Communication-Efficient Multi-Party Computation for RMS Programs

Thomas Attema\textsuperscript{1,2}, Aron van Baarsen\textsuperscript{2,3}, Stefan van den Berg\textsuperscript{1}, Pedro Capitão\textsuperscript{2,3}, Vincent Dunning\textsuperscript{1} and Lisa Kohl\textsuperscript{2}

\textsuperscript{1} TNO, Applied Cryptography and Quantum Algorithms, The Hague, The Netherlands
\textsuperscript{2} CWI, Cryptology Group, Amsterdam, The Netherlands
\textsuperscript{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.


RMS programs. In this work, we focus on computing restricted multiplication straight-line (RMS) programs. In programs of this type, only additions and multiplications with input values are performed. This means 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, 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 $\mathsf{NC}^1$) [BGI16].

Graph algorithms. 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 $\widetilde{O}(M \cdot B)$, 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.

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], we can...
We show that these improvements allow us to reduce the computation complexity to $O(M^2 \cdot B)$ 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, achieving a communication complexity which scales \textit{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 \Sigma-protocol framework of [ACC+22] to be compatible with the underlying additively homomorphic encryption scheme, we present a new instantiation of compressed \Sigma-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 \Sigma-protocols [AC20, ACC+22] can be applied to such statements.

Overall, these techniques lead to an improved per-party communication overhead of an additional $O(\log M)$ group elements, where $M$ is the number of nodes in the graph, resulting in the first maliciously secure protocol where the communication complexity scales with the number of nodes rather than edges. We refer to this as the \textit{low.comm} version of our protocol. Since our protocol 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 \textit{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 \textit{vector-RMS programs}.
- We deploy the amortization technique for \Sigma-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,

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 $\mathcal{B}$ a known upper bound on $B$ (i.e., for sparse graphs $\mathcal{B}$ is a small constant and for dense graphs $\mathcal{B}$ equals the number of nodes $M$). The entries \textit{low.comm} and \textit{low.comp} indicate the communication and computation optimized variants of our protocol, respectively. The overall complexity of the actively secure protocol equals \textbf{Passive + Overhead Active}.

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Corruption Threshold</th>
<th>Communication (messages pp)</th>
<th>Computation (operations pp)</th>
<th>Rounds</th>
<th>Identifiable Abort</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Passive</td>
<td>Overhead</td>
<td>Active</td>
<td>Passive</td>
</tr>
<tr>
<td>[DPZ22]</td>
<td>SS+MAC</td>
<td>$t \leq n-1$</td>
<td>$TM^2$</td>
<td>$TM^2$</td>
<td>$TM^2$</td>
</tr>
<tr>
<td>[AFO+21]</td>
<td>Outs. SS</td>
<td>1-out-of-3</td>
<td>$TM^2 \log^2 M$</td>
<td>$TM^2 \log^2 M$</td>
<td>$TM^2 \log^2 M$</td>
</tr>
<tr>
<td>[CDN01]</td>
<td>HR+2KP</td>
<td>$t \leq \frac{n}{2}$</td>
<td>$TM_i$</td>
<td>$TM^2$</td>
<td>$TM_i$</td>
</tr>
<tr>
<td>low_comm</td>
<td>HR+2KP</td>
<td>$t \leq n-1$</td>
<td>$TM_i$</td>
<td>$\log M$</td>
<td>$TM_i$</td>
</tr>
<tr>
<td>low_comp</td>
<td>HR+2KP</td>
<td>$t \leq n-1$</td>
<td>$TM_i$</td>
<td>$M + \log M$</td>
<td>$TM_i$</td>
</tr>
</tbody>
</table>
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 $\Sigma$-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 $g^x$ is equal to the Hamming weight of $x$, 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.comm 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 $c_1 := \text{Enc}(\text{pk}, x_1)$ and $c_2 := \text{Enc}(\text{pk}, x_2)$ can be computed by any party as $c_1 \cdot c_2 := \text{Enc}(\text{pk}, x_1 + x_2)$ and multiplications between an encrypted value $c := \text{Enc}(\text{pk}, x)$ and an input value $a \in \mathbb{Z}_m$ can be evaluated locally by the party holding the input as $c^a = \text{Enc}(\text{pk}, a \cdot x)$. 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 $j \in V^t$. 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 $z^t_j := \prod_{i \in S(j)} z^t_{i-1}$, which can be seen as an inner product between the $j$-th row $\mathbf{a}_j^t$ of their private adjacency matrix $A^t$ and the vector of previous round encrypted PageRank values $z^{t-1} := (z_{i-1}^t)_{i \in [M]}$. In the final round $T$, the parties perform a joint distributed decryption, such that each party learns the decryption of $z^T_j$ for $j \in V^t$. 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 $\Sigma$-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 $\Sigma$-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 $\Sigma$-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 $g^w$, which scale with the number of
non-zero entries of \( x \). In standard \( \Sigma \)-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 \( \Sigma \)-protocols. Intuitively, they reveal less information to the verifier (due to the reduced communication complexity), and thus require less blinding than a standard \( \Sigma \)-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 \( \Sigma \)-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: \( \text{Load}(I_x) \rightarrow S_x \).
- Add values in memory: \( \text{Add}(S_x, S_y) \rightarrow S_{x+y} \).
- Multiply input by memory value: \( \text{Mult}(I_x, S_y) \rightarrow S_{xy} \).
- Output memory value: \( \text{Output}(S_x) \rightarrow x \in \mathcal{R} \).

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 \( y \in \mathcal{R}^n \). 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 \( S^t_x \) the representation of a value \( x \in \mathcal{R} \) as a layer \( t \) memory value and \( I_y \) for the representation of a vector \( y \in \mathcal{R}^n \) as a local input value, and write \( S^t_y := (S^t_{y_1}, \ldots, S^t_{y_n}) \) for a vector of memory values with \( x := (x_1, \ldots, x_n) \in \mathcal{R}^n \) and \( u := (u_1, \ldots, u_n) \in [T]^n \).
• Load inputs into memory: \( \text{Load}(I_x) \rightarrow S^1_x \).
• For each layer \( t = 2, \ldots, T \):
  - Add values in memory: \( \text{Add}(S^u_x, S^v_y) \rightarrow S^u_{x+y} \) for \( u, v \in [t-1] \).
  - Multiply input by memory value: \( \text{Mult}(I_x, S^u_y) \rightarrow S^u_{xy} \) for \( u \in [t-1]^n \).
  - Compute inner product: \( \text{Prod}_n(S^u_x, I_y) \rightarrow S^u_{(x,y)} \) for \( u \in [t-1]^n \).
• Output memory value: \( \text{Output}(S^T_x) \rightarrow x \in \mathcal{R} \).

