Communication-Efficient Multi-Party Computation for RMS Programs

Thomas Attema[1,2], Aron van Baarsen[2,3], Stefan van den Berg[1], Pedro Capitão[2,3], Vincent Dunning[1] and Lisa Kohl[2]

[1] TNO, Applied Cryptography and Quantum Algorithms, The Hague, The Netherlands
[2] CWI, Cryptology Group, Amsterdam, The Netherlands
[3] Leiden University, Mathematical Institute, Leiden, The Netherlands

Abstract. Despite much progress, general-purpose secure multi-party computation (MPC) with active security may still be prohibitively expensive in settings with large input datasets. This particularly applies to the secure evaluation of graph algorithms, where each party holds a subset of a large graph.
Recently, Araki et al. (ACM CCS '21) showed that dedicated solutions may provide significantly better efficiency if the input graph is sparse. In particular, they provide an efficient protocol for the secure evaluation of "message passing" algorithms, such as the PageRank algorithm. Their protocol's computation and communication complexity are both $\tilde{O}(M \cdot B)$ instead of the $O(M^2)$ complexity achieved by general-purpose MPC protocols, where M denotes the number of nodes and B the (average) number of incoming edges per node. On the downside, their approach achieves only a relatively weak security notion: 1-out-of-3 malicious security with selective abort.
In this work, we show that PageRank can instead be captured efficiently as a *restricted multiplication straight-line* (RMS) program, and present a new actively secure MPC protocol tailored to handle RMS programs. In particular, we show that the local knowledge of the participants can be leveraged towards the first maliciously-secure protocol with communication complexity linear in M, independently of the sparsity of the graph. We present two variants of our protocol. In our communication-optimized protocol, going from semi-honest to malicious security only introduces a small communication overhead, but results in quadratic computation complexity $O(M^2)$. In our balanced protocol, we still achieve a linear communication complexity $O(M)$, although with worse constants, but a significantly better computational complexity scaling with $O(M \cdot B)$. Additionally, our protocols achieve security with identifiable abort and can tolerate up to $n-1$ corruptions.

Keywords: Multi-Party Computation · Active Security · RMS Programs · Zero Knowledge Proofs

1 Introduction

Secure multi-party computation (MPC) [Yao86, GMW87, BGW88, CCD88, RB89] allows analysing distributed data without revealing anything but the outcome of the analysis. Traditionally, security in MPC protocols is considered relative to two types of adversaries: *passive* and *active* adversaries. In the passive model, all parties are expected to follow the protocol honestly, whereas the active (and more realistic) model also captures adversaries which send incorrect messages or stop sending messages altogether.

E-mail: thomas.attema@tno.nl (Thomas Attema), aronvanbaarsen@gmail.com (Aron van Baarsen), stefan.vandenberg@tno.nl (Stefan van den Berg), pedro.capitao@cwi.nl (Pedro Capitão), vincent.dunning@tno.nl (Vincent Dunning), lisa.kohl@cwi.nl (Lisa Kohl)

Date of this document: 2024-10-04.

RMS programs. In this work, we focus on evaluating *restricted multiplication straight-line (RMS) programs*. Similar to arithmetic circuits, programs of this type consist of addition and multiplications gates. However, in contrast to arithmetic circuits, multiplications may only be performed if one of the two factors is an input value to the program. In the context of secure computation, this implies that whenever two values are multiplied, one of them is known to at least one of the parties participating in the computation. While every program can be captured by an RMS program with exponential size (in the size of the original program), polynomial-size RMS programs are known to capture the class of polynomial-size branching programs and circuits of constant fan-out and logarithmic depth (i.e., in the complexity class NC^1) [BGI16].

Graph algorithms. [AFO+21] A particular class adhering to the structure of RMS programs are certain graph algorithms [SvHA+19, CSA21, AFO+21, vEDvdB+24]. Here, an analysis is performed on a dataset in a graph structure, where each party knows everything about a part of the graph but should not learn anything about the parts belonging to the other parties. Furthermore, the topology of the entire graph should remain secret. A recent example use case of this is collaborative fraud detection [vEDvdB+24]. In this case, the parties are financial institutions who want to perform some analysis on their joint transaction graph. While each institution has a full view of its own customers, banks are not allowed to share this information due to data protection regulations, and thus need to rely on MPC techniques in order to detect potential high-risk customers more effectively.

Generic approaches. General-purpose protocols, such as the line of work on secret-sharing based protocols with active security in the preprocessing model [BDOZ11, DPSZ12], require the parties to input the full transaction graph to the protocol. Further, in order to ensure correctness, the parties have to jointly perform computation steps, resulting in communication and computation complexity scaling with the size of the transaction graph, and thus quadratic in the number of nodes. Even if the (average) number of outcoming edges in the graph is assumed to be public knowledge, it is not straightforward to improve over the quadratic communication complexity, since it has to remain secret which nodes are connected. As a transaction graph can easily contain millions of bank accounts, generic secret-sharing based protocols become impractical to use in this scenario.

Optimized graph algorithms. To overcome this, Araki et al. [AFO+21] propose a secret-sharing based protocol which performs significantly better for message passing algorithms on sparse graphs. Namely, they achieve a communication complexity of $\tilde{O}(M \cdot B)$, where B is the average number of incoming edges per node. Yet, their communication complexity still scales quasilinear in the *number of edges*, rather than linear in the *number of nodes*. While one could hope to improve the protocol complexity by logarithmic factors, it is inherent for their and other secret-sharing based approaches that the communication complexity scales with the number of edges, rather than with the number of nodes, since the description of the graph (which is of size $O(M \cdot B)$) has to be secret-shared between the parties.

Building on homomorphic encryption, on the other hand, [SvHA+19, vEDvdB+24] show that in the *semi-honest* security model this can be circumvented. The idea behind their approach is that additively homomorphic encryption allows to exploit the "local" information of the parties by simply letting the party that knows the required information perform that part of the computation *locally*, without sharing their local view of the graph to the other parties, resulting in linear communication complexity $O(M)$.

The issue when going from semi-honest to malicious security is that the parties have to commit to their local view of the graph to be able to prove they follow the protocol correctly, removing the advantage of the passive case. In fact, relying on maliciously secure approaches from additively homomorphic encryption, such as Cramer et al. [CDN01], again results in quadratic complexity in the number of nodes, even for sparse graphs.

Table 1: Comparison of various actively secure protocols for T iterations of a (simplified) PageRank/RiskPropagation algorithm, where we omit constant factors and low order terms. Further, we omit dependencies on the number of parties n, since the dependencies are similar for all approaches. Here, each party P_i owns M_i nodes which sum up to the total number of nodes, denoted by M. Furthermore, B is the average number of incoming edges per node with $\overline{B}$ a known upper bound on B (i.e., for sparse graphs $\overline{B}$ is a small constant and for dense graphs $\overline{B}$ equals the number of nodes M). The entries low.comm and low.comp indicate the communication and computation optimized variants of our protocol, respectively. The overall complexity of the actively secure protocol equals **Passive + Overhead Active**.

	Techniques	Corruption Threshold	Communication (messages per party)		Computation (operations per party)		Rounds	Identifiable Abort
			Passive	Overhead Active	Passive	Overhead Active		
[DPSZ12]	SS+MAC	$t \leq n-1$	TM^2	TM^2	TM^2	TM^2	T	no
[CSA21]	Outs. SS	1-out-of-2	TM^2	TM^2	TM^2	TM^2	T	no
[AFO+21]	Outs. SS	1-out-of-3	$TMB\log\overline{B}$	$TMB\log\overline{B}$	$TMB\log\overline{B}$	$TMB\log\overline{B}$	$TB\log\overline{B}$	no
[CDN01]	HE+ZKP	$t \leq \frac{n}{2}-1$	TM_i	TM^2	TM_iB	TM^2	T	yes
low.comm	HE+ZKP	$t \leq n-1$	TM_i	$\log M$	TM_iB	M^2	T	yes
low.comp	HE+ZKP	$t \leq n-1$	TM_i	$M_i+T\log M$	TM_iB	TM	T	yes

1.1 Our Contributions

In this work, we show that building on recent improvements on communication-succinct zero-knowledge proofs (ZKPs) [BCC+16, BBB+18, BCS16, AC20, ACC+22, Att23], we can achieve a communication complexity which scales *linearly* in the number of nodes also in the malicious case. We give a comparison of our approach to related work in Table 1.

In order for the compressed Σ-protocol framework of [ACC+22] to be compatible with the underlying additively homomorphic encryption scheme, we present a new instantiation of compressed Σ-protocols. More precisely:

- We adopt the framework of [ACC+22] to construct vector commitment schemes from additively homomorphic encryption schemes such as Paillier encryption;
- We capture linear statements over additively homomorphic encryptions as a class of "morphisms", and show that the theory of compressed Σ-protocols [AC20, Att23] can be applied to such statements.

Overall, these techniques lead to an improved communication overhead (when going from passive to active security) of just $O(\log M)$ group elements per party, where M is the number of nodes in the graph. We thus obtain the first maliciously secure protocol where the communication complexity scales with the number of nodes rather than the number of edges. We refer to this as the low.comm version of our protocol. Since our construction relies on zero-knowledge proofs, we further achieve security with identifiable abort, thereby disincentivizing malicious behaviour. Finally, our protocol can tolerate the optimal corruption threshold of up to $n-1$ malicious parties.

Since this approach still has a computation complexity of $O(M^2)$, we further present a more balanced variant of the protocol, which we refer to as low.comp, where we exploit the structure of the considered class of graph algorithms.

- We identify a structure common to relevant graph algorithms which we generalize as *vector-RMS programs*.
- We deploy the amortization technique for Σ-protocols (as made explicit in, e.g., [AC20]) to amortize over each level of the identified vector-RMS programs.

We show that these improvements allow us to reduce the computation complexity to $O(M \cdot B)$ group exponentiations per level whenever the vector-RMS program is sufficiently balanced (as is the case in the motivating applications), at the cost of increasing the communication overhead to an additional $O(M)$ group elements. For typical transaction

graphs of millions of nodes, this is a significant improvement over previous approaches where the M^2 scaling in communication quickly becomes infeasible. For the optimized graph approach of [AFO+21], the amount of per-party communication also grows quickly in practice due to the scaling in M instead of only the local nodes M_i. Furthermore, the upper bound $\overline{B}$ leaks information about the topology of the graph, which might be undesirable.

We further show that in the low.comm variant of the protocol, the concrete computational efficiency can be improved by introducing a new *sparse blinding* technique, which we explain in the following and for which we show concrete efficiency improvements by providing an implementation in Section 7.

- We introduce a new blinding mechanism, with which we obtain zero-knowledge for compressed Σ-protocols [AC20, ACC+22] using a single blinding coefficient, rather than a full blinding vector. This is achieved by adding an additional cheap rerandomization step at each round of compression.

We observe that the graphs considered (e.g., stemming from transaction networks) are typically very sparse. More precisely, we show how to exploit the sparseness as follows. Since only one coefficient of the secret sparse vector is blinded, its sparseness is preserved. Further, as the number of exponentiations in a vector exponentiation $\boldsymbol{g}^{\boldsymbol{x}}$ is equal to the Hamming weight of $\boldsymbol{x}$, this strategy significantly reduces the overall number of exponentiations. It should be noted that the aforementioned approach for amortizing in each layer of the vector-RMS program to lower the computation reduces the sparseness of the input vectors significantly and therefore diminishes the improvements gained from the sparse blinding strategy in the low.comp variant. On the other hand, in the low.comm variant, where we amortize over all layers of the program, our techniques are applicable to reduce the quadratic computation overhead in practice when proving knowledge of the opening of the commitment to the entire graph of size M^2.

Since this improvement does not provide an asymptotic change as our other contributions do, we have implemented the new sparse blinding mechanism and measured its performance when proving knowledge of a commitment opening. The results can be found in Section 7. For the transaction graph scenario where each node only makes 3 out of a total of 6 million possible transactions, we can save over 80% of the work of proving knowledge of a commitment opening in the low communication variant. However, we stress that our construction also leads to significant performance gains for less sparse input vectors.

1.2 Related Work

Active security for graph algorithms. In Table 1, we give an estimate for the asymptotic complexities of using generic actively secure MPC protocols ([DPSZ12, CDN01]), existing works for actively secure MPC protocols tailored to graph algorithms ([CSA21, AFO+21]) and our solution, either tailored to communication efficiency (low.comm) or computation efficiency (low.comp), when executing T rounds of (simplified) PageRank/RiskPropagation.

Here it can be seen that for both the generic secret-sharing based solutions of [DPSZ12, CSA21] as well as for the generic homomorphic encryption-based solution of Cramer et al. [CDN01], communication scales quadratically in the number of nodes M, which quickly becomes infeasible for large graphs. Moreover, each party needs to perform in the order of M^2 operations in each round as they need to perform computations on the entire graph. Here we assume that secret shares or encryptions of edges $e_{i,j} \in \{0, 1\}$ are labeled by the corresponding nodes (i, j). Because of this, the parties need to share M encryptions or secret shares for each of the M nodes in order to hide the structure of the graph. Araki et al. [AFO+21] use a more specialized approach in which only non-zero edges need to be secret shared and updates are executed simultaneously for all nodes. In particular, their

solution is able to exploit the sparsity of the input graph if an upper bound $\overline{B}$ on the in-degree of each node is known to the computing parties.

Both solutions tailored to graph algorithms [CSA21, AFO+21] rely on the idea of decoupling the input parties from the computing servers by letting the parties secret share their input graph to the servers. This approach supports a more general class of functions than we do, but cannot exploit the local structure of RMS programs. Moreover, it allows for a better scaling in the number of input parties. This is beneficial in a scenario where a large number of input parties, possibly with limited resources, each hold a small portion of the graph dataset. However, this setting provides a weaker security guarantee since it requires a majority of the computing servers to be honest and there is no way for the input-providing parties to verify that the computation was done correctly or the servers did not collude to steal the inputs/outputs of the computation.

By contrast, our solution is able to exploit the local nature as well as the sparsity of these graphs since each party only needs to perform updates on its own portion of the transaction graph. Furthermore, using the techniques from [AC20, ACC+22], the parties can prove they performed the updates correctly with only logarithmic communication overhead in the number of nodes, while unfortunately suffering quadratic computation overhead. We also present a more balanced trade-off which increases the communication overhead to linear while achieving linear computation overhead. Notably, the communication of our protocols does not scale with the density of the graph (e.g., $B \cdot M$ or M^2) but only with the number of nodes M. Our approach is best suited for the scenario where there is a smaller number of input parties, with enough resources, each holding a large portion of the input graph. Our solution provides security against a threshold of up to all but one corrupted parties and allows each of the parties to verify that computations were done correctly. This leads to our protocol realizing security with identifiable abort, meaning that a cheating party can be identified and therefore excluded from future computations. If a public bulletin board is used, our solution even allows external parties to verify the computations were done correctly or to identify a cheating party.

Comparison with HSS for RMS programs. The line of work on homomorphic secret sharing (HSS) [BGI16, DHRW16, BCG+17, FGJS17, OSY21, RS21, BKS19, COS+22] also gives dedicated MPC solutions for RMS programs. We note though that this approach is less suitable for the considered setting, since the share sizes scale with the size of the input (even for passive security), again resulting in quadratic communication overhead. Further, HSS operations are computationally more expensive, and all approaches not building on fully-homomorphic encryption are limited to the 2-party setting. An advantage of building on HSS compared to our approach is the small round complexity: there the parties do not have to communicate during the computation, whereas our protocol requires a round of communication for each level of the RMS program.

1.3 Technical Overview

The main construction presented in this work is an *actively secure* MPC protocol for RMS programs, based on an additively homomorphic encryption scheme (AHE).

RMS programs and AHE. An RMS program over a ring $\mathbb{Z}_m$ consists of addition and multiplication gates, with the restriction that "memory values" (those which are output of previous gates) can be multiplied by inputs to the program but not by other memory values. AHE is well-suited to securely evaluate programs of this type, since additions between encrypted values $c_1 := \mathsf{Enc}(\mathsf{pk}, x_1)$ and $c_2 := \mathsf{Enc}(\mathsf{pk}, x_2)$ can be computed by any party as $c_1 * c_2 = \mathsf{Enc}(\mathsf{pk}, x_1 + x_2)$ and multiplications between an encrypted value $c := \mathsf{Enc}(\mathsf{pk}, x)$ and an input value $a \in \mathbb{Z}_m$ can be evaluated locally by the party holding the input as $c^a = \mathsf{Enc}(\mathsf{pk}, a \cdot x)$. This observation was also made in previous works [SvHA+19] in the context of securely evaluating the PageRank algorithm and [vEDvdB+24] for a secure money laundering detection algorithm called "RiskPropagation" on the joint transaction

graph of several financial institutions in the passive setting. We will use the PageRank and RiskPropagation algorithms as leading examples in this overview.

Passively secure evaluation of PageRank/RiskPropagation. The idea of [SvHA+19] to securely evaluate the PageRank algorithm of [BP98] with n parties is as follows. Consider a graph of M nodes of which each party P_i holds a set of nodes $V^i \subset [M]$ and a set of incoming edges $S(j)$ for each $j \in V^i$. First, each party encrypts their initial PageRank values using an additively homomorphic encryption scheme and broadcasts them as z_0^j. Then, in round $t = 1, \ldots, T$, each party computes their updated PageRank values $z_t^j := \prod_{i \in S(j)} z_{t-1}^j$, which can be seen as an inner product between the j-th row $\boldsymbol{a}_j^i$ of their private adjacency matrix A^i and the vector of previous round encrypted PageRank values $\mathbf{z}_{t-1} := (z_{t-1}^j)_{j \in [M]}$. In the final round T, the parties perform a joint distributed decryption, such that each party learns the decryptions of z_T^j for $j \in V^i$. RiskPropagation works in a similar way, but with the entries of the adjacency matrix being weighted by incoming transaction amounts.

PageRank/RiskPropagation as vector-RMS programs. We observe that the PageRank and RiskPropagation algorithms can both be captured as an RMS program with particularly nice structure: (i) It can be split up into T rounds, where each round only depends on output values of the previous rounds. (ii) The main computational operation to be performed by the parties is an inner product of a private vector with an encrypted vector. (iii) The computation is balanced: every round, each party has to perform M inner product operations on vectors of length M. We capture programs that satisfy all three properties as *balanced vector-RMS* programs.

