relations are true when evaluating the polynomials in ω_1, …, ω_s, the evaluation points revealing the packed secrets, which implies

$$\alpha^j = \Phi^j\left(w, (\beta^k)_{k\le j}\right)\ .$$

(In the above equation Φ^j applies independently on each “slot” of the s-tuples composing the plain values.) Moreover, since $\{\mathcal{T}_k\}_k$ are valid, we have

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

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