**Example 1** (PageRank/RiskPropagation). We recall the simplified description of the PageRank/RiskPropagation algorithms presented in the introduction. The inputs to the algorithm consist of a set \( V \) consisting of \( M \) nodes together with an initial attribute value \( x_j \) for each node \( j \in V \), the \( M \times M \) (weighted) adjacency matrix \( A := (a_{ij})_{i,j \in [M]} \) 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: \( \text{Load}(I_{x_j}) \rightarrow S^1_{x_j} \).
- For each round \( t = 2, \ldots, T \):
  - Compute the inner product between \( a_{ij} \) and the previous round attribute values \( x^{t-1} := (x_1^{t-1}, \ldots, x_M^{t-1}) \): \( \text{Prod}_M(S^{t-1}_{x^{t-1}}, I_{a_{ij}}) \rightarrow S^t_{x_j} \).
- Output the round \( T \) attribute values: \( \text{Output}(S^T_{x_j}) \rightarrow x^T_j \).

### 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 \( \text{AHE} := (\text{Gen}, \text{Enc}, \text{Dec}) \) together with a key distribution protocol \( \Pi_{\text{KD}} \), a threshold decryption protocol \( \Pi_{\text{DEC}} \), 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 \( (\mathcal{C}_\lambda, \ast)_\lambda \) such that:

- \( \text{Gen}(1^\lambda) \rightarrow (\text{pk}, \text{sk}_1, \ldots, \text{sk}_n) \). Takes as input the security parameter \( \lambda \) and outputs a public key \( \text{pk} \) and secret key shares \( \text{sk}_1, \ldots, \text{sk}_n \).
- \( \Pi_{\text{KD}} \) is an \( n \)-party protocol, secure against \( n-t \) corruptions, which takes as input the security parameter \( \lambda \), computes \( (\text{pk}, \text{sk}_1, \ldots, \text{sk}_n) \leftarrow \text{Gen}(1^\lambda) \) and outputs \( (\text{pk}, \text{sk}_i) \) to party \( P_i \) for each \( i \in [n] \).
- \( \text{Enc}(\text{pk}, x; \rho) \rightarrow c \). Takes as input a public key \( \text{pk} \), a plaintext message \( x \in \mathcal{M}_\lambda \) and randomness \( \rho \in \mathcal{R}_\lambda \), and outputs a ciphertext \( c \in \mathcal{C}_\lambda \).
- \( \text{Dec}((\text{pk}, \text{sk}_1, \ldots, \text{sk}_n), c) \rightarrow x \). Takes as input a public key \( \text{pk} \), corresponding secret key shares \( (\text{sk}_1, \ldots, \text{sk}_n) \) and a ciphertext \( c \in \mathcal{C}_\lambda \) and outputs a plaintext message \( x \in \mathcal{M}_\lambda \).
- \( \Pi_{\text{DEC}} \) is an \( n \)-party protocol, secure against \( n-t \) corruptions, which takes as input a set of ciphertexts \( C \), a public key \( \text{pk} \) and corresponding secret key shares \( \text{sk}_i \) from all honest parties, and outputs \( \{\text{Dec}((\text{pk}, \text{sk}_1, \ldots, \text{sk}_n), c) : c \in C\} \) to all parties.
- \( R_{\text{AHE}}^{\text{pk}}(\cdot) \rightarrow \rho \). 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 \((pk, sk_1, \ldots, sk_n) \leftarrow \text{Gen}(1^\lambda)\), where we often omit the subscripts \( \lambda \) 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 \( c \leftarrow \text{Enc}(pk, x; \rho) \), it holds that \( \text{Dec}(pk, sk, c) \) is equal to \( x \).

- **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 \( c_1, c_2 \leftarrow \text{Enc}(pk, x_1; \rho_1), c_2 \leftarrow \text{Enc}(pk, x_2; \rho_2) \), it holds that
  \[
  c_1 \cdot c_2 = \text{Enc}(pk, x_1 + x_2; \text{R}(x_1, x_2, \rho_1, \rho_2)),
  \]
  \[
  c_1^a = \text{Enc}(pk, a \cdot x_1; \text{R}(x_1, a, \rho_1)).
  \]

- **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, \( \text{R}(x_1, x_2, \rho_1, \rho_2) \) is distributed uniformly in \( \mathcal{R} \).

- **Threshold Semantic Security**: Let \( \mathcal{A} \) be any PPT algorithm that receives as input \( 1^\lambda \), the public key \( pk \) and the secret keys \( \{sk_i\}_{i \in C} \) 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 \( s \in \{0, 1\}^\lambda \). Let

\[
X_i(\lambda, C) := \left\{ (s, c_1 \leftarrow \text{Enc}(pk, x_i; \rho)) \left| (pk, sk_1, \ldots, sk_n) \leftarrow \text{Gen}(1^\lambda), (x_0, x_1, s) \leftarrow \mathcal{A}(1^\lambda, C, pk, sk_{\lambda}, \{sk_i\}_{i \in C}) \right| \rho \leftarrow \mathcal{R}_\lambda \right\}
\]

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

**Remark 1.** Note that by the randomization property, multiplying any ciphertext \( c := \text{Enc}(pk, x; \rho) \) with a fresh encryption of zero \( \rho_0 := \text{Enc}(pk, 0; \rho_0) \), with \( \rho_0 \leftarrow \mathcal{R} \), results in a ciphertext \( c + \rho_0 = \text{Enc}(pk, x; \text{R}(x, 0, \rho, \rho_0)) \) 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 \( 1^\lambda \), samples \( (N, p, q) \) where \( N = pq \) and \( p, q \) are \( \lambda \)-bit primes (with all but negligible probability in \( \lambda \)) and outputs \( pk := N, sk := (N, \phi(N)) \), where \( \phi \) is the Euler phi function.

- **Enc**: On input a public key \( pk := N \), a message \( x \in \mathbb{Z}_N \) and randomness \( \rho \in \mathbb{Z}_N^* \), outputs \( c := [(1 + N)x \cdot \rho^N \mod N^2] \).