Active security. Our starting point is the CDN framework [CDN01], which achieves active security by having parties prove in zero-knowledge (by means of commitments and Σ-protocols) that operations are performed correctly. Namely, for an RMS program, we require proofs of plaintext knowledge (for encryptions of inputs which are broadcast) and proofs of correct multiplication. However, naively applying these methods results in a relatively inefficient protocol – for instance, in each round of PageRank/RiskPropagation, parties have to send M proofs of correct multiplication for each node they update, resulting in an $O(T \cdot M^2)$ communication overhead in total.

Efficient proofs for AHE statements. We first observe that instead of parties committing to each entry of their input separately, they can use vector commitments. To make the message space of the commitment scheme compatible with the additively homomorphic encryptions, we show that the strategy to transform a commitment scheme into a vector commitment scheme presented in [ACC+22] can be adopted to transform the encryption scheme itself into a vector commitment scheme. Applying this to Paillier encryption, we construct a "Paillier-vector-commitment", which nicely composes with the Paillier encryption scheme used to instantiate the passive protocol. We further show that compressed Σ-protocol theory [AC20, ACC+22] can be deployed to prove linear statements over Paillier encryptions, resulting in proofs with logarithmic communication.

Amortizing many proofs. We propose two variations of our MPC protocol: the first, which we call low.comm, achieves very low communication costs; the second, low.comp, is more balanced and has lower computational complexity while still being quite communication-efficient (see comparison in Table 1). This separation arises from the use of two distinct methods to amortize the costs of Σ-protocols to prove multiple related statements.

In low.comm mode, each party P_i commits to its whole input (private adjacency matrix) in a single large vector commitment. Once the whole program has been evaluated (before the decryption step), P_i broadcasts a single proof that all operations were computed correctly, by amortizing over many proofs with the same secret.

On the other hand, low.comp mode exploits the structure of the balanced vector-RMS program by having P_i commit to each row of its adjacency matrix in a separate vector commitment. Then, in each round of the protocol (corresponding to one layer of the program), P_i provides a single proof that it correctly computed all the inner products

(multiplication between a row of the adjacency matrix and a vector of encrypted values). This is possible through an amortization technique which allows proving knowledge of multiple pre-images of the same homomorphism.

Reducing computation time via sparse blinding. The main computational cost of our proofs comes from vector exponentiations of the form $\boldsymbol{g}^{\boldsymbol{x}}$, which scale with the number of non-zero entries of $\boldsymbol{x}$. In standard Σ-protocols, the input vector is blinded by a random vector, which means the exponentiation is expensive even if the initial input is sparse. We show that blinding the entire input vector is in fact not required for compressed Σ-protocols. Intuitively, they reveal less information to the verifier (due to the reduced communication complexity), and thus require less blinding than a standard Σ-protocol. Our strategy is to introduce a single blinding coefficient in each round, which turns out to be sufficient to achieve zero-knowledge and preserve knowledge soundness. This sparse blinding technique greatly reduces the number of exponentiations the prover must compute when the input is sparse, and could be of independent interest for other zero-knowledge applications as well, such as proving knowledge of a sparse witness for more general arithmetic circuits using the techniques of [AC20]. In the context of our MPC protocol, although some of the amortization methods we employ do not necessarily preserve sparseness, this optimization represents a significant improvement in prover time for the proof relative to a large vector commitment in our low.comm protocol.

The remainder of this work is structured as follows. Section 2 introduces several relevant concepts and examples. In Section 3 we construct vector commitments from an AHE scheme. In Section 4 we describe the basic Σ-protocol and the amortization techniques which will be used in our proofs, and in Section 5 we show how it can be compressed. Using all the tools from previous sections, we present our MPC protocol (in its two variants) in Section 6. Finally, in Section 7 we discuss the implementation of our zero-knowledge proofs and some experimental results.

2 Preliminaries

2.1 Restricted Multiplication Straight-Line (RMS) Programs

Restricted Multiplication Straight-line (RMS) programs can be seen as a form of arithmetic circuits which distinguish *input values* from *memory values* and impose a restriction on multiplication – two memory values cannot be multiplied.

Definition 1 (RMS programs)**.** An RMS program over a ring $\mathcal{R}$ consists of a sequence of instructions of the four types below, in which we denote the representation of a value $x \in \mathcal{R}$ as an input or in memory by I_x or S_x, respectively.

- Load input into memory: $\mathsf{Load}(I_x) \to S_x$.
- Add values in memory: $\mathsf{Add}(S_x, S_y) \to S_{x+y}$.
- Multiply input by memory value: $\mathsf{Mult}(I_x, S_y) \to S_{xy}$.
- Output memory value: $\mathsf{Output}(S_x) \to x \in \mathcal{R}$.

We introduce the notion of a *Vector*-RMS program, in which multiplications by input values followed by addition of the resulting memory values are grouped together as an inner-product operation. We do so since our MPC protocols save the most in communication complexity with respect to other approaches when the evaluation of an RMS program can be written in terms of these inner product operations for relatively large vectors.

Definition 2 (Vector-RMS programs)**.** A vector-RMS program over a ring $\mathcal{R}$ consists of a sequence of instructions on global input values $x \in \mathcal{R}$ and vectors of local input values $\boldsymbol{y} \in \mathcal{R}^n$. The program is organized into layers $t = 1, \ldots, T$, such that operations on memory values only use memory values from previous layers, and the program is evaluated

layer by layer. We denote by S_x^t the representation of a value $x \in \mathcal{R}$ as a layer t memory value and $I_{\boldsymbol{y}}$ for the representation of a vector $\boldsymbol{y} \in \mathcal{R}^n$ as a local input value, and write $\boldsymbol{S}_{\boldsymbol{x}}^{\boldsymbol{u}} := (S_{x_1}^{u_1}, \ldots, S_{x_n}^{u_n})$ for a vector of memory values with $\boldsymbol{x} := (x_1, \ldots, x_n) \in \mathcal{R}^n$ and $\boldsymbol{u} := (u_1, \ldots, u_n) \in [T]^n$.

- Load inputs into memory: $\mathsf{Load}(I_x) \to S_x^1$.
- For each layer $t = 2, \ldots, T$:
  - Add values in memory: $\mathsf{Add}(S_x^u, S_y^v) \to S_{x+y}^t$ for $u, v \in [t-1]$.
  - Multiply input by memory value: $\mathsf{Mult}(I_x, S_y^u) \to S_{xy}^t$ for $u \in [t-1]$.
  - Compute inner product: $\mathsf{Prod}_n(\boldsymbol{S}_{\boldsymbol{x}}^{\boldsymbol{u}}, I_{\boldsymbol{y}}) \to S_{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}^t$ for $\boldsymbol{u} \in [t-1]^n$.
- Output memory value: $\mathsf{Output}(S_x^T) \to x \in \mathcal{R}$.

Example 1 (PageRank/RiskPropagation)**.** We recall the simplified describtion of the PageRank/RiskPropagation algorithms we presented in the introduction. The inputs to the algorithm consist of a set V consisting of M nodes together with an initial attribute value x_j^1 for each node $j \in V$, the $M \times M$ (weighted) adjacency matrix $A := (\boldsymbol{a}_j)_{j \in [M]}$ indicating the (weighted) edges between the nodes in V. The algorithms can be seen as a vector-RMS program as follows:

- The initial attribute values are loaded into memory: $\mathsf{Load}(I_{x_j^1}) \to S_{x_j^1}^1$.
- For each round $t = 2, \ldots, T$:
  - Compute the inner product between $\boldsymbol{a}_j$ and the previous round attribute values $\boldsymbol{x}^{t-1} := (x_1^{t-1}, \ldots, x_M^{t-1})$: $\mathsf{Prod}_M(\boldsymbol{S}_{\boldsymbol{x}^{t-1}}^{t-1}, I_{\boldsymbol{a}_j}) \to S_{x_j^t}^t$.
- Output the round T attribute values: $\mathsf{Output}(S_{x_j^T}^T) \to x_j^T$.

2.2 Additively Homomorphic Encryption

An *additively homomorphic* encryption (AHE) scheme is an encryption scheme which allows a party to compute encryptions of the sum of two messages or the product of a known scalar and a message by performing operations on the encryptions of these messages which only need the public key. A *threshold* additively homomorphic encryption scheme additionally splits up the secret key in shares, gives each party one of these shares, and allows the parties to successfully decrypt a ciphertext if and only if a certain threshold of t out of n parties agree to do so. We moreover require the parties to be able to compute re-randomized encryptions of messages using only the ciphertext and the public key.

Formally, a TAHE scheme consists of a tuple of PPT algorithms $\mathsf{AHE} := (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ together with a key distribution protocol Π_{KD}, a threshold decryption protocol Π_{DEC}, and is parametrized by a number of parties n, a threshold t, a message ring family $(\mathcal{M}_\lambda, +, \cdot)_\lambda$, randomness space family $(\mathcal{R}_\lambda)_\lambda$ and ciphertext group family $(\mathcal{C}_\lambda, *)_\lambda$ such that:

- $\mathsf{Gen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n)$. Takes as input the security parameter λ and outputs a public key pk and secret key shares $\mathsf{sk}_1, \ldots, \mathsf{sk}_n$.
- Π_{KD} is an n-party protocol, secure against $n-t$ corruptions, which takes as input the security parameter λ, computes $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n) \leftarrow \mathsf{Gen}(1^\lambda)$ and outputs $(\mathsf{pk}, \mathsf{sk}_i)$ to party P_i for each $i \in [n]$.
- $\mathsf{Enc}(\mathsf{pk}, x; \rho) \to c$. Takes as input a public key pk, a plaintext message $x \in \mathcal{M}_\lambda$ and randomness $\rho \in \mathcal{R}_\lambda$, and outputs a ciphertext $c \in \mathcal{C}_\lambda$.
- $\mathsf{Dec}((\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n), c) \to x$. Takes as input a public key pk, corresponding secret key shares $(\mathsf{sk}_1 \ldots, \mathsf{sk}_n)$ and a ciphertext $c \in \mathcal{C}_\lambda$ and outputs a plaintext message $x \in \mathcal{M}_\lambda$.

- Π_{DEC} is an n-party protocol, secure against $n-t$ corruptions, which takes as input a set of ciphertexts C, a public key pk and corresponding secret key shares sk_i from all honest parties, and outputs $\{\mathsf{Dec}((\mathsf{pk}, \mathsf{sk}_1, \dots, \mathsf{sk}_n), c) : c \in C\}$ to all parties.
- $R^{\mathsf{AHE}}_{\mathsf{pk}}(\cdot) \to \rho$. Takes various inputs, which will be detailed below, and outputs randomness $\rho \in \mathcal{R}_\lambda$.

We require these to satisfy the following properties for all $\lambda \in \mathbb{N}$ and all $(\mathsf{pk}, \mathsf{sk}_1, \dots, \mathsf{sk}_n) \leftarrow \mathsf{Gen}(1^\lambda)$, where we often omit the subscripts λ and pk, and the superscript AHE, as long as they are clear from the context:

- **Correctness:** For all $x \in \mathcal{M}$, all $\rho \in \mathcal{R}$ and $c \leftarrow \mathsf{Enc}(\mathsf{pk}, x; \rho)$, it holds that $\mathsf{Dec}((\mathsf{pk}, \mathsf{sk}_1, \dots, \mathsf{sk}_n), c) = x$.
- **Homomorphic properties:** For all $a \in \mathbb{Z}$, all $x_1, x_2 \in \mathcal{M}$, all $\rho_1, \rho_2 \in \mathcal{R}$ and $c_1 \leftarrow \mathsf{Enc}(\mathsf{pk}, x_1; \rho_1)$, $c_2 \leftarrow \mathsf{Enc}(\mathsf{pk}, x_2; \rho_2)$, it holds that

$$\begin{aligned} c_1 * c_2 &= \mathsf{Enc}(\mathsf{pk}, x_1 + x_2; R(x_1, x_2, \rho_1, \rho_2)), \\ c_1^a &= \mathsf{Enc}(\mathsf{pk}, a \cdot x_1; R(x_1, a, \rho_1)). \end{aligned}$$

- **Randomization Property:** For all $x_1, x_2 \in \mathcal{M}$ and at least one of $\rho_1, \rho_2 \in \mathcal{R}$ sampled uniformly, $R(x_1, x_2, \rho_1, \rho_2)$ is distributed uniformly in $\mathcal{R}$.
- **Threshold Semantic Security:** Let $\mathcal{A}$ be any PPT algorithm that receives as input 1^λ, the public key pk and the secret keys $\{\mathsf{sk}_i\}_{i \in C}$ of a subset $C \subset [n]$ of at most $n-t$ parties, and outputs two messages $x_0, x_1 \in \mathcal{M}_\lambda$ and some arbitrary value $s \in \{0,1\}^*$. Let

$$X_i(\lambda, C) := \left\{ (s, c_i \leftarrow \mathsf{Enc}(\mathsf{pk}, x_i; \rho)) \,\middle|\, \begin{smallmatrix} (\mathsf{pk}, \mathsf{sk}_1, \dots, \mathsf{sk}_n) \leftarrow \mathsf{Gen}(1^\lambda) \\ (x_0, x_1, s) \leftarrow \mathcal{A}(1^\lambda, C, \mathsf{pk}, \{\mathsf{sk}_i\}_{i \in C}) \\ \rho \xleftarrow{\$} \mathcal{R}_\lambda \end{smallmatrix} \right\}$$

  Then we require the distribution ensembles X_0 and X_1 to be computationally indistinguishable, where $X_i := \{X_i(\lambda, C)\}_{\lambda \in \mathbb{N},\, C \subset [n]}$ and C is any subset of size at most $n-t$.

Remark 1. Note that by the randomization property, multiplying any ciphertext $c := \mathsf{Enc}(\mathsf{pk}, x; \rho)$ with a fresh encryption of zero $c_0 := \mathsf{Enc}(\mathsf{pk}, 0; \rho_0)$, with $\rho_0 \xleftarrow{\$} \mathcal{R}$, results in a ciphertext $c * c_0 = \mathsf{Enc}(\mathsf{pk}, x; R(x, 0, \rho, \rho_0))$ which is distributed identically to an encryption of x with uniformly distributed randomness.

Example 2 (Paillier Encryption)**.** Paillier's well known public key cryptosystem [Pai99] has message space $\mathcal{M} := \mathbb{Z}_N$, randomness space $\mathcal{R} := \mathbb{Z}_N^*$ and ciphertext space $\mathcal{C} := \mathbb{Z}_{N^2}^*$, and consists of the following algorithms:

- Gen : On input 1^λ, samples (N, p, q) where $N = pq$ and p, q are λ-bit primes (with all but negligible probability in λ) and outputs $\mathsf{pk} := N$, $\mathsf{sk} := (N, \phi(N))$, where ϕ is the Euler phi function.
- Enc : On input a public key $\mathsf{pk} = N$, a message $x \in \mathbb{Z}_N$ and randomness $\rho \in \mathbb{Z}_N^*$, outputs $c := [(1+N)^x \cdot \rho^N \bmod N^2]$.
- Dec : On input a private key $\mathsf{sk} = (N, \phi(N))$ and a ciphertext $c \in \mathbb{Z}_{N^2}^*$, computes $y := \frac{[c^{\phi(N)} \bmod N^2] - 1}{N}$ and outputs $x := \left[y \cdot \phi(N)^{-1} \bmod N\right]$.

It is an IND-CPA secure public key encryption scheme under the decisional composite residuosity (DCR) assumption, which states that the distributions of (N, r) and $(N, [r^N \bmod N^2])$, over $(N, p, q) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$ and $r \xleftarrow{\$} \mathbb{Z}_{N^2}^*$, are computationally indistinguishable. The homomorphic and randomization properties follow directly from the fact that the map $\psi(a, b) := [(1+N)^a \cdot b^N \bmod N^2]$ is an isomorphism $\psi : \mathbb{Z}_N \times \mathbb{Z}_N^* \to \mathbb{Z}_{N^2}^*$.

Note that the randomization function is given by $R(x, x', \rho, \rho') := [\rho \cdot \rho' \bmod N]$. A key distribution protocol Π_{KD} and a threshold decryption protocol Π_{DEC}, together with a proof of threshold semantic security, were first given in two concurrent works [DJ00, FPS00], the details of which are out of scope for this work. Note that Damgård and Jurik [DJ01] also constructed a generalization of Paillier's cryptosystem over moduli N^{s+1} for $s \in \mathbb{N}$, with message space $\mathbb{Z}_{N^s}$, together with key distribution and threshold decryption protocols. The homomorphic and randomization properties follow similarly as for Paillier's scheme, and security again holds under the DCR assumption.

We discuss some other examples of AHE schemes, and to what extent they fit into our framework, in Appendix B.

2.3 Commitment Schemes

We give the definition of a homomorphic *vector commitment scheme*, in which one can commit to a vector $\boldsymbol{x} \in \mathcal{M}^n$ of fixed length $n \in \mathbb{N}$ in a single value $c \in \mathcal{C}$, and restrict ourselves to the case where $\mathcal{M} = \mathbb{Z}_m$ for some $m \in \mathbb{N}$. A homomorphic *single-value* commitment scheme will simply be the special case where $n = 1$.

Formally, a vector commitment scheme consists of a tuple of PPT algorithms $\mathsf{COM} := (\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$, and is parametrized by a message ring family $(\mathcal{M}_\lambda, +, \cdot)_\lambda$, a randomness space family $(\mathcal{R}_\lambda)_\lambda$, a commitment group family $(\mathcal{C}_\lambda, *)_\lambda$ and a vector length $n \in \mathbb{N}$.

- $\mathsf{Gen}(1^\lambda, n, m) \to \mathsf{pp}$. Takes as input the security parameter λ, the vector length n and the modulus m, and outputs public parameters pp.
- $\mathsf{Com}(\mathsf{pp}, \boldsymbol{x}; r) \to c$. Takes as input the public parameters pp, a vector $\boldsymbol{x} \in \mathcal{M}_\lambda^n$ and randomness $r \in \mathcal{R}_\lambda$, and outputs a commitment $c \in \mathcal{C}_\lambda$.
- $\mathsf{Open}(\mathsf{pp}, c, \boldsymbol{x}, r) \to b$. Takes as input the public parameters pp, a commitment $c \in \mathcal{C}_\lambda$, a vector $\boldsymbol{x} \in \mathcal{M}_\lambda^n$ and randomness $r \in \mathcal{R}_\lambda$, and outputs a bit $b \in \{0, 1\}$.
- $R_{\mathsf{pp}}^{\mathsf{COM}}(\cdot) \to r$. Takes various inputs, which will be detailed below, and outputs randomness $r \in \mathcal{R}_\lambda$.