- **Dec**: On input a private key \( sk = (N, \phi(N)) \) and a ciphertext \( c \in \mathbb{Z}_N^{*2} \), computes \( y := \left[ \frac{\phi(N)}{N} \mod N^{N-2} \right]^{-1} \) and outputs \( x := [y \cdot \phi(N)^{-1} \mod N] \).

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 \( (N, [r^N \mod N^2]) \), over \( (N, p, q) \leftarrow \mathcal{Gen}(1^\lambda) \) and \( r \leftarrow \mathbb{Z}_N^{*2} \), are computationally indistinguishable. The homomorphic and randomization properties follow directly from the fact that the map \( \psi(a, b) := [(1 + N)a \cdot b^N \mod N^2] \) is an isomorphism \( \psi : \mathbb{Z}_N \times \mathbb{Z}_N^{*2} \to \mathbb{Z}_N^{*2} \) and \( \mathcal{R} \). Note that the randomization function is given by \( \text{R}(x, x', \rho, \rho') := [\rho \cdot \rho' \mod N] \). A key distribution protocol \( \Pi_{\text{KD}} \) and a threshold decryption protocol \( \Pi_{\text{DEC}} \), 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.

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 \( x \in M^n \) of fixed length \( n \in \mathbb{N} \) in a single value \( c \in C \), and restrict ourselves to the case where \( M = \mathbb{Z}_m \) 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 \( \text{COM} := (\text{Gen}, \text{Com}, \text{Open}) \), and is parametrized by a message ring family \( (M_\lambda, +, \cdot)_\lambda \), a randomness space family \( (R_\lambda)_\lambda \), a commitment group family \( (G_\lambda, *)_\lambda \) and a vector length \( n \in \mathbb{N} \).

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

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

- **Correctness:** For all \( x \in M^n \), \( r \in R \) and \( c \leftarrow \text{Com}(pp, x; r) \), it holds that \( \text{Open}(pp, c, x, r) = 1 \).
- **Hiding:** Let \( A \) be a PPT (resp. any) algorithm that receives \( 1^\lambda \) and the public parameters \( pp \), and outputs two vectors \( x_1, x_2 \in M^n \) and some arbitrary value \( s \in \{0, 1\}^* \). Let
  \[
  X_i(\lambda) := \left\{ (s, c_i \leftarrow \text{Com}(pp, x_i; r)) \mid pp \leftarrow \text{Gen}(1^\lambda, n, m), x_1, x_2 \in M^n, (s, r) \leftarrow A(1^\lambda, pp) \right\}
  \]
  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 \( A \) be a PPT (resp. any) algorithm that receives as input \( 1^\lambda \) and the public parameters \( pp \), and outputs two vectors \( x_1, x_2 \in M^n \) and corresponding randomness \( r_1, r_2 \in R \). Then we say that the commitment scheme is computationally (resp. statistically) binding if the following probability is bounded by some negligible function \( \text{negl} : \mathbb{N} \to \mathbb{R}_{>0} \) for all \( \lambda \in \mathbb{N} \):
  \[
  \Pr \left[ \text{Com}(pp, x_1; r_1) = \text{Com}(pp, x_2; r_2) \mid pp \leftarrow \text{Gen}(1^\lambda, n, m), (x_1, r_1, x_2, r_2) \leftarrow A(1^\lambda, pp), x_1 \neq x_2 \right]\right]
  \]
  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 \( x_1, x_2 \in M^n \), \( r_1, r_2 \in R \) and \( c_1 \leftarrow \text{Com}(pp, x_1; r_1) \), \( c_2 \leftarrow \text{Com}(pp, x_2; r_2) \), it holds that
  \[
  c_1 + c_2 = \text{Com}(pp, x_1 + x_2; R(x_1, x_2, r_1, r_2)),
  \]
  \[
  c_1^* = \text{Com}(pp, a \cdot x_1; R(x_1, a, r_1)).
  \]
- **Randomization Property:** For all \( x_1, x_2 \in M^n \) and at least one of \( r_1, r_2 \in R \) sampled uniformly, \( R(x_1, x_2, r_1, r_2) \) is distributed uniformly in \( R \).
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 \( \text{PKE} := (\text{Gen}, \text{Enc}, \text{Dec}) \) 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 \( (\text{pk}, \text{sk}) \leftarrow \text{Gen}(1^\lambda) \), setting \( \text{pp} := \text{pk} \) and defining \( c := \text{Com}(\text{pp}, x; r) := \text{Enc}(\text{pk}, x; r) \). 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 \( c = \text{Enc}(\text{pk}, x; r) \). As we will see shortly, \( \text{PKE} \) needs to satisfy the additional condition that one is able to efficiently check the validity of a public key \( \text{pk} \).

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

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 \( \mathbb{Z}_{nm} \). 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 \( \text{COM}' := (\text{Gen}', \text{Com}', \text{Open}') \) be a homomorphic single-value commitment scheme with message ring \( \mathbb{Z}_m \). Then we say that \( \text{COM}' \) satisfies the zero-opening condition if there exists an efficiently computable function \( R \) such that for any commitment \( c := \text{Com}'(\text{pp}, x; r) \) we have
\[
e^m = \text{Com}'(\text{pp}, 0; R(c)).
\]

The construction of a vector commitment scheme \( \text{COM} \) for \( \mathbb{Z}_{nm} \) from a single-value commitment scheme \( \text{COM}' \) over \( \mathbb{Z}_m \) is detailed in Figure 1.

**Remark 2.** In [ACC+22, Thm. 1] it is proven that if \( \text{COM}' \) is a perfectly hiding and computationally binding single-value commitment scheme, then the construction in Figure 1 gives a vector commitment scheme \( \text{COM} \) which is perfectly hiding and computationally 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 \( \Sigma \)-protocols with the standard properties of completeness, \( k \)-special soundness and special honest verifier zero-knowledge.
We defined special soundness only for 3-move protocols, but we also consider the generalized notion of \((k_1, \ldots, k_e)\)-special soundness for \((2\ell + 1)\)-move protocols, as defined in [ACC+22], which implies knowledge soundness.

2.4.1 Composition of protocols.

Given two interactive proofs \(\Pi_1\) and \(\Pi_2\), we denote by \(\Pi_2 \circ \Pi_1\) their composition, defined as follows. On input a statement and witness pair \((x; w)\), it consists of an execution of \(\Pi_1(x; w)\) in which the final message \(z\) sent by the prover is replaced by an execution of \(\Pi_2(x'; z)\), where the statement \(x'\) can be publicly computed from the transcript of \(\Pi_1\).

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 \(E \subseteq \mathbb{Z}_m\). We say that \(E\) is exceptional if \(e - e'\) is an invertible element of \(\mathbb{Z}_m\) for all distinct \(e, e' \in E\). 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, 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 \(e, e' \in \mathbb{Z}_N\) 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
commitment scheme for Figure 1 is a computationally hiding and computationally binding homomorphic vector
Theorem 1. When based on a homomorphic single-value commitment scheme
where
is computationally hiding,
the randomization property since

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

\[ \text{Enc}(pk, 0; [C \mod N]) = C^N \mod N^2. \]

**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 \( \mathbb{Z}_m^* \).

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

- For \( i = 1, \ldots, n \), sample \( a_i \in \mathbb{Z}_m \) and \( r_i \in \mathcal{R} \), and put \( g_i := \text{Com}'(\text{pp}', a_i, r_i) \). Put
  \[ \text{pp} := (\text{pp}', g_1, \ldots, g_n). \]
- Query \( (x^0, x^1, s) \leftarrow A(1^\lambda, \text{pp}) \) 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 \( c_b \leftarrow \text{Com}'(\text{pp}', x^b; r') \) for \( b \in \{0, 1\}, r' \in \mathcal{R} \)).
- Query \( b' \leftarrow A(s, c_b) \) and output \( b' \).

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

\[ \text{Com}'(\text{pp}', x^b; r') = \prod_{i=1}^n \text{Com}'(\text{pp}', a_i; r_i)x_i^b \cdot \text{Com}'(\text{pp}', 0; r) \]

\[ = \text{Com}(\text{pp}, x^b; r), \text{ for } r := R(x^b, -x^b, r', \tilde{r}), \]

where \( \tilde{r} \) is the randomness of \( \prod_{i=1}^n \text{Com}'(\text{pp}', a_i; r_i)^{-x_i^b} \) 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, \( A' \) succeeds with at most negligible probability and COM is computationally hiding.

Now consider an adversary \( A \) against the binding property of COM, then we can use it to construct an adversary \( A' \) against the hiding property of COM', on input \( 1^\lambda \), pp' \( \leftarrow \text{Gen}'(1^\lambda, m) \), as follows:
Remark 14 Communication-Efficient Multi-Party Computation for RMS Programs

The idea behind the reduction is the following. If the adversary \( A \) successfully outputs \((x, r, x', r')\) with \( \text{COM}(\text{pp}, x; r') = \text{COM}(\text{pp}, x'; r') \), \( x \neq x' \), then perfect binding of \( \text{COM'} \) implies that \( \sum a_i x_i \equiv \sum a_i x'_i \equiv \mod m \). With probability \( \geq 1/n \) we have \( x_i \neq x'_i \equiv \mod m \). If \( a_i \neq a'_i \equiv \mod m \), output the bit
\[
b := \langle a_i \rangle \equiv -\langle x_i - x'_i \rangle^{-1} \cdot \sum_{i \neq i'} a_i (x_i - x'_i) \mod M.
\]

The homomorphic properties of \( \text{COM} \) follow from the homomorphic and zero-opening properties of \( \text{COM'} \) 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 \( \text{Gen}(1^\lambda, M, m) \) of \( \text{COM} \) 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 \( \text{pp} := \text{pk} \) are already generated as part of the key distribution protocol \( \Pi_{KD} \), and knowledge of \( \leq n - t \) secret key-shares \( \text{sk}_i \) does not break the hiding property by the threshold security of AHE. Furthermore, the basis elements \( g_i := \text{Enc}(\text{pk}, a_i; r_i) \) need to be generated with \( a_i \equiv Z_m, r_i \equiv \mathcal{R} \) not known to any of the parties, since otherwise this would break the binding property of \( \text{COM} \). This can be realized by letting each party \( P_i \) sample \( a_{i,j} \equiv Z_m, r_{i,j} \equiv \mathcal{R} \) and publish \( g_{i,j} := \text{Enc}(\text{pk}, a_{i,j}, r_{i,j}) \) 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 \( g_i := g_{i,1} \cdots \cdot g_{i,n} \). 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 \( Z_{N^2} \) has negligible statistical distance to sampling uniformly random elements in \( Z_{N^2} \).

4 Standard \( \Sigma \)-Protocols

In this section we will describe a standard \( \Sigma \)-protocol for proving knowledge of a pre-image of a “homomorphism” \( \Psi_n \) which takes as inputs a vector \( \mathbf{x} \) of length \( n \in \mathbb{N} \) over a ring \( Z_m \) and some “randomness” \( \gamma \) from a finite set \( S \), and outputs an element \( \Psi_n(\mathbf{x}; \gamma) \) of a group
Figure 2 Standard Σ-protocol Π_Σ for relation \( \mathcal{R}_\Psi \)

<table>
<thead>
<tr>
<th>Prover</th>
<th>Verifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a \xleftarrow{} \mathbb{Z}_m^* ), ( \rho \xleftarrow{} S )</td>
<td></td>
</tr>
<tr>
<td>( A := \Psi_n(a; \rho) )</td>
<td></td>
</tr>
<tr>
<td>( e \xleftarrow{} \mathcal{E} )</td>
<td></td>
</tr>
<tr>
<td>( z := a + e \cdot x )</td>
<td></td>
</tr>
<tr>
<td>( \sigma := R(a, e \cdot x, \rho, R(x, e, \gamma)) )</td>
<td></td>
</tr>
<tr>
<td>( \Psi_R(z; \sigma) \xleftarrow{} A + C^e )</td>
<td></td>
</tr>
</tbody>
</table>

(\( \mathbb{H}, \ast \)). The map \( \Psi_n : \mathbb{Z}_m^n \times S \rightarrow \mathbb{H} \) is not a homomorphism in the mathematical sense since we do not assume a group structure on the set \( S \), 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 \( S \), such that for all \( x, x' \in \mathbb{Z}_m^n \), all \( \gamma, \gamma' \in \mathbb{S} \), all \( a \in \mathbb{Z} \) and all \( C \in \mathbb{H} \):

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

We show that when these conditions are satisfied, there exists a standard zero-knowledge \( \Sigma \)-protocol for proving knowledge of a pre-image \( (x, \gamma) \in \mathbb{Z}_m^n \times S \) such that \( \Psi_n(x; \gamma) = C \) for some public element \( C \in \mathbb{H} \). In Section 4.1 we give examples of morphisms \( \Psi_n \) 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 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 \( \Pi_\Sigma \) from Figure 2 is a complete, special honest verifier zero-knowledge, 2-special sound protocol for the relation \( \mathcal{R}_\Psi := \{(C, \Psi_n; x, \gamma) : C = \Psi_n(x; \gamma)\} \).