We require these to satisfy the following properties for all $\lambda, n, m \in \mathbb{N}$ and all $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda, n, m)$, where we often omit the subscripts λ and pp, and the superscript COM, as long as they are clear from the context:

- **Correctness:** For all $\boldsymbol{x} \in \mathcal{M}^n$, $r \in \mathcal{R}$ and $c \leftarrow \mathsf{Com}(\mathsf{pp}, \boldsymbol{x}; r)$, it holds that $\mathsf{Open}(\mathsf{pp}, c, \boldsymbol{x}, r) = 1$.
- **Hiding:** Let $\mathcal{A}$ be a PPT (resp. any) algorithm that receives 1^λ and the public parameters pp, and outputs two vectors $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{M}^n$ and some arbitrary value $s \in \{0, 1\}^*$. Let

$$X_i(\lambda) := \left\{ (s, c_i \leftarrow \mathsf{Com}(\mathsf{pp}, \boldsymbol{x}_i; r)) \,\middle|\, \begin{smallmatrix} \mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda, n, m) \\ (\boldsymbol{x}_1, \boldsymbol{x}_2, s) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp}) \\ r \xleftarrow{\$} \mathcal{R} \end{smallmatrix} \right\}$$

  Then we say that the commitment scheme is computationally (resp. statistically) hiding if the distribution ensembles $X_i := (X_i(\lambda))_{\lambda \in \mathbb{N}}$ are computationally (resp. statistically) indistinguishable for $i \in \{1, 2\}$. We say that the commitment scheme is *perfectly* hiding if it is statistically hiding and the distribution ensembles are identical.
- **Binding:** Let $\mathcal{A}$ be a PPT (resp. any) algorithm that receives as input 1^λ and the public parameters pp, and outputs two vectors $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{M}^n$ and corresponding randomness $r_1, r_2 \in \mathcal{R}$. Then we say that the commitment scheme is computationally (resp. statistically) binding if the following probability is bounded by some negligible function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}_{>0}$ for all $\lambda \in \mathbb{N}$:

$$\Pr\left[\mathsf{Com}(\mathsf{pp}, \boldsymbol{x}_1; r_1) = \mathsf{Com}(\mathsf{pp}, \boldsymbol{x}_2; r_2) \,\middle|\, \begin{smallmatrix} \mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda, n, m) \\ (\boldsymbol{x}_1, r_1, \boldsymbol{x}_2, r_2) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp}) \\ \boldsymbol{x}_1 \neq \boldsymbol{x}_2 \end{smallmatrix} \right]$$

We say that the commitment scheme is *perfectly* binding if it is statistically binding and the probability is equal to 0.

- **Homomorphic properties:** For all $a \in \mathbb{Z}$, all $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{M}^n$, $r_1, r_2 \in \mathcal{R}$ and $c_1 \leftarrow \mathsf{Com}(\mathsf{pp}, \boldsymbol{x}_1; r_1)$, $c_2 \leftarrow \mathsf{Com}(\mathsf{pp}, \boldsymbol{x}_2; r_2)$, it holds that

$$c_1 * c_2 = \mathsf{Com}(\mathsf{pp}, \boldsymbol{x}_1 + \boldsymbol{x}_2; R(\boldsymbol{x}_1, \boldsymbol{x}_2, r_2, r_2)),$$
$$c_1^a = \mathsf{Com}(\mathsf{pp}, a \cdot \boldsymbol{x}_1; R(\boldsymbol{x}_1, a, r_1)).$$

- **Randomization Property:** For all $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{M}^n$ and at least one of $r_1, r_2 \in \mathcal{R}$ sampled uniformly, $R(x_1, x_2, r_1, r_2)$ is distributed uniformly in $\mathcal{R}$.

2.3.1 Commitments from public-key encryption.

It is well known that public-key encryption schemes can also be used as commitment schemes. Suppose we have an IND-CPA secure public-key encryption scheme $\mathsf{PKE} := (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\mathcal{M}$, randomness space $\mathcal{R}$ and ciphertext space $\mathcal{C}$. Intuitively, we can use this as a single-value commitment scheme with message space $\mathcal{M}$, randomness space $\mathcal{R}$ and commitment space $\mathcal{C}$ by running $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$, setting $\mathsf{pp} := \mathsf{pk}$ and defining $c := \mathsf{Com}(\mathsf{pp}, x; r) := \mathsf{Enc}(\mathsf{pk}, x; r)$. In order to open the commitment c, the original message x and the randomness r used to encrypt can be revealed and it can be checked whether $c = \mathsf{Enc}(\mathsf{pk}, x; r)$. As we will see shortly, PKE needs to satisfy the additional condition that one is able to efficiently check the validity of a public key pk.

In order to show that this indeed yields a valid single-value commitment scheme, we prove that it is computationally hiding and perfectly binding:

- **Computationally Hiding:** Follows from IND-CPA security of PKE as the hiding experiment for a single-value commitment scheme is equal to the IND-CPA experiment.
- **Perfectly Binding:** Follows from the correctness of PKE as long as the decryption function is uniquely determined for given public parameters $\mathsf{pp} := \mathsf{pk}$. This is immediate for any validly generated keypair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$, and thus the commitment scheme is perfectly binding if it is possible to efficiently check the validity of a public key pk.

2.3.2 Vector Commitments from single-value commitments.

A recent result by Attema et al. [ACC+22] shows that it is possible to turn a homomorphic single-value commitment scheme over a ring $\mathbb{Z}_m$ into a homomorphic vector commitment scheme over $\mathbb{Z}_m^n$. Next to the conditions specified in Section 2.3 with $n = 1$, the single-value commitment scheme needs to satisfy an additional *zero-opening* property, which is defined as follows.

Definition 3 ([ACC+22, Def. 2])**.** Let $\mathsf{COM}' := (\mathsf{Gen}', \mathsf{Com}', \mathsf{Open}')$ be a homomorphic single-value commitment scheme with message ring $\mathbb{Z}_m$. Then we say that COM' satisfies the *zero-opening condition* if there exists an efficiently computable function R such that for any commitment $c := \mathsf{Com}'(\mathsf{pp}, x; r)$ we have

$$c^m = \mathsf{Com}'(\mathsf{pp}, 0; R(c)).$$

The construction of a vector commitment scheme COM for $\mathbb{Z}_m^n$ from a single-value commitment scheme COM' over $\mathbb{Z}_m$ is detailed in Figure 1.

Remark 2. In [ACC+22, Thm. 1] it is proven that if COM' is a *perfectly* hiding and *computationally* binding single-value commitment scheme, then the construction in Figure 1 gives a vector commitment scheme COM which is perfectly hiding and computationally

Figure 1 Vector Commitment Scheme for $\mathbb{Z}_m^n$

- $\mathsf{Gen}(1^\lambda, n, m)$ for $\lambda, n, m \in \mathbb{N}$:
  - Generate $\mathsf{pp}' \leftarrow \mathsf{Gen}'(1^\lambda, 1, m)$.
  - For $i = 1, \ldots, n$, sample $a_i \xleftarrow{\$} \mathbb{Z}_m$ and $r_i \xleftarrow{\$} \mathcal{R}$. Set $g_i := \mathsf{Com}'(\mathsf{pp}', a_i; r_i)$.
  - Output $\mathsf{pp} = (\mathsf{pp}', g_1, \ldots, g_n)$.
- $\mathsf{Com}(\mathsf{pp}, \boldsymbol{x}; r)$ for $\boldsymbol{x} := (x_1, \ldots, x_n) \in \mathbb{Z}_m^n$ and $r \in \mathcal{R}$:
  - Output $\mathsf{Com}'(\mathsf{pp}', 0; r) * g_1^{x_1} * \cdots * g_n^{x_n}$.
- $\mathsf{Open}(\mathsf{pp}, c, \boldsymbol{x}, r)$ for $c \in \mathcal{C}$, $\boldsymbol{x} \in \mathbb{Z}_m^n$ and $r \in \mathcal{R}$:
  - If $c = \mathsf{Com}(\mathsf{pp}, \boldsymbol{x}; r)$ output 1, otherwise output 0.

binding as well. However, in this work we will be constructing a vector commitment scheme by using single-value commitments from an additively homomorphic public-key encryption scheme, which are *computationally* hiding and *perfectly* binding. Therefore, we prove in Section 3 that using such commitments results in a vector commitment scheme which is *computationally* hiding as well as *computationally* binding.

2.4 Interactive Proofs

An interactive proof for an NP-relation $\mathcal{R}$ is a protocol which allows a prover to convince a verifier that it knows a witness w for a given statement x, such that $(x, w) \in \mathcal{R}$. We will consider Σ-protocols with the standard properties of completeness, k-special soundness and special honest verifier zero-knowledge.

- **Completeness:** If $(x, w) \in \mathcal{R}$, the verifier accepts with probability 1.
- **k-special soundness:** In a 3-move protocol, given a statement x and k accepting transcripts with distinct verifier messages, it is possible to efficiently extract a witness w such that $(x, w) \in \mathcal{R}$.
- **Special honest verifier zero-knowledge (SHVZK):** There exists a simulator which, on input a statement x and uniformly random verifier messages, can efficiently generate a transcript identically distributed to an interaction between the prover with input $(x, w) \in \mathcal{R}$ and the verifier.

We have defined special soundness only for 3-move protocols, but we also consider the generalized notion of $(k_1, \ldots, k_\ell)$-special soundness for $(2\ell + 1)$-move protocols, as defined in [ACC+22], which implies knowledge soundness.

2.4.1 Composition of protocols.

Let Π_1 be a $(2\ell_1 + 1)$-move interactive proof for relation $\mathcal{R}_1$ and Π_2 a $(2\ell_2 + 1)$-move interactive proof for relation $\mathcal{R}_2$. Let also f be an efficiently computable function such that a transcript $(a_1, e_1, \ldots, a_\ell, e_\ell, a_{\ell+1})$ of Π_1 on public input x_1 is accepting if and only if $(x_2, a_{\ell+1}) \in \mathcal{R}_2$, where $a_{\ell+1}$ is the last message of the transcript and $x_2 = f(x_1, a_1, e_1, \ldots, a_\ell, e_\ell)$. The composition of Π_1 with Π_2, denoted by $\Pi_2 \diamond \Pi_1$ (with f clear from the context), is defined as follows. On input a statement and witness pair $(x_1; w_1)$, it consists of an execution of $\Pi_1(x_1; w_1)$ in which the final message $a_{\ell+1}$ sent by the prover is replaced by an execution of $\Pi_2(x_2; a_{\ell+1})$. More specifically, after exchanging the partial transcript $(a_1, e_1, \ldots, a_\ell, e_\ell)$, the prover and verifier both compute

$x_2 = f(x_1, a_1, e_1, \ldots, a_\ell, e_\ell)$ and the prover additionally computes $a_{\ell+1}$. They then run $\Pi_2(x_2; a_{\ell+1})$, and the verifier accepts the overall proof if verification of Π_2 succeeds.

The composed protocol $\Pi_2 \diamond \Pi_1$ enjoys the following properties (see [Att23]):

- $\Pi_2 \diamond \Pi_1$ is a $(2\ell_1 + 2\ell_2 + 1)$-round interactive proof for $\mathcal{R}_1$.
- If Π_1 and Π_2 are complete, then so is $\Pi_2 \diamond \Pi_1$.
- If Π_1 is $(k_1, \ldots, k_{\ell_1})$-special sound and Π_2 is $(k'_1, \ldots, k'_{\ell_2})$-special sound, then $\Pi_2 \diamond \Pi_1$ is $(k_1, \ldots, k_{\ell_1}, k'_1, \ldots, k'_{\ell_2})$-special sound.
- If Π_1 is special honest verifier zero-knowledge, then so is $\Pi_2 \diamond \Pi_1$.

2.4.2 Exceptional challenge sets.

In the interactive protocols we present, the verifier's challenges are elements of a ring $\mathbb{Z}_m$, where m is not necessarily a prime. For soundness to hold, we need the verifier to sample challenges from an *exceptional* subset $\mathcal{E} \subseteq \mathbb{Z}_m$. We say that $\mathcal{E}$ is exceptional if $e - e'$ is an invertible element of $\mathbb{Z}_m$ for all distinct $e, e' \in \mathcal{E}$. Note that the cardinality of the largest exceptional subset of $\mathbb{Z}_m$ is equal to the smallest prime p dividing m. A smaller challenge set leads to a larger knowledge error, which can mostly be overcome by parallel repetition of the protocol. However, for the compressed protocols in Section 5, the challenge set needs to have at least cardinality 3. For this reason we assume in this work that $2 \nmid m$. For even moduli m, it is possible to work over an extension of the ring $\mathbb{Z}_m$ to allow for larger challenge set (see [ACC+22]).

In some cases it is not possible to efficiently determine an exceptional subset of $\mathbb{Z}_m$, such as when $m = N = pq$ is the product of two large primes p and q as is the case for the message space of the Paillier encryption scheme. In this case the probability that $\gcd(e - e', N) > 1$ for uniformly random $e, e' \xleftarrow{\$} \mathbb{Z}_N$ is negligible, so it suffices to sample challenges from the whole ring $\mathbb{Z}_N$. This does however have the consequence that the resulting protocol has computational soundness under the assumption that factoring N is intractable. Alternatively one can adapt the relation underlying the protocol such that a witness for the new relation is either a witness for the original relation or a factor of N.

3 Vector Commitments from AHE

In this section we construct a homomorphic vector commitment scheme from a homomorphic single-value commitment scheme using the construction from [ACC+22] and single-value commitments coming from an additively homomorphic encryption scheme. Since a single-value commitment scheme obtained in this way is *computationally* hiding and *perfectly* binding, we present an adapted version of [ACC+22, Thm. 1]. Note that AHE additionally needs to satisfy the zero-opening condition from Definition 3.

Example 3. The Paillier encryption scheme as defined in Example 2 satisfies the zero-opening condition from Definition 3. That is, for all $x \in \mathbb{Z}_N, \rho \in \mathbb{Z}_N^*$ and $C := \mathsf{Enc}(\mathsf{pk}, x; \rho) := (1+N)^x \cdot \rho^N \bmod N^2$, we have

$$\mathsf{Enc}(\mathsf{pk}, 0; [C \bmod N]) = C^N \bmod N^2.$$

Theorem 1. *When based on a homomorphic single-value commitment scheme* COM' *for* $\mathbb{Z}_m$ *satisfying the conditions in Section 2.3 with computational hiding, perfect binding and satisfying the zero-opening condition from Definition 3, the construction* COM *from Figure 1 is a computationally hiding and computationally binding homomorphic vector commitment scheme for* $\mathbb{Z}_m^n$.

Proof. The proof proceeds similarly to [ACC+22, Thm. 1], so we focus on the areas that are different. Consider an adversary $\mathcal{A}$ against the hiding property of COM, then we can use it to construct an adversary $\mathcal{A}'$ against the hiding property of COM', on input 1^λ, $\mathsf{pp}' \leftarrow \mathsf{Gen}'(1^\lambda, m)$, as follows:

- For $i = 1, \ldots, n$, sample $a_i \xleftarrow{\$} \mathbb{Z}_m$ and $r_i \xleftarrow{\$} \mathcal{R}$, and put $g_i := \mathsf{Com}'(\mathsf{pp}', a_i, r_i)$. Put $\mathsf{pp} := (\mathsf{pp}', g_1, \ldots, g_n)$.
- Query $(\boldsymbol{x}^0, \boldsymbol{x}^1, s) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp})$ and put $x^b := \sum_{i=1}^n a_i x_i^b$ for $b \in \{0, 1\}$.
- Output (x^0, x^1, s) to the challenger and receive (s, c_b) (where $c_b \leftarrow \mathsf{Com}'(\mathsf{pp}', x^b; r')$ for $b \xleftarrow{\$} \{0, 1\}$, $r' \xleftarrow{\$} \mathcal{R}$).
- Query $b' \leftarrow \mathcal{A}(s, c_b)$ and output b'.

Then we claim that $\mathcal{A}'$ distinguishes between the distributions of (s, c_0) and (s, c_1) with at least the probability that $\mathcal{A}$ wins the hiding experiment. This holds because: (1) $\mathcal{A}'$ generates the public parameters pp identical to $\mathsf{Gen}(1^\lambda, n, m)$; (2) The element c_b is identically distributed to $\mathsf{Com}(\mathsf{pp}, \boldsymbol{x}^b; r)$ with $r \xleftarrow{\$} \mathcal{R}$ since

$$\begin{aligned}\mathsf{Com}'(\mathsf{pp}', x^b; r') &= \prod_{i=1}^n \mathsf{Com}'(\mathsf{pp}', a_i; r_i)^{x_i^b} * \mathsf{Com}'(\mathsf{pp}, 0; r) \\ &= \mathsf{Com}(\mathsf{pp}, \boldsymbol{x}^b; r), \text{ for } r := R(x^b, -x^b, r', \tilde{r}),\end{aligned}$$

where $\tilde{r}$ is the randomness of $\prod_{i=1}^n \mathsf{Com}'(\mathsf{pp}', a_i; r_i)^{-x_i^b}$ which can be defined through the homomorphic properties of Com', and r is distributed uniformly random in $\mathcal{R}$ by the randomization property since r' is distributed uniformly random in $\mathcal{R}$. Since COM' is computationally hiding, $\mathcal{A}'$ succeeds with at most negligible probability and COM is computationally hiding.