**Proof.** Completeness: Follows directly from the homomorphic properties of \( \Psi_n \).

**SHVZK:** We can construct a simulator on input public parameters \( m, n, S, \mathbb{H} \) and a statement \( C, \Psi_n \) as follows. Sample \( e \xleftarrow{} \mathcal{E}, \sigma \xleftarrow{} S \) and \( z \xleftarrow{} \mathbb{Z}_m^* \), put \( A := \Psi_n(z; \sigma) \ast C^{-e} \), and output the transcript \( (A, e, z, \sigma) \). 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 \( (A, e, z, \gamma), (A', e', z', \gamma') \) be two accepting transcripts with \( e \neq e' \) for the same statement \( C, \Psi_n \) and with respect to the same public parameters
16 Communication-Efficient Multi-Party Computation for RMS Programs

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

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 \( \Psi_n \) we can write

\[ C = \Psi_n(z - z'; \hat{\sigma})^\beta \cdot \Psi_n(0; R(C))^{-k} \]
\[ = \Psi_n(\beta \cdot (z - z'); R(\beta(z - z'), 0, R(z - z', \beta, \hat{\sigma}), R(0, -k, R(C)))) \]

So we conclude that we can efficiently extract a witness for the statement \( C, \Psi_n \).

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 \( \Psi_1(x; \gamma) := \text{Enc}(pk, x; \gamma) \).

Example 5 (Proof of Commitment Opening). Let \( \text{COM} \) be a homomorphic vector commitment scheme over \( \mathbb{Z}_m^n \) constructed from a homomorphic single value commitment scheme \( \text{COM}' \) over \( \mathbb{Z}_m \) satisfying the zero-opening condition of Definition 3. Then \( \text{COM} \) satisfies an analogous zero-opening condition. That is, let \( pp = (pp', g_1, \ldots, g_n) \) and \( C := \text{Com}(pp, x; \gamma) = \text{Com}(pp', 0; \gamma') \ast g_1^{q_1} \ast \cdots \ast g_n^{q_n} \). Then by construction \( C \) is a valid commitment under \( \text{COM}' \) and hence we have

\[ C^m = \text{Com}(pp', 0; R'(C)) = \text{Com}(pp, 0; R(C)), \quad R(C) := R'(C). \]

Hence by letting \( \Psi_n(x; \gamma) := \text{Com}(pp, x; \gamma) \), 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 \( x \) satisfies some linear form constraints.

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

\[ \Psi_n(x; \gamma, \rho) := (\text{Com}(pp, x; \gamma), C^x \ast \text{Enc}(pk, 0, \rho)) \]

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

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

\[ \Psi_n(x; \gamma, \rho) \ast \Psi_n(x'; \gamma', \rho') = (\text{Com}(pp, z; \hat{\gamma}), C^{x+x'} \ast \text{Enc}(pk, 0; \hat{\rho})) = (\ast) \]

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

\[ (\ast) = (\text{Com}(pp, z; \hat{\gamma}), C^x \ast \text{Enc}(pk, 0; \hat{\rho})) = \Psi_n(z; \hat{\gamma}, \hat{\rho}). \]
Figure 3: Amortized $\Sigma$-protocol $\Pi_{\text{am-x}}$ for relation $\mathcal{R}_{\text{multi-x}}$

<table>
<thead>
<tr>
<th>Prover</th>
<th>Verifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbf{a} \leftarrow \mathbb{Z}_m^n, \hat{\mathbf{a}} \leftarrow \mathcal{S}$</td>
<td>$\mathbf{A} := \Psi_n(\mathbf{a}; \hat{\mathbf{a}})$</td>
</tr>
<tr>
<td>$\mathbf{A} := \Psi_n(\mathbf{a}; \hat{\mathbf{a}})$</td>
<td>$\mathbf{A} \rightarrow \varepsilon$</td>
</tr>
<tr>
<td>$\mathbf{z} := \mathbf{a} + \sum_{i=1}^{k} e^i \cdot x_i$</td>
<td>$\mathbf{z}, \sigma$</td>
</tr>
<tr>
<td>$\sigma := R(\mathbf{a}, \sum_{i=1}^{k} e^i \cdot x_i; \mathbf{a}, \mathbf{e}, \mathbf{\gamma}_k)$</td>
<td>$\Psi_n(\mathbf{z}; \sigma) \equiv A \cdot \prod_{i=1}^{k} C_i e_i$</td>
</tr>
</tbody>
</table>

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

$$\Psi_n(\mathbf{x}; \mathbf{\gamma}, \rho) = (\text{Com}(\mathbf{pp}, \mathbf{z}; \hat{\mathbf{z}}), C^\sigma \cdot \text{Enc}(\mathbf{pk}, 0; \hat{\mathbf{a}})) = \Psi_n(\mathbf{z}; \hat{\mathbf{a}}, \hat{\mathbf{a}}),$$

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

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

4.2 Amortizing over Pre-Images

We can apply standard techniques from $\Sigma$-protocol theory to amortize the costs of proving knowledge of many pre-images of the same homomorphism $\Psi_n$. 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 $\phi_i$ for the efficiently computable value defined recursively by $\phi_1 := R(\mathbf{x}_1, \mathbf{e}, \mathbf{\gamma}_1)$ and for $j \geq 2$:

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

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

$$\mathcal{R}_{\text{multi-x}} := \{(C_1, \ldots, C_k, \Psi_n; \mathbf{x}_1, \ldots, \mathbf{x}_k, \mathbf{\gamma}_1, \ldots, \mathbf{\gamma}_k) : C_i = \Psi_n(\mathbf{x}_i; \mathbf{\gamma}_i), \forall i \in [k]\}.$$

Proof. Completeness: Follows directly from repeatedly applying the homomorphic properties of $\Psi_n$.

SHVZK: We can construct a simulator on input public parameters $m, n, k, \mathcal{S}, \mathcal{H}$ and a statement $C_1, \ldots, C_k, \Psi_n$ as follows. Sample $e \leftarrow \mathbb{E}, \sigma \leftarrow \mathcal{S}$ and $\mathbf{z} \leftarrow \mathbb{Z}_m^n$, put $A := \Psi_n(\mathbf{z}; \sigma) \cdot \prod_{i=1}^{k} C_i e_i$ and output the transcript $(A, e, \mathbf{z}, \sigma)$. 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 \((A, e_1, z_1, \sigma_1), \ldots, (A, e_{k+1}, z_{k+1}, \sigma_{k+1})\) be \(k + 1\) accepting transcripts for the same statement \(C_1, \ldots, C_k, \Psi_n\), with respect to the same public parameters \(m, n, k, S, H\) 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 \(\mathbb{Z}_m\)

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

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 \(b_j, d_j \in \mathbb{Z}^{k+1}\) such that the \(j\)-th unit vector \(1_j\) can be written as \(1_j = Vb_j + m \cdot d_j\) over the integers. Recall that the accepting condition gives \(\Psi_n(z_j; \sigma_j) = A \cdot \prod_{i=1}^k C_i^{e_{j,i}}\) for all \(j \in [k + 1]\). Thus for each \(v \in [k]\) we can write

\[
C_v = \prod_{j=1}^{k+1} \left( \left( A^{b_{v+1,j}} \prod_{i=1}^k C_i^{e_{j,i}} \right) \right) \cdot (A^{d_{v+1,1}} \prod_{i=1}^k C_i^{e_{v+1,i+1}})^m
\]

\[
= \prod_{j=1}^{k+1} \Psi_n(z_j; \sigma_j)^{b_{v+1,j}} \cdot \Psi_n(0; R(A^{d_{v+1,1}} \prod_{i=1}^k C_i^{e_{v+1,i+1}}))
\]

\[
= \Psi_n(\bar{x}_v; \bar{\gamma}_v),
\]

where \(\bar{x}_v := \sum_{j=1}^{k+1} b_{v+1,j} \cdot z_j\) and \(\bar{\gamma}_v\) can be computed efficiently from the values \(z_j, \sigma_j, b_{v+1,j}, d_{v+1,j}, e_j, A, C_i\), for \(j \in [k + 1], \ i \in [k]\) using the homomorphic properties of \(\Psi_n\). Hence for each \(v \in [k]\) we can efficiently extract a witness \((\bar{x}_v, \bar{\gamma}_v)\) such that \(C_v = \Psi_n(\bar{x}_v; \bar{\gamma}_v)\).

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

\[
\mathcal{R}_{k\text{-mult}} := \{(P, B_1, \ldots, B_k, C_1, \ldots, C_k; \bar{x}, r, \gamma_1, \ldots, \gamma_k) : P = \text{Com}(pp; x; r) \land B_i = C_i^{e_i} \cdot \text{Enc}(pk, 0; \gamma_i), \forall i \in [k]\}
\]

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

\[
\mathcal{R}_\Psi := \{(C, \Psi_{n,e}; x, r, \gamma) : C = \Psi_{n,e}(x; r, \gamma)\}, \quad C := (P, \prod_{i=1}^k B_i^{e_i}),
\]

\[
\Psi_{n,e}(x; r, \gamma) := (\text{Com}(pp; x; r) \cdot \prod_{i=1}^k C_i^{e_i} \cdot \text{Enc}(pk, 0; \gamma)),
\]
Figure 4 Argument of knowledge $\Pi_{\text{am-mult}}$ for relation $\mathcal{R}_{\text{k-mult}}$.

Reduction from $\mathcal{R}_{\text{k-mult}}$ to $\mathcal{R}_\Psi$.

Parameters: $m, n, k, \text{pp, } S, \text{H}$

Input($P, B_1, \ldots, B_k, C_1, \ldots, C_k$; $x, r, \gamma_1, \ldots, \gamma_k$)

$P = \text{Com(pp; } x; r)$

$B_i = C_i^x * \text{Enc(pk; } 0; \gamma_i)$,

for all $i \in [k]$

\[ \begin{array}{ccc}
\text{Prover} & \downarrow & \text{Verifier} \\
\phi_k := \gamma & \downarrow & e \phi \notin \mathcal{E}
\end{array} \]

\[ \begin{aligned}
\text{Com(pp; } x; r) & \rightarrow P, \\
\prod_{i=1}^{k} C_i^{x} \cdot \text{Enc(pk; } 0; \gamma) & \equiv \\
\prod_{i=1}^{k} B_i^{c_i}
\end{aligned} \]

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

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

where $R$ is the randomization function of AHE.

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

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

Special soundness: Consider the $k + 1$ accepting transcripts $(c_1, x_1, r_1, \gamma_1), \ldots, (c_{k+1}, x_{k+1}, r_{k+1}, \gamma_{k+1})$ for the same statement $P, B_1, \ldots, B_k, C_1, \ldots, C_k$, with respect to the same public parameters $m, n, k, \text{pp, } S, \text{H}$ 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 $\text{Com(pp; } x_i; r_i) = P$ for all $i \in [k]$. So either we can break the binding property of $\text{COM}$ or $x_1 = x_2 = \ldots = x_k = x$.

Again consider the $(k + 1) \times (k + 1)$ Vandermonde matrix $V$ as in equation (1). Since $c_j - c_i$ is invertible for all $i \neq j$, $V$ is invertible over $\mathbb{Z}_m$. Hence for each $j \in [k+1]$ there exist $b_j, d_j \in \mathbb{Z}^{k+1}$ such that the $j$-th unit vector $1_j$ can be written as $1_j = Vb_j + m \cdot d_j$ over the integers. By the second part of the verification equation, we see that for any $v \in [k]$:

$B_v = \prod_{j=1}^{k+1} \left( \prod_{i=1}^{k} B_i^{c_i(b_{i+1} + j)} \right) \left( \prod_{i=1}^{k} B_i^{d_{i+1}1_j} \right)^m$

$= \prod_{j=1}^{k+1} \left( \prod_{i=1}^{k} C_i^{x} \cdot \text{Enc(pk; } 0; \gamma_j) \right)^{b_{i+1}1_j} \cdot \prod_{i=1}^{k} B_i^{d_{i+1}1_j}$

$= C_v^{x} \cdot \left( \prod_{j=1}^{k+1} (B_i \ast C_i^{-x})^{d_{i+1}1_j} \right)^m \ast \prod_{j=1}^{k+1} \text{Enc(pk; } 0; \gamma_j)^{b_{i+1}1_j}$