Now consider an adversary $\mathcal{A}$ against the binding property of COM, then we can use it to construct an adversary $\mathcal{A}'$ against the hiding property of COM' , on input 1^λ, $\mathsf{pp}' \leftarrow \mathsf{Gen}'(1^\lambda, m)$, as follows:

- Sample $i^* \xleftarrow{\$} [n]$ and $a_{i^*}^0, a_{i^*}^1 \xleftarrow{\$} \mathbb{Z}_m$, output $(a_{i^*}^0, a_{i^*}^1)$ to the challenger and receive g_{i^*} (where $g_{i^*} \leftarrow \mathsf{Com}'(\mathsf{pp}', a_{i^*}^b; r_{i^*})$ for $b \xleftarrow{\$} \{0, 1\}$, $r_{i^*} \xleftarrow{\$} \mathcal{R}$).
- For $i \neq i^*$, sample $a_i \xleftarrow{\$} \mathbb{Z}_m$, $r_i \xleftarrow{\$} \mathcal{R}$ and put $g_i := \mathsf{Com}'(\mathsf{pp}', a_i; r_i)$. Put $\mathsf{pp} := (\mathsf{pp}', g_1, \ldots, g_n)$.
- Query $(\boldsymbol{x}, r, \boldsymbol{x}', r') \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp})$.
- If $\mathsf{Com}(\mathsf{pp}, \boldsymbol{x}; r) = \mathsf{Com}(\mathsf{pp}, \boldsymbol{x}'; r')$ and $x_{i^*} \not\equiv x'_{i^*} \mod m$:
  - Let $d := \gcd(x_{i^*} - x'_{i^*}, m)$, let e be the multiplicity with which d divides m as an integer and let $M := m/d^e$. Note that $M \geq 2$.
  - If $a_{i^*}^0 \not\equiv a_{i^*}^1 \mod M$, output the bit $b := (a_{i^*}^1 \stackrel{?}{\equiv} -(x_{i^*} - x'_{i^*})^{-1} \cdot \sum_{i \neq i^*} a_i (x_i - x'_i) \mod M)$.
  - Else, sample $b \xleftarrow{\$} \{0, 1\}$ and output b.
- Else, sample $b \xleftarrow{\$} \{0, 1\}$ and output b.

The idea behind the reduction is the following. If the adversary $\mathcal{A}$ successfully outputs $(\boldsymbol{x}, r, \boldsymbol{x}', r')$ with $\mathsf{Com}(\mathsf{pp}, \boldsymbol{x}; r) = \mathsf{Com}(\mathsf{pp}, \boldsymbol{x}'; r')$, $\boldsymbol{x} \neq \boldsymbol{x}'$, then perfect binding of Com' implies that $\sum a_i x_i \equiv \sum a_i x'_i \mod m$. With probability $\geq 1/n$ we have $x_{i^*} \not\equiv x'_{i^*} \mod m$ since if $\mathcal{A}$ succeeds, $\boldsymbol{x}$ and $\boldsymbol{x}'$ differ in at least one coordinate. If this is the case we can solve for the i^*-th coordinate and compute $a_{i^*} := -(x_{i^*} - x'_{i^*})^{-1} \cdot \sum_{i \neq i^*} a_i (x_i - x'_i) \mod M$. If $a_{i^*}^0 \not\equiv a_{i^*}^1 \mod M$ this means that $\mathcal{A}'$ can distinguish between $\mathsf{Com}'(\mathsf{pp}', a_{i^*}^0; r_{i^*})$ and $\mathsf{Com}'(\mathsf{pp}', a_{i^*}^1; r_{i^*})$. Since $M \geq 2$ we have $a_{i^*}^0 \not\equiv a_{i^*}^1 \mod M$ with probability $\geq 1/2$. Working out the probabilities we see that $\mathcal{A}'$ succeeds with probability at least $\varepsilon/(4n) + 1/2$, where ε is the success probability of $\mathcal{A}$. Since COM' is computationally hiding and n is

polynomial, it follows that $\mathcal{A}$ succeeds with at most negligible probability and COM is computationally binding.

The homomorphic properties of COM follow from the homomorphic and zero-opening properties of COM' completely analogous to [ACC+22, Thm. 1]. The same goes for the randomization property. □

Remark 3 (Setup). We would like to run the setup algorithm $\mathsf{Gen}(1^\lambda, M, m)$ of COM without the need for a trusted setup. In our construction from a threshold AHE scheme as the underlying single-value commitment scheme, the public parameters $\mathsf{pp} := \mathsf{pk}$ are already generated as part of the key distribution protocol Π_{KD}, and knowledge of $\leq n - t$ secret key-shares sk_j does not break the hiding property by the threshold security of AHE. Furthermore, the basis elements $g_i := \mathsf{Enc}(\mathsf{pk}, a_i; r_i)$ need to be generated with $a_i \xleftarrow{\$} \mathbb{Z}_m$, $r_i \xleftarrow{\$} \mathcal{R}$ not known to any of the parties, since otherwise this would break the binding property of COM. This can be realized by letting each party P_j sample $a_{i,j} \xleftarrow{\$} \mathbb{Z}_m$, $r_{i,j} \xleftarrow{\$} \mathcal{R}$ and publish $g_{i,j} := \mathsf{Enc}(\mathsf{pk}, a_{i,j}, r_{i,j})$ together with a proof of plaintext knowledge (as in Example 4). The parties then verify each other's proofs and compute the basis elements as $g_i := g_{i,1} * \cdots * g_{i,n}$. As long as at least one of the parties is honest, and therefore sampled $a_{i,j}$, $r_{i,j}$ uniformly random, the resulting g_i have the desired distribution.

In case the base elements are simply random elements of the commitment space, we can do better by using a public PRF F to sample the base elements as $F(k, 1), \ldots, F(k, M)$, where the seed k can be generated using, for example, a secure coin-tossing protocol. Note that this is the case when we use Paillier encryptions (Example 2) as the underlying single-value commitment scheme, and that sampling uniformly random elements in $\mathbb{Z}_{N^2}$ has negligible statistical distance to sampling uniformly random elements in $\mathbb{Z}^*_{N^2}$.

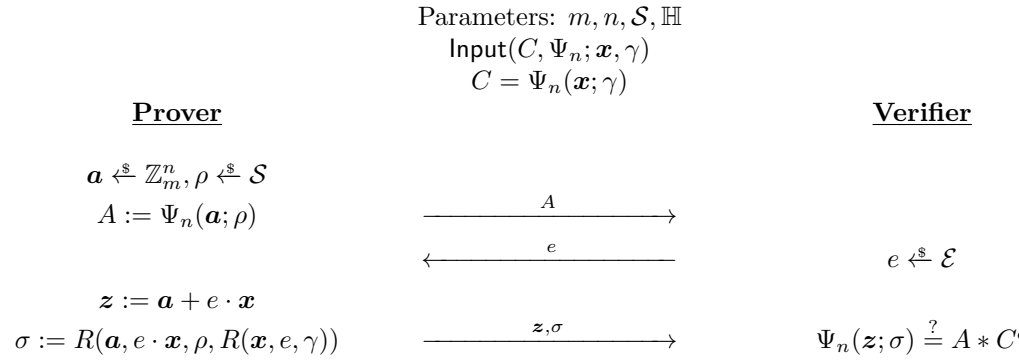
4 Standard Σ-Protocols

In this section we will describe a standard Σ-protocol for proving knowledge of a pre-image of a "homomorphism" Ψ_n which takes as inputs a vector $\boldsymbol{x}$ of length $n \in \mathbb{N}$ over a ring $\mathbb{Z}_m$ and some "randomness" γ from a finite set $\mathcal{S}$, and outputs an element $\Psi_n(\boldsymbol{x}; \gamma)$ of a group $(\mathbb{H}, *)$. The map $\Psi_n : \mathbb{Z}_m^n \times \mathcal{S} \to \mathbb{H}$ is not a homomorphism in the mathematical sense since we do not assume a group structure on the set $\mathcal{S}$, but we rather require it to satisfy the following homomorphic properties. There exists an efficiently computable function R which takes various inputs, as detailed below, and has outputs in $\mathcal{S}$, such that for all $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{Z}_m^n$, all $\gamma, \gamma' \in \mathcal{S}$, all $a \in \mathbb{Z}$ and all $C \in \mathbb{H}$:

- **Multiplication:** $\Psi_n(\boldsymbol{x}; \gamma) * \Psi_n(\boldsymbol{x}'; \gamma') = \Psi_n(\boldsymbol{x} + \boldsymbol{x}'; R(\boldsymbol{x}, \boldsymbol{x}', \gamma, \gamma'))$;
- **Scalar Multiplication:** $\Psi_n(\boldsymbol{x}; \gamma)^a = \Psi_n(a \cdot \boldsymbol{x}; R(\boldsymbol{x}, a, \gamma))$;
- **Randomization:** If at least one of γ, γ' is sampled uniformly at random from $\mathcal{S}$, then $R(\boldsymbol{x}, \boldsymbol{x}', \gamma, \gamma')$ is distributed uniformly in $\mathcal{S}$.
- **Zero-opening:** $C^m = \Psi_n(\boldsymbol{0}; R(C))$.

We show that when these conditions are satisfied, there exists a standard zero-knowledge Σ-protocol for proving knowledge of a pre-image $(\boldsymbol{x}, \gamma) \in \mathbb{Z}_m^n \times \mathcal{S}$ such that $\Psi_n(\boldsymbol{x}; \gamma) = C$ for some public element $C \in \mathbb{H}$. In Section 4.1 we give examples of morphisms Ψ_n to show that this immediately gives a proof of plaintext knowledge for an additively homomorphic encryption scheme and a proof of correct multiplication of a vector of ciphertexts with a vector of committed exponents. These two proofs form the main building blocks in our actively secure MPC protocol in Section 6. Additionally it gives a protocol for proving knowledge of a commitment opening satisfying a constraint, which is the main building block for constructing zero-knowledge arguments for arithmetic circuits in [AC20] (see Example 5 and Appendix A). We furthermore show in Section 4.2 that is possible to amortize the proofs over multiple witnesses for the same morphism and in Section 4.3 that

Figure 2 Standard Σ-protocol Π_Σ for relation $\mathscr{R}_\Psi$



it is possible to amortize over the same witness for multiple morphisms for some specific choices of morphisms. The standard protocol is given in Figure 2, where $\mathcal{E} \subset \mathbb{Z}_m$ denotes an exceptional subset (see Section 2.4.2).

Theorem 2. *Protocol Π_Σ from Figure 2 is a complete, special honest verifier zero-knowledge, 2-special sound protocol for the relation*

$$\mathscr{R}_\Psi := \{(C, \Psi_n; \boldsymbol{x}, \gamma) : C = \Psi_n(\boldsymbol{x}; \gamma)\}.$$

Proof. **Completeness:** Follows directly from the homomorphic properties of Ψ_n.

SHVZK: We can construct a simulator on input public parameters $m, n, \mathcal{S}, \mathbb{H}$ and a statement C, Ψ_n as follows. Sample $e \xleftarrow{\$} \mathcal{E}$, $\sigma \xleftarrow{\$} \mathcal{S}$ and $\boldsymbol{z} \xleftarrow{\$} \mathbb{Z}_m^n$, put $A := \Psi_n(\boldsymbol{z}; \sigma) * C^{-e}$, and output the transcript $(A, e, \boldsymbol{z}, \sigma)$. It is clear that this is an accepting transcript, and that it is distributed identically to a real transcript by the randomization property of $R(\cdot)$.

Special soundness: Let $(A, e, \boldsymbol{z}, \gamma)$, $(A, e', \boldsymbol{z}', \gamma')$ be two accepting transcripts with $e \neq e'$ for the same statement C, Ψ_n and with respect to the same public parameters $m, n, \mathcal{S}, \mathbb{H}$. From the accepting condition we have

$$C^{e-e'} = \Psi_n(\boldsymbol{z} - \boldsymbol{z}'; \hat{\sigma}), \quad \hat{\sigma} := R(\boldsymbol{z}, -\boldsymbol{z}', \sigma, R(\boldsymbol{z}', -1, \sigma'))$$

Since e, e' are distinct and sampled from an exceptional subset of $\mathbb{Z}_m$, there exists a $\beta \in \mathbb{Z}$ such that $\beta \cdot (e - e') = 1 + km$ for some $k \in \mathbb{Z}$. By the zero opening condition and the homomorphic properties of Ψ_n we can write

$$\begin{aligned} C &= \Psi_n(\boldsymbol{z} - \boldsymbol{z}'; \hat{\sigma})^\beta * \Psi_n(0; R(C))^{-k} \\ &= \Psi_n(\beta \cdot (\boldsymbol{z} - \boldsymbol{z}'); R(\beta(\boldsymbol{z} - \boldsymbol{z}'), 0, R(\boldsymbol{z} - \boldsymbol{z}', \beta, \hat{\sigma}), R(0, -k, R(C)))) \end{aligned}$$

So we conclude that we can efficiently extract a witness for the statement C, Ψ_n. □

4.1 Examples

Example 4 (Proof of Plaintext Knowledge)**.** The standard protocol from Figure 2 immediately gives us a proof of plaintext knowledge for an additively homomorphic encryption scheme AHE as in Subsection 2.2 with message space $\mathcal{M} := \mathbb{Z}_m$ and which satisfies the zero opening condition from Definition 3, by letting $\Psi_1(x; \gamma) := \mathsf{Enc}(\mathsf{pk}, x; \gamma)$.

Example 5 (Proof of Commitment Opening)**.** Let COM be a homomorphic vector commitment scheme over $\mathbb{Z}_m^n$ constructed from a homomorphic single value commitment scheme COM' over $\mathbb{Z}_m$ satisfying the zero-opening condition of Definition 3. Then COM satisfies an analogous zero-opening condition. That is, let $\mathsf{pp} = (\mathsf{pp}', g_1, \ldots, g_n)$ and

$C := \mathsf{Com}(\mathsf{pp}, \boldsymbol{x}; \gamma) = \mathsf{Com}'(\mathsf{pp}', 0; \gamma) * g_1^{x_1} * \cdots * g_n^{x_n}$. Then by construction C is a valid commitment under Com' and hence we have

$$C^m = \mathsf{Com}'(\mathsf{pp}', 0; R'(C)) = \mathsf{Com}(\mathsf{pp}, \mathbf{0}; R(C)), \quad R(C) := R'(C).$$

Hence by letting $\Psi_n(\boldsymbol{x}; \gamma) := \mathsf{Com}(\mathsf{pp}, \boldsymbol{x}; \gamma)$, the protocol from Figure 2 gives us a protocol for proving knowledge of an opening of a commitment. We show in Appendix A how to extend this protocol to additionally prove that $\boldsymbol{x}$ satisfies some linear form constraints.

Example 6 (Proof of Correct Multiplication)**.** Let $\boldsymbol{C} := (C_1, \ldots, C_n)$ be public, valid ciphertexts of an additively homomorphic scheme AHE with message space $\mathbb{Z}_m$ and ciphertext space $\mathcal{C}'$, and again let COM be a homomorphic commitment scheme over $\mathbb{Z}_m^n$ with commitment space $\mathcal{C}$ constructed from a homomorphic single value commitment scheme COM' over $\mathbb{Z}_m$. Then by plugging in the map

$$\Psi_n(\boldsymbol{x}; \gamma, \rho) := (\mathsf{Com}(\mathsf{pp}, \boldsymbol{x}; \gamma), \boldsymbol{C}^{\boldsymbol{x}} * \mathsf{Enc}(\mathsf{pk}, 0, \rho))$$

into the protocol from Figure 2, this gives us a protocol for proving knowledge of an opening $(\boldsymbol{x}, \gamma, \rho)$ additionally satisfying some relation $\boldsymbol{C}^{\boldsymbol{x}} * \mathsf{Enc}(\mathsf{pk}, 0, \rho) = B \in \mathcal{C}'$. It remains to show that Ψ_n satisfies the homomorphic, randomization and zero-opening condition.

Let R be the randomization function of COM with randomness space $\mathcal{R}$ and let R' be the randomization function of AHE with randomness space $\mathcal{R}'$. Then for $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{Z}_m^n$, $\gamma, \gamma' \in \mathcal{R}$ and $\rho, \rho' \in \mathcal{R}'$, let $\boldsymbol{z} := [\boldsymbol{x} + \boldsymbol{x}' \bmod m]$. We have

$$\Psi_n(\boldsymbol{x}; \gamma, \rho) * \Psi_n(\boldsymbol{x}'; \gamma', \rho') = (\mathsf{Com}(\mathsf{pp}, \boldsymbol{z}; \hat{\gamma}), \boldsymbol{C}^{\boldsymbol{x}+\boldsymbol{x}'} * \mathsf{Enc}(\mathsf{pk}, 0; \hat{\rho})) = (\star)$$

with $\hat{\gamma} := R(\boldsymbol{x}, \boldsymbol{x}', \gamma, \gamma')$ and $\hat{\rho} := R'(0, 0, \rho, \rho')$. We can pick $\boldsymbol{k} \in \mathbb{Z}^n$ such that $\boldsymbol{x} + \boldsymbol{x}' = \boldsymbol{z} + m \cdot \boldsymbol{k}$ and put $\tilde{\rho} := R'(0, 0, \hat{\rho}, R(\boldsymbol{C}^{\boldsymbol{k}}))$. Then

$$(\star) = (\mathsf{Com}(\mathsf{pp}, \boldsymbol{z}; \hat{\gamma}), \boldsymbol{C}^{\boldsymbol{z}} * \mathsf{Enc}(\mathsf{pk}, 0; \tilde{\rho})) = \Psi_n(\boldsymbol{z}; \hat{\gamma}, \tilde{\rho}).$$

Moreover, the randomization property follows directly from the randomization properties of R and R'. Similarly, for $a \in \mathbb{Z}$, let $\boldsymbol{z} := [a \cdot \boldsymbol{x} \bmod m]$, $\hat{\gamma} := R(\boldsymbol{x}, a, \gamma)$ and $\hat{\rho} := R(0, a, \rho)$. Then

$$\Psi_n(\boldsymbol{x}; \gamma, \rho)^a = (\mathsf{Com}(\mathsf{pp}, \boldsymbol{z}; \hat{\gamma}), C^{a \cdot \boldsymbol{x}} * \mathsf{Enc}(\mathsf{pk}, 0; \hat{\rho})) = \Psi_n(\boldsymbol{z}; \hat{\gamma}, \tilde{\rho}),$$

where $\tilde{\rho} = R(0, 0, \hat{\rho}, R(\boldsymbol{C}^{\boldsymbol{k}}))$ with $\boldsymbol{k} \in \mathbb{Z}^n$ such that $a \cdot \boldsymbol{x} = \boldsymbol{z} + m \cdot \boldsymbol{k}$. For $A \in \mathcal{C}$, $B \in \mathcal{C}'$ the zero-opening condition follows directly from the zero-opening conditions of COM and AHE as

$$(A^m, B^m) = (\mathsf{Com}(\mathsf{pp}, \mathbf{0}; R(A)), \mathsf{Enc}(\mathsf{pk}, 0; R'(B))) = \Psi_n(\mathbf{0}; R(A), R'(B)).$$