$= C_v^{x} \cdot \text{Enc(pk; } 0; \gamma_v)$,
Figure 5 Compression Mechanism $\Pi_C$ for relation $\mathcal{R}_\Psi$

<table>
<thead>
<tr>
<th>Parameters: $m, n, S, \mathbb{H}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input($C, \Psi_n; x, \gamma$)</td>
</tr>
<tr>
<td>$C = \Psi_n(x, \gamma)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Prover</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\rho_A, \rho_B \leftarrow S$</td>
</tr>
<tr>
<td>$A := \Psi_n((0, x_L); \rho_A)$</td>
</tr>
<tr>
<td>$B := \Psi_n((x_R, 0); \rho_B)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Verifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\rho_A, \rho_B \leftarrow S$</td>
</tr>
<tr>
<td>$A, B \rightarrow$</td>
</tr>
<tr>
<td>$\phi(x, \gamma, \rho_A, \rho_B, c)$</td>
</tr>
</tbody>
</table>

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

Post-composing the reduction $\Pi_{\text{am-mult}}$ with the standard protocol $\Pi_C$ from Figure 2 or the compressed protocol $\Pi$ 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 $\Sigma$-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 $x \in \mathbb{Z}_m^n$, we denote by $x_L$ and $x_R$ the left and right halves of $x$, i.e. $x = (x_L, x_R) \in \mathbb{Z}_m^{n/2} \times \mathbb{Z}_m^{n/2}$. We also use as shorthand the efficiently computable function $\phi$ defined as follows:

$$\phi(x, \gamma, \rho_A, \rho_B, c) := R((c x_L, x_L + c x_R), (c^2 x_R, 0), \alpha, \beta),$$

where $\alpha := R((0, x_L), c x, \rho_A, R(x, e, \gamma)), \beta := R((x_L, 0), c^2, \rho_B)$.

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

Proof. Completeness: Follows from the definition of $\phi$ and the homomorphic properties of $\Psi_n$.

Special soundness: Let $(A, B, e_1, z_1, \sigma_1), (A, B, e_2, z_2, \sigma_2)$ and $(A, B, e_3, z_3, \sigma_3)$ be three accepting transcripts for the statement $(C, \Psi_n)$, 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 \(\Psi_n((e_i z_i, z_i); \sigma_i) = AC^n B^n\) for \(i = 1, 2, 3\). Then

\[
C = A^{a_1 + a_2 + a_3} C^{e_1 a_1 + e_2 a_2 + e_3 a_3} B^{k_1 + k_2 + k_3} (A^{k_1} C^{k_2} B^{k_3})^m
\]

\[
= \Psi_n((e_1 z_1, z_1); \sigma_1)^{a_1} \Psi_n((e_2 z_2, z_2); \sigma_2)^{a_2} \Psi_n((e_3 z_3, z_3); \sigma_3)^{a_3} \Psi_n(0; R(A^{k_1} C^{k_2} B^{k_3}))
\]

\[
= \Psi_n(\bar{x}; \bar{\gamma}),
\]

where \(\bar{x} = \sum_{i=1}^{3} a_i (e_i z_i, z_i)\) and \(\bar{\gamma}\) can be computed from the values \(a_i, k_i, e_i, C, A, B\). This shows we can compute a witness \((\bar{x}, \bar{\gamma})\) for the statement \((C, \Psi_n)\) and relation \(R\).

We show how the compression mechanism can be composed with the basic \(\Sigma\)-protocol to give a zero-knowledge proof of a pre-image of the map \(\Psi_n\), with communication complexity logarithmic in \(n\). Moreover, we require only a logarithmic number of “blinding” elements (consisting of the variables \(a, \rho_A, \rho_B\)). Compared to the naive approach, which uses a blinding vector \(a\) of length \(n\), this significantly decreases the computation costs of the protocol when the input vector \(x\) is sparse.

Consider the protocol \(\Pi'_\Sigma\) obtained from the basic protocol \(\Pi_\Sigma\) of Figure 2 by replacing the blinding vector \(a' \in \mathbb{Z}_m^n\) sampled by the prover by \(a := (a, 0, \ldots, 0)\), where \(a \neq \mathbb{Z}_m\). Clearly \(\Pi'_\Sigma\) still satisfies completeness and 2-special soundness, although it is no longer zero-knowledge. The compressed protocol is obtained as the composition of this modified basic protocol and the compression mechanism: \(\Pi = \Pi_\Sigma \circ \cdots \circ \Pi_\Sigma \circ \Pi'_\Sigma\). A full description is given in Figure 6, where we use the maps \(\Psi_i\) defined recursively from \(\Psi_n = \Psi_{2^l}\) as follows for \(i \in \{1, 2, \ldots, 2^{l-1}\}\):

\[
\Psi_i(y; \rho) := \Psi_{2^{l}}((e_{l-\log_2 i} y; y); \rho).
\]

In addition to the homomorphic properties defined in Section 4, the morphism \(\Psi_n\) 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 \(\gamma \in \mathcal{S}\), the distributions of \(\Psi_n(x; \gamma)\) and \(\Psi_n(0; \gamma)\) 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 \(C_1, \ldots, C_n\) are valid ciphertexts.

**Theorem 6.** *The compressed protocol \(\Pi\) is a complete, \((2, 3, \ldots, 3)\)-special-sound and computational special honest-verifier zero-knowledge protocol for \(R\).*

**Proof.** **Completeness:** Follows from the completeness of \(\Pi'_\Sigma\) and \(\Pi_\Sigma\).

**Special soundness:** Follows from the 2-special-soundness of \(\Pi'_\Sigma\) and 3-special-soundness of \(\Pi_\Sigma\). An extraction algorithm for \(\Pi\) can be obtained by combining the extraction procedures of \(\Pi_\Sigma\) and \(\Pi'_\Sigma\).
Figure 6 Compressed $\Sigma$-protocol II for relation $\mathcal{A}_\Psi$

**Parameters:**

$m, n = 2^f, S, \mathbb{H}$

**Input** $(C, \Psi_n; x, \gamma)$

$C = \Psi_n(x; \gamma)$

### Prover

- $a \xleftarrow{\$} \mathbb{Z}_m, \rho \xleftarrow{\$} S$
- $a := (a_0, \ldots, 0) \in \mathbb{Z}_m^n$
- $A_0 := \Psi_n(a; \rho)$

\[ x^1 := (x^1_L, x^1_R) := a + e_1 x \]