4.2 Amortizing over Pre-Images

We can apply standard techniques from Σ-protocol theory to amortize the costs of proving knowledge of many pre-images of the same homomorphism Ψ_n. This lets us do so at the same communication costs of a single proof. The amortized protocol is given in Figure 3, where we use the shorthand ϕ_k for the efficiently computable value defined recursively by $\phi_1 := R(\boldsymbol{x}_1, e, \gamma_1)$ and for $j \geq 2$:

$$\phi_j := R(e^j \cdot \boldsymbol{x}_j, \sum_{i=1}^{j-1} e^i \cdot \boldsymbol{x}_i, R(\boldsymbol{x}_j, e^j, \gamma_j), \phi_{j-1}).$$

Figure 3 Amortized Σ-protocol $\Pi_{\mathsf{am\text{-}}\boldsymbol{x}}$ for relation $\mathscr{R}_{\mathsf{multi\text{-}}\boldsymbol{x}}$

Parameters: $m, n, k, \mathcal{S}, \mathbb{H}$
$\mathsf{Input}(C_1, \ldots, C_k, \Psi_n; \boldsymbol{x}_1, \ldots, \boldsymbol{x}_k, \gamma_1, \ldots, \gamma_k)$
$C_i = \Psi_n(\boldsymbol{x}_i; \gamma_i), \forall i$

Prover		**Verifier**
$\boldsymbol{a} \xleftarrow{\$} \mathbb{Z}_m^n, \rho \xleftarrow{\$} \mathcal{S}$		
$A := \Psi_n(\boldsymbol{a}; \rho)$	$\xrightarrow{\quad A \quad}$	
	$\xleftarrow{\quad e \quad}$	$e \xleftarrow{\$} \mathcal{E}$
$\boldsymbol{z} := \boldsymbol{a} + \sum_{i=1}^k e^i \cdot \boldsymbol{x}_i$		
$\sigma := R(\boldsymbol{a}, \sum_{i=1}^k e^i \cdot \boldsymbol{x}_i, \rho, \phi_k)$	$\xrightarrow{\quad \boldsymbol{z}, \sigma \quad}$	$\Psi_n(\boldsymbol{z}; \sigma) \stackrel{?}{=} A * \prod_{i=1}^k C_i^{e^i}$

Theorem 3. *The protocol $\Pi_{\mathsf{am\text{-}}\boldsymbol{x}}$ from Figure 3 is a complete, special honest verifier zero-knowledge, $(k+1)$-special sound protocol for the relation*

$$\mathscr{R}_{\mathsf{multi\text{-}}\boldsymbol{x}} := \{(C_1, \ldots, C_k, \Psi_n; \boldsymbol{x}_1, \ldots, \boldsymbol{x}_k, \gamma_1, \ldots, \gamma_k) : C_i = \Psi_n(\boldsymbol{x}_i; \gamma_i), \forall i \in [k]\}.$$

Proof. **Completeness:** Follows directly from repeatedly applying the homomorphic properties of Ψ_n.

SHVZK: We can construct a simulator on input public parameters $m, n, k, \mathcal{S}, \mathbb{H}$ and a statement $C_1, \ldots, C_k, \Psi_n$ as follows. Sample $e \xleftarrow{\$} \mathcal{E}$, $\sigma \xleftarrow{\$} \mathcal{S}$ and $\boldsymbol{z} \xleftarrow{\$} \mathbb{Z}_m^n$, put $A := \Psi_n(\boldsymbol{z}; \sigma) * \prod_{i=1}^k C_i^{-e^i}$ and output the transcript $(A, e, \boldsymbol{z}, \sigma)$. It is clear that this is an accepting transcript, and that it is distributed identically to a real transcript by the randomization property of $R(\cdot)$.

Special soundness: Let $(A, e_1, \boldsymbol{z}_1, \sigma_1), \ldots, (A, e_{k+1}, \boldsymbol{z}_{k+1}, \sigma_{k+1})$ be $k+1$ accepting transcripts for the same statement $C_1, \ldots, C_k, \Psi_n$, with respect to the same public parameters $m, n, k, \mathcal{S}, \mathbb{H}$ and with pairwise distinct challenges $e_i \neq e_j$ for $i \neq j$. Consider the $(k+1) \times (k+1)$ Vandermonde matrix with entries in $\mathbb{Z}_m$

$$V = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ e_1 & e_2 & \cdots & e_{k+1} \\ \vdots & \vdots & \ddots & \vdots \\ e_1^k & e_2^k & \cdots & e_{k+1}^k \end{pmatrix}, \tag{1}$$

which has determinant $\prod_{i<j}(e_j - e_i) \in \mathbb{Z}_m$. Since $e_j - e_i$ is invertible for all $i \neq j$, V is invertible over $\mathbb{Z}_m$. Hence for each $j \in [k+1]$ there exist $\boldsymbol{b}_j, \boldsymbol{d}_j \in \mathbb{Z}^{k+1}$ such that the j-th unit vector $\mathbf{1}_j$ can be written as $\mathbf{1}_j = V\boldsymbol{b}_j + m \cdot \boldsymbol{d}_j$ over the integers. Recall that the accepting condition gives $\Psi_n(\boldsymbol{z}_j; \sigma_j) = A * \prod_{i=1}^k C_i^{e_j^i}$ for all $j \in [k+1]$. Thus for each $v \in [k]$ we can write

$$\begin{aligned} C_v &= \prod_{j=1}^{k+1} \left(A^{b_{v+1,j}} \prod_{i=1}^k C_i^{e_j^i b_{v+1,j}} \right) * (A^{d_{v+1,1}} \prod_{i=1}^k C_i^{d_{v+1,i+1}})^m \\ &= \prod_{j=1}^{k+1} \Psi_n(\boldsymbol{z}_j; \sigma_j)^{b_{v+1,j}} * \Psi_n(\mathbf{0}; R(A^{d_{v+1,1}} \prod_{i=1}^k C_i^{d_{v+1,i+1}})) \\ &= \Psi_n(\tilde{\boldsymbol{x}}_v; \tilde{\gamma}_v), \end{aligned}$$

where $\tilde{\boldsymbol{x}}_v := \sum_{j=1}^{k+1} b_{v+1,j} \cdot \boldsymbol{z}_j$ and $\tilde{\gamma}_v$ can be computed efficiently from the values $\boldsymbol{z}_j, \sigma_j$, $b_{v+1,j}, d_{v+1,j}, e_j, A, C_i$, for $j \in [k+1]$, $i \in [k]$ using the homomorphic properties of Ψ_n.

Figure 4 Argument of knowledge $\Pi_{\mathsf{am\text{-}mult}}$ for relation $\mathscr{R}_{k\text{-}\mathsf{mult}}$.
Reduction from $\mathscr{R}_{k\text{-}\mathsf{mult}}$ to $\mathscr{R}_{\Psi}$.

Parameters: $m, n, k, \mathsf{pp}, \mathcal{S}, \mathbb{H}$
$\mathsf{Input}(P, B_1, \dots, B_k, \boldsymbol{C}_1, \dots, \boldsymbol{C}_k; \boldsymbol{x}, r, \gamma_1, \dots, \gamma_k)$
$P = \mathsf{Com}(\mathsf{pp}, \boldsymbol{x}; r)$
$B_i = \boldsymbol{C}_i^{\boldsymbol{x}} * \mathsf{Enc}(\mathsf{pk}, 0; \gamma_i)$, for all $i \in [k]$

Prover		**Verifier**
	$\xleftarrow{e}$	$e \xleftarrow{\$} \mathcal{E}$
$\gamma := \phi_k$	$\xrightarrow{\boldsymbol{x}, r, \gamma}$	$\mathsf{Com}(\mathsf{pp}, \boldsymbol{x}; r) \stackrel{?}{=} P$, $\prod_{i=1}^{k} \boldsymbol{C}_i^{e^i \cdot \boldsymbol{x}} * \mathsf{Enc}(\mathsf{pk}, 0; \gamma) \stackrel{?}{=} \prod_{i=1}^{k} B_i^{e^i}$

Hence for each $v \in [k]$ we can efficiently extract a witness $(\tilde{\boldsymbol{x}}_v, \tilde{\gamma}_j)$ such that $C_v = \Psi_n(\tilde{\boldsymbol{x}}_v; \tilde{\gamma}_j)$. □

By plugging the morphism from Example 4 (resp. Example 6) into the protocol from Figure 3, one can prove knowledge of many plaintexts (resp. correctness of many multiplications) at the cost of one. These amortized protocols are main building blocks of our actively secure MPC protocol for vector-RMS programs in Section 6.

4.3 Amortizing Proofs of Correct Multiplication

For some specific classes of morphisms it is possible to amortize the cost of proving knowledge of a pre-image for many morphisms. We will detail below the specific example of proving correctness of many multiplications of a single vector of committed exponents with many different ciphertext vectors, since it will be part of our actively secure MPC protocol in Section 6. More formally, we show that proving knowledge of a witness for the relation

$$\begin{aligned}\mathscr{R}_{k\text{-}\mathsf{mult}} := \{(P, B_1, \dots, B_k, \boldsymbol{C}_1, \dots, \boldsymbol{C}_k; \boldsymbol{x}, r, \gamma_1, \dots, \gamma_k) : \\ P = \mathsf{Com}(\mathsf{pp}, \boldsymbol{x}; r) \wedge B_i = \boldsymbol{C}_i^{\boldsymbol{x}} * \mathsf{Enc}(\mathsf{pk}, 0; \gamma_i), \forall i \in [k]\}\end{aligned}$$

can be reduced to proving knowledge of a witness for the relation

$$\mathscr{R}_{\Psi} := \{(C, \Psi_{n,e}; \boldsymbol{x}, r, \gamma) : C = \Psi_{n,e}(\boldsymbol{x}; r, \gamma)\}, \quad C := (P, \prod_{i=1}^{k} B_i^{e^i}),$$

$$\Psi_{n,e}(\boldsymbol{x}; r, \gamma) := (\mathsf{Com}(\mathsf{pp}, \boldsymbol{x}; r), \prod_{i=1}^{k} \boldsymbol{C}_i^{e^i \cdot \boldsymbol{x}} * \mathsf{Enc}(\mathsf{pk}, 0; \gamma)),$$

for a challenge $e \xleftarrow{\$} \mathcal{E}$ sampled by the verifier. Note that the latter relation is a special case of the relation $\mathscr{R}_{\Psi}$ from Theorem 2. The argument of knowledge protocol for $\mathscr{R}_{k\text{-}\mathsf{mult}}$, which reduces $\mathscr{R}_{k\text{-}\mathsf{mult}}$ to $\mathscr{R}_{\Psi}$, is given in Figure 4. We use the shorthand ϕ_k for the efficiently computable value defined recursively as

$$\phi_i = R(0, 0, R(0, e^i, \gamma_i), \phi_{i-1}) \text{ for } i \geq 2, \ \phi_1 := R(0, e, \gamma_1),$$

where R is the randomization function of AHE.

Theorem 4. *The protocol* $\Pi_{\mathsf{am\text{-}mult}}$ *from Figure 4 is a complete and computationally* $(k+1)$*-special sound protocol for the relation* $\mathscr{R}_{k\text{-}\mathsf{mult}}$*, where soundness holds under the assumption that* COM *is binding.*

Proof. **Completeness:** Follows directly from the homomorphic properties of AHE.

Special soundness: Consider the $k+1$ accepting transcripts $(e_1, \boldsymbol{x}_1, r_1, \gamma_1), \ldots, (e_{k+1}, \boldsymbol{x}_{k+1}, r_{k+1}, \gamma_{k+1})$ for the same statement $P, B_1, \ldots, B_k, \boldsymbol{C}_1, \ldots, \boldsymbol{C}_k$, with respect to the same public parameters $m, n, k, \mathsf{pp}, \mathcal{S}, \mathbb{H}$ and with pairwise distinct challenges $e_i \neq e_j$ for $i \neq j$. By the first part of the verification equation, we have that $\mathsf{Com}(\mathsf{pp}, \boldsymbol{x}_i; r_i) = P$ for all $i \in [k]$. So either we can break the binding property of COM or $\boldsymbol{x}_1 = \boldsymbol{x}_2 = \cdots = \boldsymbol{x}_k =: \boldsymbol{x}$. Again consider the $(k+1) \times (k+1)$ Vandermonde matrix V as in equation (1). Since $e_j - e_i$ is invertible for all $i \neq j$, V is invertible over $\mathbb{Z}_m$. Hence for each $j \in [k+1]$ there exist $\boldsymbol{b}_j, \boldsymbol{d}_j \in \mathbb{Z}^{k+1}$ such that the j-th unit vector $\mathbf{1}_j$ can be written as $\mathbf{1}_j = V\boldsymbol{b}_j + m \cdot \boldsymbol{d}_j$ over the integers. By the second part of the verification equation, we see that for any $v \in [k]$:

$$\begin{aligned}
B_v &= \prod_{j=1}^{k+1} \left(\prod_{i=1}^{k} B_i^{e_j^i b_{v+1,j}} \right) * \left(\prod_{i=1}^{k} B_i^{d_{v+1,i+1}} \right)^m \\
&= \prod_{j=1}^{k+1} \left(\prod_{i=1}^{k} \boldsymbol{C}_i^{e_j^i \cdot \boldsymbol{x}} * \mathsf{Enc}(\mathsf{pk}, 0, \gamma_j) \right)^{b_{v+1,j}} * \prod_{i=1}^{k} B_i^{d_{v+1,i+1}} \\
&= \boldsymbol{C}_v^{\boldsymbol{x}} * \left(\prod_{i=1}^{k} (B_i * \boldsymbol{C}_i^{-\boldsymbol{x}})^{d_{v+1,i+1}} \right)^m * \prod_{j=1}^{k+1} \mathsf{Enc}(\mathsf{pk}, 0; \gamma_j)^{b_{v+1,j}} \\
&= \boldsymbol{C}_v^{\boldsymbol{x}} * \mathsf{Enc}(\mathsf{pk}, 0; \tilde{\gamma}_v),
\end{aligned}$$

where $\tilde{\gamma}_v$ can be computed efficiently using the randomness function R of AHE. Hence we can efficiently extract a witness $\boldsymbol{x}, r, \tilde{\gamma}_1, \ldots, \tilde{\gamma}_k$ such that $P = \mathsf{Com}(\mathsf{pp}, \boldsymbol{x}; r)$ and $B_v = \boldsymbol{C}_v^{\boldsymbol{x}} * \mathsf{Enc}(\mathsf{pk}, 0; \tilde{\gamma}_v)$ for each $v \in [k]$. □

Post-composing the reduction $\Pi_{\mathsf{am\text{-}mult}}$ with the standard protocol Π_Σ from Figure 2 or the compressed protocol Π from Figure 6 lets us prove knowledge of a witness for k multiplication relations at the cost proving knowledge of a witness for a single relation. This allows us to amortize the proofs of essentially all the operations a party performs in the actively secure MPC protocol for RMS programs in Section 6.

Remark 4. Another class of morphisms for which we can amortize the cost of proving knowledge of a pre-image for many different morphisms is that of *linear forms*, which is a main building block in zero-knowledge proofs for arithmetic circuits [AC20]. Since we will not make use of this in our MPC protocols we move the discussion to Appendix A.

5 Compressed Σ-Protocols with Sparse Blinding

In Figure 5 we describe the compression mechanism with which we can reduce proving knowledge of a vector of length n to proving knowledge of a vector of length $n/2$. For n a power of 2 and a vector $\boldsymbol{x} \in \mathbb{Z}_m^n$, we denote by $\boldsymbol{x}_L$ and $\boldsymbol{x}_R$ the left and right halves of $\boldsymbol{x}$, i.e. $\boldsymbol{x} = (\boldsymbol{x}_L, \boldsymbol{x}_R) \in \mathbb{Z}_m^{n/2} \times \mathbb{Z}_m^{n/2}$. We also use as shorthand the efficiently computable function ϕ defined as follows:

$$\phi(\boldsymbol{x}, \gamma, \rho_A, \rho_B, e) := R((e\boldsymbol{x}_L, \boldsymbol{x}_L + e\boldsymbol{x}_R), (e^2\boldsymbol{x}_R, \mathbf{0}), \alpha, \beta),$$

where $\alpha := R((\mathbf{0}, \boldsymbol{x}_L), e\boldsymbol{x}, \rho_A, R(\boldsymbol{x}, e, \gamma))$, $\beta := R((\boldsymbol{x}_R, \mathbf{0}), e^2, \rho_B)$.

Figure 5 Compression Mechanism Π_{C} for relation $\mathscr{R}_{\Psi}$

Parameters: $m, n, \mathcal{S}, \mathbb{H}$
$\mathsf{Input}(C, \Psi_n; \boldsymbol{x}, \gamma)$
$C = \Psi_n(\boldsymbol{x}, \gamma)$

Prover		**Verifier**
$\rho_A, \rho_B \xleftarrow{\$} \mathcal{S}$		
$A := \Psi_n((\mathbf{0}, \boldsymbol{x}_L); \rho_A)$		
$B := \Psi_n((\boldsymbol{x}_R, \mathbf{0}); \rho_B)$	$\xrightarrow{A,B}$	
	$\xleftarrow{e}$	$e \xleftarrow{\$} \mathcal{E}$
$\boldsymbol{z} := \boldsymbol{x}_L + e \cdot \boldsymbol{x}_R$		
$\sigma := \phi(\boldsymbol{x}, \gamma, \rho_A, \rho_B, e)$	$\xrightarrow{\boldsymbol{z},\sigma}$	$\Psi_n((e\boldsymbol{z}, \boldsymbol{z}); \sigma) \stackrel{?}{=} A * C^e * B^{e^2}$

Theorem 5. *The compression mechanism* Π_{C} *from Figure 5 is a complete and 3-special-sound protocol for the relation* $\mathscr{R}_{\Psi}$.

Proof. **Completeness:** Follows from the definition of ϕ and the homomorphic properties of Ψ_n.

Special soundness: Let $(A, B, e_1, \boldsymbol{z}_1, \sigma_1)$, $(A, B, e_2, \boldsymbol{z}_2, \sigma_2)$ and $(A, B, e_3, \boldsymbol{z}_3, \sigma_3)$ be three accepting transcripts for the statement (C, Ψ_n), with common first message (A, B) and distinct challenges $e_1, e_2, e_3 \in \mathcal{E}$. Consider the Vandermonde matrix with entries in $\mathbb{Z}_m$

$$V = \begin{pmatrix} 1 & 1 & 1 \\ e_1 & e_2 & e_3 \\ e_1^2 & e_2^2 & e_3^2 \end{pmatrix},$$

which has determinant $(e_3 - e_2)(e_3 - e_1)(e_2 - e_1)$. Since $e_i - e_j$ is invertible for $i \neq j$, V is invertible over $\mathbb{Z}_m$. Therefore there exist $a_1, a_2, a_3, k_1, k_2, k_3 \in \mathbb{Z}$ such that

$$\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = V \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} + m \begin{pmatrix} k_1 \\ k_2 \\ k_3 \end{pmatrix}$$

holds over the integers. Recall that the accepting condition gives $\Psi_n((e_i\boldsymbol{z}_i, \boldsymbol{z}_i); \sigma_i) = AC^{e_i}B^{e_i^2}$ for $i = 1, 2, 3$. Then

$$\begin{aligned} C &= A^{a_1+a_2+a_3} C^{e_1a_1+e_2a_2+e_3a_3} B^{e_1^2a_1+e_2^2a_2+e_3^2a_3} (A^{k_1} C^{k_2} B^{k_3})^m \\ &= \Psi_n((e_1\boldsymbol{z}_1, \boldsymbol{z}_1); \sigma_1)^{a_1} \Psi_n((e_2\boldsymbol{z}_2, \boldsymbol{z}_2); \sigma_2)^{a_2} \Psi_n((e_3\boldsymbol{z}_3, \boldsymbol{z}_3); \sigma_3)^{a_3} \Psi_n(\mathbf{0}; R(A^{k_1} C^{k_2} B^{k_3})) \\ &= \Psi_n(\tilde{\boldsymbol{x}}; \tilde{\gamma}), \end{aligned}$$

where $\tilde{\boldsymbol{x}} = \sum_{i=1}^{3} a_i(e_i\boldsymbol{z}_i, \boldsymbol{z}_i)$ and $\tilde{\gamma}$ can be computed from the values a_i, k_i, e_i, C, A, B. This shows we can compute a witness $(\tilde{\boldsymbol{x}}, \tilde{\gamma})$ for the statement (C, Ψ_n) and relation $\mathscr{R}_{\Psi}$. □

We show how the compression mechanism can be composed with the basic Σ-protocol to give a zero-knowledge proof of a pre-image of the map Ψ_n, with communication complexity logarithmic in n. Moreover, we require only a logarithmic number of "blinding" elements (consisting of the variables a, ρ_A^i, ρ_B^i). Compared to the naive approach, which uses a blinding vector $\boldsymbol{a}$ of length n, this significantly decreases the computation costs of the prover when the input vector $\boldsymbol{x}$ is sparse.

Consider the protocol Π'_Σ obtained from the basic protocol Π_Σ of Figure 2 by replacing the blinding vector $\boldsymbol{a} \xleftarrow{\$} \mathbb{Z}_m^n$ sampled by the prover by $\boldsymbol{a} := (a, 0, \ldots, 0)$, where $a \xleftarrow{\$} \mathbb{Z}_m$. Clearly Π'_Σ still satisfies completeness and 2-special soundness, although it is no longer zero-knowledge. The compressed protocol can be seen as the composition of this modified basic protocol with the compression mechanism, applied successively until the final message cannot be compressed any more: $\Pi = \Pi_{\mathrm{C}} \diamond \cdots \diamond \Pi_{\mathrm{C}} \diamond \Pi'_\Sigma$ (see Section 2.4.1). A full description is given in Figure 6, where we use the maps Ψ_i defined recursively from $\Psi_n = \Psi_{2^\ell}$ as follows for $i \in \{1, 2, \ldots, 2^{\ell-1}\}$:

$$\Psi_i(\boldsymbol{y}; \rho) := \Psi_{2i}((e_{\ell - \log i + 1}\boldsymbol{y}, \boldsymbol{y}); \rho).$$

Figure 6 Compressed Σ-protocol Π for relation $\mathscr{R}_\Psi$

Parameters:
$m, n = 2^\ell, \mathcal{S}, \mathbb{H}$
$\mathsf{Input}(C, \Psi_n; \boldsymbol{x}, \gamma)$
$C = \Psi_n(\boldsymbol{x}; \gamma)$

Prover		**Verifier**
$a \xleftarrow{\$} \mathbb{Z}_m, \rho \xleftarrow{\$} \mathcal{S}$ $\boldsymbol{a} := (a, 0, \ldots, 0) \in \mathbb{Z}_m^n$ $A_0 := \Psi_n(\boldsymbol{a}; \rho)$	$\xrightarrow{A_0}$	
	$\xleftarrow{e_1}$	$e_1 \xleftarrow{\$} \mathcal{E}$
$\boldsymbol{x}^1 = (\boldsymbol{x}_L^1, \boldsymbol{x}_R^1) := \boldsymbol{a} + e_1\boldsymbol{x}$ $\gamma_1 := R(\boldsymbol{a}, e_1\boldsymbol{x}, \rho, R(\boldsymbol{x}, e_1, \gamma))$		$C_1 := A_0 * C^{e_1}$
$\rho_A^1, \rho_B^1 \xleftarrow{\$} \mathcal{S}$ $A_1 := \Psi_n((\boldsymbol{0}, \boldsymbol{x}_L^1); \rho_A^1)$ $B_1 := \Psi_n((\boldsymbol{x}_R^1, \boldsymbol{0}); \rho_B^1)$	$\xrightarrow{A_1, B_1}$	
	$\xleftarrow{e_2}$	$e_2 \xleftarrow{\$} \mathcal{E}$
$\boldsymbol{x}^2 := \boldsymbol{x}_L^1 + e_2\boldsymbol{x}_R^1$ $\gamma_2 := \phi(\boldsymbol{x}^2, \gamma_1, \rho_A^1, \rho_B^1, e_2)$		$C_2 := A_1 * C_1^{e_2} * B_1^{e_2^2}$
$\vdots$	$\vdots$	$\vdots$
$\rho_A^\ell, \rho_B^\ell \xleftarrow{\$} \mathcal{S}$ $A_\ell := \Psi_2((\boldsymbol{0}, \boldsymbol{x}_L^\ell); \rho_A^\ell)$ $B_\ell := \Psi_2((\boldsymbol{x}_R^\ell, \boldsymbol{0}); \rho_B^\ell)$	$\xrightarrow{A_\ell, B_\ell}$	
	$\xleftarrow{e_{\ell+1}}$	$e_{\ell+1} \xleftarrow{\$} \mathcal{E}$
$\boldsymbol{z} := \boldsymbol{x}_L^\ell + e_{\ell+1}\boldsymbol{x}_R^\ell$ $\sigma := \phi(\boldsymbol{x}^\ell, \gamma_\ell, \rho_A^\ell, \rho_B^\ell, e_{\ell+1})$		$C_{\ell+1} := A_\ell * C_\ell^{e_{\ell+1}} * B_\ell^{e_{\ell+1}^2}$
	$\xrightarrow{\boldsymbol{z}, \sigma}$	$\Psi_1(\boldsymbol{z}; \sigma) \stackrel{?}{=} C_{\ell+1}$

In addition to the homomorphic properties defined in Section 4, the morphism Ψ_n needs to satisfy the following *indistinguishability* condition in order for the protocol in Figure 6 to be computational special honest-verifier zero-knowledge.

- **Indistinguishability:** For a uniform $\gamma \xleftarrow{\$} \mathcal{S}$, the distributions of $\Psi_n(\boldsymbol{x}; \gamma)$ and $\Psi_n(\boldsymbol{0}; \gamma)$ are computationally indistinguishable.

Note that the morphisms in all the examples presented in Section 4.1 satisfy the indistinguishability condition. For the proof of plaintext knowledge in Example 4 this follows from the semantic security of AHE. For the proof of commitment opening in Example 5 this follows from the hiding property of COM. For the proof of correct multiplication in Example 6 this follows from the hiding property of COM and the semantic security of AHE, under the assumption that $C_1, \dots, C_n$ are valid ciphertexts.

Theorem 6. *The compressed protocol* Π *is a complete,* $(2, 3, \dots, 3)$*-special-sound and computational special honest-verifier zero-knowledge protocol for* $\mathscr{R}_\Psi$*.*

Proof. **Completeness:** Follows from the completeness of Π'_Σ and Π_{C}.

Special soundness: Follows from the 2-special-soundness of Π'_Σ and 3-special-soundness of Π_{C}. An extraction algorithm for Π can be obtained by combining the extraction procedures of Π_{C} and Π'_Σ.

Zero-knowledge: The simulator generates a transcript $(A_0, e_1, A_1, B_1, \dots, A_\ell, B_\ell, e_{\ell+1}, \boldsymbol{z}, \sigma)$ as follows. Sample $e_1, \dots, e_{\ell+1} \leftarrow \mathcal{E}$, $\boldsymbol{z} \leftarrow \mathbb{Z}_m$ and $\rho_A^1, \dots, \rho_A^{\ell-1}$, $\rho_B^1, \dots, \rho_B^\ell$, $\rho, \sigma \leftarrow \mathcal{S}$. Then set $A_0 := \Psi_n(\mathbf{0}; \rho)$, $A_i := \Psi_n(\mathbf{0}; \rho_A^i)$ for $1 \le i \le \ell - 1$, and $B_i := \Psi_n(\mathbf{0}; \rho_B^i)$ for $1 \le i \le \ell$. Finally, define $C_1, \dots, C_\ell$ as in the real protocol and let $A_\ell := \Psi_1(\boldsymbol{z}; \sigma) * C_\ell^{-e_{\ell+1}} * B_\ell^{-e_{\ell+1}^2}$.

Let us now check that the distribution of the simulated transcript is computationally indistinguishable from that of an honest transcript. First, the simulated challenges $e_1, \dots, e_{\ell+1}$ are sampled as in the real protocol and the prover's messages $A_1, \dots, A_{\ell-1}$, $B_1, \dots, B_\ell$ are all of the form $\Psi_n(\mathbf{0}, \delta)$ for freshly sampled δ, which are computationally indistinguishable from $\Psi_n(\boldsymbol{y}, \delta)$ for any $\boldsymbol{y}$ by the indistinguishability property of Ψ_n. Observe that in the real protocol $\boldsymbol{z}$ is uniform (even when conditioned on the aforementioned elements of the transcript), due to $a \leftarrow \mathbb{Z}_m$ sampled by the prover. The first message A_0 is indistinguishable in the real and simulated transcripts, since ρ is independent from the other variables discussed so far. Conditioned on the previous transcript elements, σ is uniform in the real protocol since ρ_A is uniform. Finally, in both the real and simulated transcript $A_\ell \in \mathbb{H}$ is the unique element which satisfies the verification equation. □

Remark 5. The communication costs of Π consist of $2\ell + 2$ elements of $\mathbb{Z}_m$ and 1 element of $\mathcal{S}$ sent from prover to verifier, and $\ell + 1$ elements of $\mathbb{Z}_m$ sent from verifier to prover.

Remark 6. We have shown that the protocol Π is computationally SHVZK based on the assumption that $\Psi_n(\boldsymbol{x}; \rho) \approx_c \Psi_n(\mathbf{0}; \rho)$. However, if the map Ψ_n has the stronger property that $\Psi_n(\boldsymbol{x}; \rho)$ is uniformly distributed in $\mathbb{H}$ (for uniform ρ), then the real and simulated transcripts are identically distributed and Π achieves perfect SHVZK. This is the case for Pedersen commitments, for example.

6 Improved Actively Secure MPC for RMS Programs

In this section we present our construction of an actively secure MPC protocol for evaluating a vector-RMS program (see Definition 2). To allow multiple parties to effectively evaluate the program using their private inputs, we assume that for each layer the (encrypted) memory values $C_{t,j}$ and the gates $f_{t,j}$ used to compute them are labeled by an index (t, j) with $t \in [T]$ and $j \in [\ell]$. Moreover, we assume that each gate $f_{t,j}$ contains a description of the party's local input vector and the vector of previous layer memory values that are needed to evaluate the gate. For more details see Figure 7. For conciseness we only consider inner product gates between a party's private input vector and a vector of previous layer memory values in the protocol. Addition gates between memory values, as well as multiplication gates between a public input value and a memory value can simply be evaluated by every party taking part in the protocol, using the properties of the additively homomorphic encryption scheme, without needing to prove anything about

Figure 7 Protocol Π_{RMS} for Maliciously Secure Computation of vector-RMS program

Parameters: Parties $P_1, \ldots, P_n$. Public modulus $m \in \mathbb{N}$, maximum local input vector length M, maximum local input vectors $\nu := \max_i(\nu_i)$.

- *Inputs:* Party P_i holds local input vectors $\{\boldsymbol{x}_{i,k}\}_{k\in[\nu_i]} \subset \mathbb{Z}_m^M$, global input values $\{y_{i,k}\}_{k\in[\mu_i]} \subset \mathbb{Z}_m$.
- *Program:* Vector-RMS program $f = \{f_{t,j}\}_{t\in[T],j\in[\ell]}$ with T layers, where layer T consists of ℓ gates. Gates are of the form $f_{t,j} := \langle (i,k), O\rangle$, where $i \in [n]$, $k \in [\nu_i]$ specifies the local input vector $\boldsymbol{x}_{i,k}$ held by party P_i and $O \in ([t-1]\times[\ell])^M$ specifies the global memory vector. The description of the program moreover includes maps $\mathsf{in}_i : [\nu_i] \to [\ell]$ and sets $\mathsf{Out}_i \subset [\ell]$ for each $i \in [n]$.
- *Mode:* An option $\mathsf{mode} \in \{\mathsf{low.comm}, \mathsf{low.comp}\}$.

Setup. Each party P_i sends $(\mathsf{sid}, \mathsf{id}_i, m)$ to $\mathcal{F}_{\mathsf{KD}}$ and receives $(\mathsf{pk}, \mathsf{sk}_i)$.

- If $\mathsf{mode} = \mathsf{low.comp}$, P_i sends $(\mathsf{sid}, \mathsf{id}_i, \mathsf{pk}, M, m)$ to $\mathcal{F}_{\mathsf{COM\text{-}Gen}}$ and receives $\mathsf{pp} := (\mathsf{pk}, g_1, \ldots, g_M)$.
- If $\mathsf{mode} = \mathsf{low.comm}$, P_i sends $(\mathsf{sid}, \mathsf{id}_i, \mathsf{pk}, \nu \cdot M, m)$ to $\mathcal{F}_{\mathsf{COM\text{-}Gen}}$ and receives $\mathsf{pp} := (\mathsf{pk}, g_1, \ldots, g_{\nu\cdot M})$.

Program evaluation. The RMS program f is evaluated layer by layer, as follows.

- *Load inputs in memory:*
  - For each $k \in [\mu_i]$, party P_i samples $\rho_k \xleftarrow{\$} \mathcal{R}$ and broadcasts $C_{1,\mathsf{in}_i(k)} := \mathsf{Enc}(\mathsf{pk}, y_{i,k}; \rho_k)$ together with $\pi_{\mathsf{in}_i} \leftarrow \Pi_{\mathsf{pk}}(\boldsymbol{C}_{1,\mathsf{in}_i}; \boldsymbol{y}_i, \boldsymbol{\rho})$.
  - If $\mathsf{mode} = \mathsf{low.comp}$, then for each $k \in [\nu_i]$, party P_i samples $\gamma_k \xleftarrow{\$} \mathcal{R}$ and broadcasts $A_{i,k} := \mathsf{Com}(\mathsf{pp}, \boldsymbol{x}_{i,k}; \gamma_k)$.
  - If $\mathsf{mode} = \mathsf{low.comm}$, then party P_i samples $\gamma \xleftarrow{\$} \mathcal{R}$ and broadcasts $A_i := \mathsf{Com}(\mathsf{pp}, \boldsymbol{x}_i; \gamma)$, where $\boldsymbol{x}_i := (\boldsymbol{x}_{i,1}, \ldots, \boldsymbol{x}_{i,\nu_i})$.
  - Party P_i verifies the proofs π_{in_j} for all $j \neq i$, and aborts if any of them do not accept.
- *Evaluate layers ($t = 2, \ldots, T$):*
  - For gate $f_{t,j} = \langle (i,k), O\rangle$, party P_i samples $\rho_{t,j} \xleftarrow{\$} \mathcal{R}$ and broadcasts $C_{t,j} := \boldsymbol{C}_O^{\boldsymbol{x}_{i,k}} * \mathsf{Enc}(\mathsf{pk}, 0; \rho_{t,j})$, where $\boldsymbol{C}_O := (C_{u,v})_{(u,v)\in O}$.
    - If $\mathsf{mode} = \mathsf{low.comp}$, P_i broadcasts $\pi_{i,t,O} \leftarrow \Pi_{\mathsf{low.comp}}((A_{i,k}, C_{t,j})_{(k,j)\in I_{t,O}}, \boldsymbol{C}_O; (\boldsymbol{x}_{i,k}, \gamma_k, \rho_j)_{(k,j)\in I_{t,O}})$, where $I_{t,O} \subset [\nu_i]\times[\ell]$ is the set of indices (k,j) of local input vectors $\boldsymbol{x}_{i,k}$ for which $f_{t,j} = \langle (i,k), O\rangle$ with the same O.
      Party P_i verifies the proofs of all other parties after each round, and aborts if any of them do not accept.
    - If $\mathsf{mode} = \mathsf{low.comm}$, P_i broadcasts $\pi_i \leftarrow \Pi_{\mathsf{low.comm}}(A_i, (C_{t,j}, \boldsymbol{C}'_{t,j})_{(t,j)\in I}; \boldsymbol{x}_i, \gamma_i, (\rho_{t,j})_{(t,j)\in I})$, where $I \subset [T]\times[\ell]$ is the set of indices (t,j) for which $f_{t,j} = \langle (i,k), O\rangle$ for some $k \in [\nu_i]$ and $O \in ([t-1]\times[\ell])^M$, and $\boldsymbol{C}'_{t,j} := (0, \ldots, \boldsymbol{C}_O, \ldots, 0)$ is the vector of length $\nu \cdot M$ with $\boldsymbol{C}_O$ at the k-th entry and zeros at all other entries. (i.e., such that $\boldsymbol{C}'^{\boldsymbol{x}_i}_{t,j} = \boldsymbol{C}_O^{\boldsymbol{x}_{i,k}}$).
      Party P_i verifies the proofs of all other parties at the end of round T, and aborts if any of them do not accept.
- *Output final layer T values:* For each $j \in \mathsf{Out}_i$, the value $C_{T,j}$ will be decrypted to party P_i as follows:
  - Party P_i samples $r_j \xleftarrow{\$} \mathbb{Z}_m$ and $\rho_j \xleftarrow{\$} \mathcal{R}$, and broadcasts $C_{\mathsf{Out}_i,j} := \mathsf{Enc}(\mathsf{pk}, r_j; \rho_j)$ together with $\pi_{\mathsf{Out}_i} \leftarrow \Pi_{\mathsf{pk}}(\boldsymbol{C}_{\mathsf{Out}_i}; \boldsymbol{r}, \boldsymbol{\rho})$.
  - If the proof verifies, for $j \in \mathsf{Out}_i$, each party sends $(\mathsf{sid}, \mathsf{id}_i, (\mathsf{pk}, \mathsf{sk}_i), C_{T,j} * C_r)$ to $\mathcal{F}_{\mathsf{DEC}}$ and receives $\tilde{z}_{T,j} \in \mathbb{Z}_m$.
  - Party P_i computes its output values $z_{T,j} := [\tilde{z}_{T,j} - r \bmod m]$ for $j \in \mathsf{Out}_i$.

these operations. The protocol allows a choice between low communication mode and low computation mode, as described below. In both modes, we make use of the following sub-protocol:

- $\pi \leftarrow \Pi_{\mathsf{pk}}(\boldsymbol{C}; \boldsymbol{x}, \boldsymbol{\gamma})$: The amortized protocol from Figure 3 with the morphism from Example 4, $C_j = \mathsf{Enc}(\mathsf{pk}, x_j; \gamma_j)$, compiled into a non-interactive proof π.

Low Computation Mode ($\mathsf{mode} = \mathsf{low.comp}$) Lower computation complexity at the cost of higher communication complexity by letting each party commit separately to all their local input vectors and amortizing the proofs of correct multiplication over all the gates $f_{t,j} = \langle (i,k), O \rangle$ within a layer t that use the same global memory vector O. In this mode, we use the following sub-protocol:

- $\pi \leftarrow \Pi_{\mathsf{low.comp}}((P_j, B_j)_{j\in[k]}, \boldsymbol{C}; (\boldsymbol{x}_j, \gamma_j, \rho_j)_{j\in[k]})$: The protocol from Figure 3 with the morphism from Example 4, with $(P_j, B_j) = (\mathsf{Com}(\mathsf{pp}, \boldsymbol{x}_j; \gamma_j), \boldsymbol{C}^{\boldsymbol{x}_j} * \mathsf{Enc}(\mathsf{pk}, 0; \rho_j))$, compressed using the mechanism from Section 5 and compiled into a non-interactive proof π.

Low Communication Mode ($\mathsf{mode} = \mathsf{low.comm}$) Lower communication complexity at the cost of higher computation complexity by letting each party commit to all their local input vectors in a single compact commitment and amortizing all proofs of correct multiplication over all layers and gates they evaluate. We make use of the following sub-protocol:

- $\pi \leftarrow \Pi_{\mathsf{low.comm}}(P, (B_j, \boldsymbol{C}_j)_{j\in[k]}; \boldsymbol{x}, \gamma, (\rho_j)_{j\in[k]})$: The protocol from Figure 4 with $P = \mathsf{Com}(\mathsf{pp}, \boldsymbol{x}; \gamma)$ and $B_j = \boldsymbol{C}_j^{\boldsymbol{x}} * \mathsf{Enc}(\mathsf{pk}, 0; \rho_j)$, compressed using the mechanism from Section 5 and compiled into a non-interactive proof π.

Example 7 (PageRank/RiskPropagation)**.** Coming back to the simplified version of the PageRank/RiskPropagation algorithms as discussed in Example 1. Each party P_i has local input vectors $\boldsymbol{a}_{i,k} \in \mathbb{Z}^M$ corresponding to the rows of their local (weighted) adjacency matrix and has global input values x_k^1 the initial attribute values for $k \in [M_i]$, together with an embedding $\phi : [M_i] \to [M]$ of the local graph in the global graph. The corresponding vector-RMS program consists of T layers with $\ell = M$ gates in each layer, where each gate is of the form $f_{t,j} = \langle (i,k), \boldsymbol{d}_{t-1} \rangle$ with $\boldsymbol{d}_{t-1} = ((t-1,1), \ldots, (t-1,M))$ for each $i \in [N]$, $j \in [M]$, $k \in [M_i]$ and $t = 2, \ldots, T$. Note that party P_i evaluates the gates $f_{t,j}$ for $j = \phi(k)$ as $C_{t,j} := \boldsymbol{C}_{\boldsymbol{d}_{t-1}}^{\boldsymbol{a}_{i,k}} * \mathsf{Enc}(\mathsf{pk}, 0, \rho_{t,j})$ for each $k \in [M_i]$.

- $\mathsf{mode} = \mathsf{low.comp}$: P_i publishes M_i commitments $A_{i,k}$. Moreover, $I_{t,\boldsymbol{d}_{t-1}} = \{(k, \phi(k)) : k \in [M_i]\}$, which means that per round $t = 2, \ldots, T$, party P_i amortizes the proofs over all the gates they evaluate in that round. This results in $O(M_i + T \cdot \log M)$ communication overhead and $O(T \cdot M)$ computation overhead. Unfortunately, the sparsity of the input vectors $\boldsymbol{a}_{i,k} \in \mathbb{Z}^M$ will largely be lost after the amortization from Figure 3, since we are proving knowledge of a witness $\sum_{k=1}^{M_i} e^k \cdot \boldsymbol{a}_{i,k}$ for a random challenge e. It is hard to predict how much we save by using sparse blinding compared to full blinding in this case.

- $\mathsf{mode} = \mathsf{low.comm}$: P_i only publishes a single commitment A_i and amortizes the proofs over the gates $f_{t,j}$ for all $t = 2, \ldots, T$ and $j \in \phi([M_i])$. This results in $O(\log M)$ communication overhead and $O(M^2)$ computation overhead. Note that after the reduction from Figure 4, the morphism that we plug into the compressed protocol from Figure 6 is given by (for a random challenge e):

$$\begin{aligned}\Psi(\boldsymbol{a}_i; \gamma, \rho) : &= (\mathsf{Com}(\mathsf{pp}, \boldsymbol{a}_i; \gamma), \prod_{k\in[M_i],\, t\in[T]} \boldsymbol{C}'^{\,e^{k+t\cdot M}\boldsymbol{a}_i}_{t,\phi(k)} * \mathsf{Enc}(\mathsf{pk}, 0; \gamma)) \\ &= (\mathsf{Com}(\mathsf{pp}, \boldsymbol{a}_i; \gamma), \prod_{t=1}^{T} \boldsymbol{C}_{\boldsymbol{d}_{t-1}}^{\sum_{k=1}^{M_i} e^{k+t\cdot M}\boldsymbol{a}_{i,k}} * \mathsf{Enc}(\mathsf{pk}, 0; \gamma)).\end{aligned}$$

The computational complexity of the compressed protocol scales linearly in the evaluation time of Ψ, which is dominated by the $O(M^2)$ complexity of computing the commitment to a vector of length M^2. However, note that the vector $\boldsymbol{a}_i$ will in practice be very sparse, i.e., more than 99% of its entries will be zero, which results in about a 80% saving in the prover's runtime for proving knowledge of a commitment opening as reported in Section 7.3. Since the second component of Ψ has insignificant evaluation time $O(T \cdot M)$ compared to the first component for large inputs, e.g., of size $M = 10^6$, we predict a saving of around 80% in the prover's runtime using our sparse blinding mechanism as opposed to using the full blinding mechanism from [AC20, ACC+22].

Let AHE be a threshold additively homomorphic encryption scheme with message space $\mathbb{Z}_m$ with threshold τ as defined in Section 2.2 and which satisfies the zero-opening condition of Definition 3. Furthermore, let $\mathcal{F}_{\mathsf{KD}}$ be an ideal key distribution functionality which takes input $(\mathsf{sid}, \mathsf{id}_i, m)$ from party P_i for each $i \in [n]$, and outputs $(\mathsf{pk}, \mathsf{sk}_i)$ to party P_i for $i \in [n]$. Let $\mathcal{F}_{\mathsf{DEC}}$ be an ideal distributed decryption functionality, which takes as input $(\mathsf{sid}, \mathsf{id}_i, (\mathsf{pk}, \mathsf{sk}_i), C)$ from at least τ honest parties P_i and outputs the set $\{\mathsf{Dec}((\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n), c) : c \in C\}$ to all parties. Finally, let $\mathcal{F}_{\mathsf{COM\text{-}Gen}}$ be an ideal functionality that takes as input $(\mathsf{sid}, \mathsf{id}_i, \mathsf{pp}', M, m)$ from each party P_i, where pp' are public parameters of a single-value commitment scheme COM' over $\mathbb{Z}_m$, and outputs $\mathsf{pp} := (\mathsf{pp}', g_1, \ldots, g_M)$ to each party, computed as in Figure 1.

Theorem 7. *Let f be a vector-RMS program over $\mathbb{Z}_m$ as defined in Definition 2. Consider the setting where there are n parties $P_1, \ldots, P_n$ and the ideal functionality $\mathcal{F}_{\mathsf{RMS}}$ where each party P_i provides a set of local input vectors and global input values, and receives a subset of final layer memory values. The protocol Π_{RMS} from Figure 7 realizes the functionality $\mathcal{F}_{\mathsf{RMS}}$ against at most $n - \tau$ maliciously corrupted parties with identifiable abort in the random oracle, $\mathcal{F}_{\mathsf{KD}}$, $\mathcal{F}_{\mathsf{DEC}}$, $\mathcal{F}_{\mathsf{COM\text{-}Gen}}$-hybrid model.*

Proof sketch. Since the main structure of the simulation strategy proceeds similarly to [CDN01], we will only provide a sketch of the simulator and stress areas where it differs. Note that by using interactive instead of non-interactive proofs, as in [CDN01], security can also be proven in the standard model at the cost of increased round complexity.

We consider the scenario where an adversary $\mathcal{A}$ maliciously corrupts an arbitrary subset $\mathfrak{c} \subset [n]$ of at most $n - \tau$ parties $(P_i)_{i \in \mathfrak{c}}$. The simulator Sim interacts with the corrupted parties $(P_i)_{i \in \mathfrak{c}}$ and the ideal functionality $\mathcal{F}_{\mathsf{RMS}}$ on behalf of $(P_i)_{i \in \mathfrak{c}}$, takes the role of $\mathcal{F}_{\mathsf{KD}}$, $\mathcal{F}_{\mathsf{DEC}}$, $\mathcal{F}_{\mathsf{COM\text{-}Gen}}$ and simulates the random oracle, as follows.

- To handle the setup, the simulator can run $\mathsf{Gen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n)$ and distribute the secret key shares sk_i to the malicious parties $i \in \mathfrak{c}$. It can subsequently simulate $\mathcal{F}_{\mathsf{COM\text{-}Gen}}$ as in Figure 1 and distribute the public parameters pp to the malicious parties.
- Upon receiving the input ciphertexts and proofs of plaintext knowledge of the malicious parties, the simulator aborts if any of them do not verify. Using standard techniques using rewinding and programming the random oracle the simulator can extract the plaintexts corresponding to the global inputs of the malicious parties.
- The simulator simulates the input messages of the honest parties by broadcasting random encryptions of zero and random commitments to zero vectors, which are indistinguishable by the threshold semantic security of AHE and the hiding property of COM, respectively. Moreover, the simulator uses the zero-knowledge simulator to simulate the corresponding proofs of plaintext knowledge.
- At each layer the simulator verifies the (amortized) proofs sent by the malicious parties and aborts if any of them do not verify or are not consistent with the commitments provided by the malicious parties in the first round. Again using standard techniques using rewinding and programming of the random oracle the

simulator can extract the vectors corresponding to the local inputs of the malicious parties.
- The simulator simulates the layer evaluation messages of the honest parties by broadcasting random encryptions of zero and using the zero-knowledge simulator to simulate the corresponding proofs of correct multiplication.
- After having extracted all the inputs of the malicious parties over the various layers, the simulator sends these to the ideal functionality $\mathcal{F}_{\mathsf{RMS}}$ and receives their outputs $z_{T,j}$ for $j \in \mathsf{Out}_i$ for each $i \in \mathfrak{c}$.
- At the final layer, upon receiving the ciphertexts C_r and proofs of plaintext knowledge from the malicious parties, the simulator aborts if any of the proofs do not verify. Using standard techniques using rewinding and programming the random oracle the simulator can extract the plaintexts corresponding to the random blinding values r of the malicious parties.
- The simulator simulates the honest parties' ciphertexts and proofs of plaintext knowledge similar to before.
- As the malicious parties send a ciphertext C to the ideal threshold decryption functionality, the simulator checks if it matches $C_{T,j} * C_r$ for a $j \in \mathsf{Out}_i$ for some $i \in \mathfrak{c}$ and the corresponding ciphertext C_r the simulator received at the precious step, and if the correct $\mathsf{pk}, \mathsf{sk}_i$ are provided, and aborts if this is not the case. Otherwise, the simulator returns $\tilde{z}_{T,j} := [z_{T,j} + r \bmod m]$ to all of the parties.

□

7 Evaluation of Sparse Blinding Technique

Since the sparse blinding technique does not yield an asymptotic improvement, we have implemented the compressed Σ-protocol construction presented in Sections 3 and 4 as well as the improvements from Section 5.

For the experiments, we are specifically interested in the *prover complexity* of this construction when proving knowledge of a commitment opening. For this, we computed theoretical bounds and evaluated the performance of our implementation. More specifically, we measure the performance of this construction when generating a compressed Σ-protocol for proving knowledge of a pre-image for a homomorphism of the form $\Psi(\mathbf{x};\gamma) = (\mathsf{Com}(\mathbf{x};\gamma), L(\mathbf{x}))$ with $\mathbf{x} \in \mathbb{Z}_m^n$ the secret input vector, γ the randomness used to make a commitment and $L(\cdot)$ an arbitrary linear form defined over $\mathbb{Z}_m$.

We use the compression strategy which consists of computing the new generators A_i, B_i for a homomorphism with half of the input length after receiving each challenge, as in Figure 6[1].

7.1 Bounds on Prover Complexity

We measure the prover complexity in terms of the number of group exponentiations that the prover needs to perform.

We start with an n-dimensional vector $\mathbf{x} \in \mathbb{Z}_m^n$ that has Hamming weight $k \leq n$, i.e., the parameter k defines the sparsity of $\mathbf{x}$.

First, we compute both an upper and a lower bound on the amount of group exponentiations that are performed by the prover. For the lower bound, we assume the relative sparsity k/n of the vector to remain the same after each folding operation. This is of course unlikely to happen, because it would require the nonzero entries in the right and left half to coincide exactly.

[1] An alternative strategy would be to "decompress" the input vector after every iteration such that it fits in the original homomorphism. However, this is strictly more complex than the approach we follow.

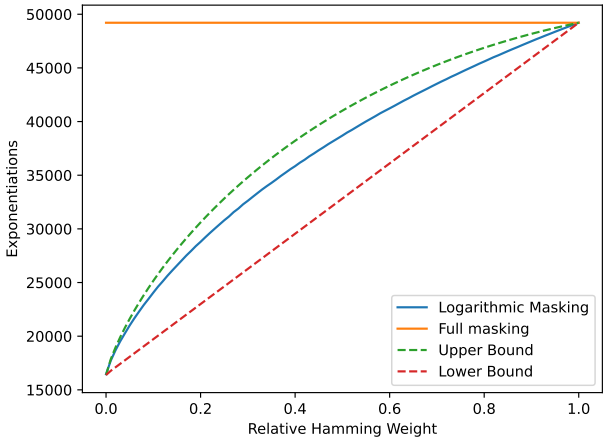


Figure 8: Prover complexity for proving knowledge of a Paillier vector commitment opening using sparse blinding versus full blinding, for vectors of length 2^{14}, and varying Hamming weight, with 2048-bit Paillier keys.

For the upper bound, we assume the absolute sparsity k to remain the same after each folding operation. Concretely, this results in the following bounds for the number of exponentiations:

$$n + 2k + 4\log n + \log \frac{n}{k} \leq \#\mathsf{exp} \leq n + 2k + 4\log n + k \log \frac{n}{k}$$

7.2 Performance

We measure the performance of the implementation for vectors $\mathbf{x}$ of size $|\mathbf{x}| = 2^{14}$ using 2048-bit Paillier keys. Furthermore, we let k range from 0 to 2^{14} where we randomly set k elements to $\frac{m}{2}$ and the other $n - k$ to zero. Because the positions of the zeros can influence the amount of exponentiations required, we run the (randomized) experiments ten times and present the average results. We picked 100 evenly distributed values for k. The results of this experiment, compared to the theoretical bounds, can be found in figure 8.

As expected, blinding a vector with the maximum relative Hamming weight of 1 using a logarithmic number of blinding elements still leads to 49205 exponentiations in total, which is precisely the number of exponentiations always required when fully blinding the vector. For sparser input vectors, we see the amount of exponentiations drop significantly. When half of the input vector are zeros, we already save around 25% of the number of operations. When only 1% of the inputs are non-zero, almost 66% of the exponentiations are saved compared to blinding the entire vector. This is for example the case for the transaction graph application, where each account is known to make only 2-3 transactions per month while a typical dataset contains millions of accounts.

7.3 Runtime

All experiments have been run on a laptop with an AMD Ryzen 7 Pro 4750U with 4.1GHz and 16GB of RAM. Finally, we use the Gmpy2 library to perform modular exponentiations. As can be seen, similar savings are observed compared to the exponentiations, confirming the assumption that the majority of the runtime is dominated by the modular exponentiations. When all entries in the input vector are non-zero, we see that we can compress vectors

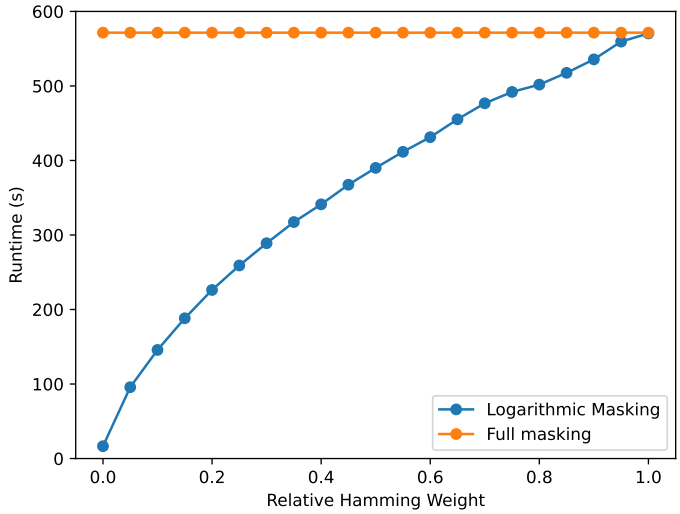


Figure 9: Runtime of compressing a Σ-protocol for proving knowledge for a Paillier vector commitment opening using sparse blinding versus full blinding, for vectors of length 2^{14}, and varying Hamming weight, with 2048-bit Paillier keys.

of 2^{14} Paillier ciphertexts of 2048 bits in around 570 seconds. For vectors with only 1% of non-zero inputs, we see that fully blinding the vector also results in a runtime of 570 seconds while using logarithmic blinding reduces this to only 100 seconds, saving almost 80% of the runtime. This reduction is even greater than the savings in number of exponentiations, which can be explained by the fact that we also save some runtime in other parts of the algorithm due to how the compression is currently implemented. Namely, all the non-zero entries of the vector are first filtered and put into a list, after which this list is iterated to construct the commitments.

Note that currently the improved runtime for sparse inputs could lead to a side-channel attack that leaks the sparsity of the input vector of the prover. In practice, this could be overcome by assuming an upper bound in the sparsity of the input and simply adding some dummy operations. Furthermore, the implementation should also be changed to run in constant time regardless of the sparsity of the input.

Acknowledgements

The research activities that led to this result were supported by ABN AMRO, CWI, Rabobank, TMNL, De Volksbank and PPS-surcharge for Research and Innovation of the Dutch Ministry of Economic Affairs and Climate Policy, and TNO's Appl.AI programme; A. van Baarsen is supported by NWO/TKI Grant 628.009.014; P. Capitão and L. Kohl are supported by the NWO Gravitation Project QSC; L. Kohl is additionally supported by the NWO Talent Programme Veni (VI.Veni.222.348).

References

[AC20] Thomas Attema and Ronald Cramer. Compressed Σ-protocol theory and practical application to plug & play secure algorithmics. In *CRYPTO*

(3), volume 12172 of *Lecture Notes in Computer Science*, pages 513–543. Springer, 2020. doi:10.1007/978-3-030-56877-1_18.

[ACC+22] Thomas Attema, Ignacio Cascudo, Ronald Cramer, Ivan Damgård, and Daniel Escudero. Vector commitments over rings and compressed Σ-protocols. In *TCC (1)*, volume 13747 of *Lecture Notes in Computer Science*, pages 173–202. Springer, 2022. doi:10.1007/978-3-031-22318-1_7.

[ACK21] Thomas Attema, Ronald Cramer, and Lisa Kohl. A compressed Σ-protocol theory for lattices. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 549–579, Virtual Event, August 2021. Springer, Heidelberg. doi:10.1007/978-3-030-84245-1_19.

[AFO+21] Toshinori Araki, Jun Furukawa, Kazuma Ohara, Benny Pinkas, Hanan Rosemarin, and Hikaru Tsuchida. Secure graph analysis at scale. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, page 610–629, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3460120.3484560.

[Att23] Thomas Attema. *Compressed Σ-Protocol Theory*. PhD thesis, Leiden University, 2023.

[BBB+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Gregory Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society, 2018. doi:10.1109/SP.2018.00020.

[BCC+16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In *EUROCRYPT (2)*, volume 9666 of *Lecture Notes in Computer Science*, pages 327–357. Springer, 2016. doi:10.1007/978-3-662-49896-5_12.

[BCG+17] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, and Michele Orrù. Homomorphic secret sharing: Optimizations and applications. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 2105–2122. ACM Press, October / November 2017. doi:10.1145/3133956.3134107.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In *TCC (B2)*, volume 9986 of *Lecture Notes in Computer Science*, pages 31–60, 2016. doi:10.1007/978-3-662-53644-5_2.

[BDO23] Lennart Braun, Ivan Damgård, and Claudio Orlandi. Secure multiparty computation from threshold encryption based on class groups. In *CRYPTO (1)*, volume 14081 of *Lecture Notes in Computer Science*, pages 613–645. Springer, 2023. doi:10.1007/978-3-031-38557-5_20.

[BDOZ11] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 169–188. Springer, 2011. doi:10.1007/978-3-642-20465-4_11.

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under DDH. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 509–539. Springer, Heidelberg, August 2016. doi:10.1007/978-3-662-53018-4_19.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10. ACM Press, May 1988. doi:10.1145/62212.62213.

[BKS19] Elette Boyle, Lisa Kohl, and Peter Scholl. Homomorphic secret sharing from lattices without FHE. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 3–33. Springer, Heidelberg, May 2019. doi:10.1007/978-3-030-17656-3_1.

[BP98] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Comput. Networks*, 30(1-7):107–117, 1998. doi:10.1016/S0169-7552(98)00110-X.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *20th ACM STOC*, pages 11–19. ACM Press, May 1988. doi:10.1145/62212.62214.

[CCL+20] Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. Bandwidth-efficient threshold EC-DSA. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 266–296. Springer, Heidelberg, May 2020. doi:10.1007/978-3-030-45388-6_10.

[CDN01] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. Multiparty computation from threshold homomorphic encryption. In *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 280–299. Springer, 2001. doi:10.1007/3-540-44987-6_18.

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 103–118. Springer, Heidelberg, May 1997. doi:10.1007/3-540-69053-0_9.

[CL15] Guilhem Castagnos and Fabien Laguillaumie. Linearly homomorphic encryption from DDH. In Kaisa Nyberg, editor, *CT-RSA 2015*, volume 9048 of *LNCS*, pages 487–505. Springer, Heidelberg, April 2015. doi:10.1007/978-3-319-16715-2_26.

[CLT18] Guilhem Castagnos, Fabien Laguillaumie, and Ida Tucker. Practical fully secure unrestricted inner product functional encryption modulo p. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 733–764. Springer, Heidelberg, December 2018. doi:10.1007/978-3-030-03329-3_25.

[COS+22] Ilaria Chillotti, Emmanuela Orsini, Peter Scholl, Nigel P. Smart, and Barry Van Leeuwen. Scooby: Improved multi-party homomorphic secret sharing based on FHE. In *SCN*, volume 13409 of *Lecture Notes in Computer Science*, pages 540–563. Springer, 2022. doi:10.1007/978-3-031-14791-3_24.

[CSA21] Daniele Cozzo, Nigel P. Smart, and Younes Talibi Alaoui. Secure fast evaluation of iterative methods: With an application to secure PageRank. In Kenneth G. Paterson, editor, *CT-RSA 2021*, volume 12704 of *LNCS*, pages 1–25. Springer, Heidelberg, May 2021. doi:10.1007/978-3-030-75539-3_1.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. doi:10.1109/TIT.1976.1055638.

[DHRW16] Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky encryption and its applications. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 93–122. Springer, Heidelberg, August 2016. doi:10.1007/978-3-662-53015-3_4.

[DJ00] Ivan B. Damgård and Mads J. Jurik. Efficient protocols based on probabilistic encryption using composite degree residue classes, Jan. 2000. doi:10.7146/brics.v7i5.20133.

[DJ01] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136. Springer, 2001. doi:10.1007/3-540-44586-2_9.

[dPLS19] Raël del Pino, Vadim Lyubashevsky, and Gregor Seiler. Short discrete log proofs for FHE and ring-lwe ciphertexts. In *Public Key Cryptography (1)*, volume 11442 of *Lecture Notes in Computer Science*, pages 344–373. Springer, 2019. doi:10.1007/978-3-030-17253-4_12.

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 643–662. Springer, 2012. doi:10.1007/978-3-642-32009-5_38.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985. doi:10.1109/TIT.1985.1057074.

[FGJS17] Nelly Fazio, Rosario Gennaro, Tahereh Jafarikhah, and William E. Skeith. Homomorphic secret sharing from Paillier encryption. In Tatsuaki Okamoto, Yong Yu, Man Ho Au, and Yannan Li, editors, *Provable Security*, pages 381–399, Cham, 2017. Springer International Publishing. doi:10.1007/978-3-319-68637-0_23.

[FPS00] Pierre-Alain Fouque, Guillaume Poupard, and Jacques Stern. Sharing decryption in the context of voting or lotteries. In *Financial Cryptography*, volume 1962 of *Lecture Notes in Computer Science*, pages 90–104. Springer, 2000. doi:10.1007/3-540-45472-1_7.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987. doi:10.1145/28395.28420.

[OSY21] Claudio Orlandi, Peter Scholl, and Sophia Yakoubov. The rise of paillier: Homomorphic secret sharing and public-key silent OT. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 678–708. Springer, Heidelberg, October 2021. doi:10.1007/978-3-030-77870-5_24.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999. doi:10.1007/3-540-48910-X_16.

[Ped91a] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, 1991. doi:10.1007/3-540-46766-1_9.

[Ped91b] Torben P. Pedersen. A threshold cryptosystem without a trusted party (extended abstract) (rump session). In Donald W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 522–526. Springer, Heidelberg, April 1991. doi:10.1007/3-540-46416-6_47.

[RB89] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *21st ACM STOC*, pages 73–85. ACM Press, May 1989. doi:10.1145/73007.73014.

[RS21] Lawrence Roy and Jaspal Singh. Large message homomorphic secret sharing from DCR and applications. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part III*, volume 12827 of *LNCS*, pages 687–717, Virtual Event, August 2021. Springer, Heidelberg. doi:10.1007/978-3-030-84252-9_23.

[SvHA+19] Alex Sangers, Maran van Heesch, Thomas Attema, Thijs Veugen, Mark Wiggerman, Jan Veldsink, Oscar Bloemen, and Daniël Worm. Secure multiparty pagerank algorithm for collaborative fraud detection. In *Financial Cryptography*, volume 11598 of *Lecture Notes in Computer Science*, pages 605–623. Springer, 2019. doi:10.1007/978-3-030-32101-7_35.

[vEDvdB+24] Marie Beth van Egmond, Vincent Dunning, Stefan van den Berg, Thomas Rooijakkers, Alex Sangers, Ton Poppe, and Jan Veldsink. Privacy-preserving anti-money laundering using secure multi-party computation. Cryptology ePrint Archive, Paper 2024/065, 2024. URL: https://eprint.iacr.org/2024/065.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986. doi:10.1109/SFCS.1986.25.

A Linear Forms and Arithmetic Circuits

Similar to [ACC+22, Prot. 1], we can extend the protocol for proving knowledge of an opening of a commitment from Example 5 to additionally prove that an opening $(\boldsymbol{x}, \gamma)$ satisfies some linear constraint $L(\boldsymbol{x}) = y \in \mathbb{Z}_m$, where $L : \mathbb{Z}_m^n \to \mathbb{Z}_m$ is a linear form, by letting $\Psi_n(\boldsymbol{x}; \gamma) := (\mathsf{Com}(\mathsf{pp}, \boldsymbol{x}; \gamma), L(\boldsymbol{x}))$. As in the reduction in [AC20, Prot. 3], it is possible to reduce proving knowledge of a witness for the morphism $\Psi_n(\boldsymbol{x}; \gamma) := (\mathsf{Com}(\mathsf{pp}, \boldsymbol{x}; \gamma), L(\boldsymbol{x}))$ to proving knowledge of a witness for the morphism $\tilde{\Psi}_n(\boldsymbol{x}; \gamma) := \mathsf{Com}(\mathsf{pp}, (\boldsymbol{x}, e \cdot L(\boldsymbol{x})); \gamma)$ for a random challenge $e \xleftarrow{\$} \mathcal{E}$ sampled by the verifier. This comes at the cost of achieving computational soundness under the assumption that the commitment scheme is binding and one extra round of communication (in which the verifier sends e), but approximately halves the communication cost of the complete protocol when post-composing the reduction with the compressed protocol Π from Figure 6.

When proving knowledge of a witness for many linear forms, i.e. for the morphism $\Psi_n(\boldsymbol{x}, \gamma) := (\mathsf{Com}(\mathsf{pp}, \boldsymbol{x}; \gamma), L_1(\boldsymbol{x}), \ldots, L_k(\boldsymbol{x}))$ one can first apply a reduction similar to [AC20, Prot. 7] to reduce this to proving knowledge of a witness for a single linear form $\tilde{L}_e(\boldsymbol{x}) := \sum_{i=1}^{k} e^i \cdot L_i(\boldsymbol{x})$. Similar to the proof of Theorem 4 this reduction protocol is computationally sound under the assumption that the commitment scheme is binding. Subsequently, one can apply the reduction described in the previous paragraph to reduce to proving knowledge of just a commitment opening. When finally composing with the compressed protocol Π from Figure 6 this saves approximately a factor k in communication.

This shows that the concrete communication costs when using the techniques from [ACC+22] to prove statements about arithmetic circuits can be lowered using the techniques of [AC20]. Additionally this enables us to use the sparse blinding mechanism from Section 5 when proving statements about linear forms and arithmetic circuits in the more general ring setting of [ACC+22], since the morphism $\Psi_n(\boldsymbol{x}, \gamma) := (\mathsf{Com}(\mathsf{pp}, \boldsymbol{x}; \gamma), L(\boldsymbol{x}))$ does *not* satisfy the indistinguishability condition as defined in Section 4 but the morphism $\tilde{\Psi}_n(\boldsymbol{x}; \gamma) := \mathsf{Com}(\mathsf{pp}, (\boldsymbol{x}, e \cdot L(\boldsymbol{x})); \gamma)$ *does*. Another option would be to define the morphism as $\hat{\Psi}(\boldsymbol{x}; \gamma, \rho) := (\mathsf{Com}(\mathsf{pp}, \boldsymbol{x}; \gamma), L(\boldsymbol{x}) + \rho)$ with $\rho \in \mathbb{Z}_m$ so that it fits our framework for sparse blinding, but this approximately doubles the communication compared to the other approach. We leave it to future work to explore more applications where our sparse blinding mechanism can be useful to reduce the prover's computational complexity.

B Other AHE Schemes

Example 8 (Exponential ElGamal)**.** ElGamal's cryptosystem [ElG85] can be turned from a multiplicatively into an additively homomorphic encryption scheme by encrypting a message $x \in \mathbb{Z}_p$ in the exponent as $(c_1, c_2) := (g^r, h^r \cdot g^x)$, where $\mathbb{G} = \langle g \rangle$ is a cyclic group of prime order p, $h = g^s$ is the public key and $s \in \mathbb{Z}_p$ is the private key [CGS97]. The downside of this adaptation is that decryption requires solving the discrete logarithm of $c_1^{-s} \cdot c_2$ with respect to g, which requires to limit the message space to $\mathbb{Z}_m$ for $m \ll p$, e.g., $m = 2^{32}$, to make sure the discrete logarithm is computable in reasonable time. ElGamal's cryptosystem is IND-CPA secure under the decisional Diffie-Hellman (DDH) assumption [DH76], and threshold security with a corresponding key distribution and threshold decryption protocol were shown by Pederson [Ped91b, Ped91a]. Exponential ElGamal satisfies all the conditions necessary to fit in our framework (including the zero-opening condition from Definition 3) if considered with $\mathbb{Z}_p$ as the message space, except for the existence of an efficient decryption protocol. Therefore, when used in the MPC protocol for RMS programs Figure 7, the parties additionally need to prove that all the plaintexts and committed exponents are small enough to allow for efficient decryption. For this purpose the efficient *range proofs* of [AC20] can be adapted to our setting. We leave it to future work to explore specific parameters and whether this approach can be

competitive with a Paillier-based approach.

Example 9 (HSM-CL Encryption)**.** Castagnos, Laguillaumie and Tucker [CLT18] introduced a linearly homomorphic encryption scheme based on the hard subgroup membership assumption (HSM). This scheme can be instantiated in the so-called CL framework [CL15] of a group of unknown order which has a subgroup where the discrete logarithm problem is easy, for which a main candidate is the class group of a non-maximal order in an imaginary quadratic number field. We will not go into the details, but in spirit the HSM-CL encryption scheme is very similar to the exponential ElGamal scheme where during decryption the discrete logarithm can be efficiently solved in the subgroup where the discrete logarithm problem is easy. Braun, Damgård and Orlandi [BDO23] present a key distribution and threshold decryption protocol for the HSM-CL encryption scheme and extend the proof of plaintext knowledge of [CCL+20] with a proof of correct multiplication. The scheme fits almost all the necessary conditions to fit in our framework except for the zero-opening condition from 3. This causes issues with the (special) knowledge soundness of our Σ-protocols, which is common for hidden order groups. Braun et al. solve this issue by showing that for their MPC protocol it is sufficient to satisfy a relaxed notion of soundness, where the plaintext can be extracted but not the randomness. We leave it to future work to extend our results to the hidden order group setting.

Example 10 (Lattice-Based Encryption)**.** Lattice-based AHE constructions are easy to obtain; in fact, standard encryption schemes based on Learning with Errors (LWE) or Ring-LWE enjoy these properties. In the context of these schemes it is necessary to prove not only knowledge of a preimage of a homomorphism, but also that such a preimage is *short*. Although this means the techniques in this work cannot be applied directly, they can be used in conjunction with the compressed Σ-protocols for lattices developed in [ACK21]. The resulting proofs enjoy statistical zero-knowledge and unconditional special soundness, but proving that preimages are short introduces a *soundness slack*: a prover who knows a preimage x such that $\|x\| \leq t$ can only convince a verifier that $\|x\| \leq \alpha t$ for some factor $\alpha > 1$. Soundness slack has an impact on efficiency, as it demands choosing larger parameters to achieve the desired level of security.

Another approach to proving statements about lattice-based encryption is presented in [dPLS19], using Pederson commitments. These proofs have statistical zero-knowledge and computational soundness based on the discrete logarithm problem. Compared to lattice-based compressed Σ-protocols, this technique allows shorter proofs but has higher computation costs.