- $\gamma_1 := R(a, e_1 x, \rho, R(x, e_1, \gamma))$
- $\rho^A_1, \rho^B_1 \xleftarrow{\$} S$
- $A_1 := \Psi_n((0, x^1_L); \rho^A_1)$
- $B_1 := \Psi_n((x^1_R, 0); \rho^B_1)$

\[ x^2 := x^1_L + e_2 x^1_R \]

- $\gamma_2 := \phi(x^2, \gamma_1, \rho^A_1, \rho^B_1, e_2)$

\[ \vdots \]

\[ \vdots \]

- $\rho^A_\ell, \rho^B_\ell \xleftarrow{\$} S$
- $A_\ell := \Psi_n((0, x^\ell_L); \rho^A_\ell)$
- $B_\ell := \Psi_n((x^\ell_R, 0); \rho^B_\ell)$

\[ z := x^\ell_L + e_{\ell+1} x^\ell_R \]

- $\sigma := \phi(x^\ell, \gamma_\ell, \rho^A_\ell, \rho^B_\ell, e_{\ell+1})$

### Verifier

\[ C_0 := A_0 \]

\[ C_1 := A_0 \]

\[ C_2 := A_1 \]

\[ C_\ell := A_\ell \]

\[ \Psi_1(z; \sigma) \xRightarrow{\$} C_{\ell+1} \]

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

Let us now check that the distribution of the simulated transcript is computationally indistinguishable from that of an honest transcript. First, the simulated challenges $e_1, \ldots, e_{\ell+1}$ are sampled as in the real protocol and the prover’s messages $A_1, \ldots, A_{\ell-1}, B_1, \ldots, B_\ell$ are all of the form $\Psi_n(0; \delta)$ for freshly sampled $\delta$, which are computationally indistinguishable from $\Psi_n(y; \delta)$ for any $y$ by the indistinguishability property of $\Psi_n$. Observe that in the real protocol $z$ is uniform (even when conditioned on the aforementioned elements of the transcript), due to $a \xleftarrow{\$} \mathbb{Z}_m$ sampled by the prover. The first message $A_0$ is indistinguishable in the real and simulated transcripts, since $\rho$ is independent from the other variables discussed so far. Conditioned on the previous transcript elements, $\sigma$ is uniform in the real protocol since $\rho_A$ is uniform. Finally, in both the real and simulated
Remark 5. The communication costs of $\Pi$ 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 $\Pi$ is computationally SHVZK based on the assumption that $\Psi_n(x; \rho) \approx\!\approx \Psi_n(0; \rho)$. However, if the map $\Psi_n$ has the stronger property that $\Psi_n(x; \rho)$ is uniformly distributed in $\mathcal{H}$ (for uniform $\rho$), then the real and simulated transcripts are identically distributed and $\Pi$ 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 $j \in [l]$. 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 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:

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

Low Computation Mode (mode = 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:

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

Low Communication Mode (mode = 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:

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

transcript $A_{t} \in \mathcal{H}$ is the unique element which satisfies the verification equation. \hfill \blacksquare
Parameters: Parties $P_1, \ldots, P_n$. Public modulus $m \in \mathbb{N}$, maximum local input vector length $M$, maximum local input vectors $\nu \bydef \max_i(n_i)$.

- Inputs: Party $P_i$ holds local input vectors $\{x_{i,k}\}_{k \in [n]} \subset \mathbb{Z}_m^M$, global inputs values $\{y_{i,k}\}_{k \in [n]} \subset \mathbb{Z}_m$.

- Program: Vector-RMS program $f = \{f_{t,j}\}_{t \in [T], j \in [\ell]}$ with $T$ layers, where layer $T$ consists of $\ell$ gates. Gates are of the form $f_{t,j} := (i, k), O_i$, where $i \in [n], k \in [\nu_i]$ specifies the local input vector $x_{i,k}$ 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 $i_n : [n] \to [\ell]$ and sets $\text{Out}_i \subset [\ell]$ for each $i \in [n]$.

- Mode: An option mode $\in \{\text{low.comm}, \text{low.comp}\}$

**Setup.** Each party $P_i$ sends $(\text{sid}, i_t, m)$ to $\mathcal{F}_{KD}$ and receives $(pk, sk_i)$.

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

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

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

- **Load inputs in memory:**
  - For each $k \in [\nu_i]$, party $P_i$ samples $\rho_k \bydef \mathcal{R}$ and broadcasts $C_{1,1} := \text{Enc}(pk, y_{i,k}; \rho_k)$ together with $\pi_{i_t} := \Pi_{pk}(C_{1,1}; y_{i,1}, \rho)$.
  - If mode = low.comp, then for each $k \in [\nu_i]$, party $P_i$ samples $\gamma_k \bydef \mathcal{R}$ and broadcasts $A_{i,1} := \text{Com}(pp, x_{i,1}; \gamma)$.
  - If mode = low.comm, then party $P_i$ samples $\gamma \bydef \mathcal{R}$ and broadcasts $A_{i,1} := \text{Com}(pp, x_{i,1}; \gamma)$, where $x_{i,1} := (x_{i,1,1}, \ldots, x_{i,1,n})$.
  - Party $P_i$ verifies the proofs $\pi_{i_t}$ for all $j \neq i$, and aborts if any of them do not accept.

- **Evaluate layers ($t = 2, \ldots, T$):**
  - For gate $f_{t,j} = (i(k), O)$, party $P_i$ samples $\rho_{t,j} \bydef \mathcal{R}$ and broadcasts $C_{t,j} := C_{t,j}^\nu \cdot \text{Enc}(pk, 0; \rho_{t,j})$, where $C_{t,j} = (C_{t,j}(u,v))_{(u,v) \in O}$.
  - If mode = low.comp, $P_i$ broadcasts $\pi_{t,1,0} := \Pi_{\text{low.comp}}(A_{i,1}, C_{t,j})_{(t,j) \in I_{t,0}}, C_{t,j}(x_{i,k}; y_{i,k}, (k, j))_{(t,j) \in I_{t,0}}$, where $I_{t,0} \subseteq [\nu] \times [\ell]$ is the set of indices $(k, j)$ of local input vectors $x_{i,k}$ for which $f_{t,j} = ((i, k), O)$ 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 mode = low.comm, $P_i$ broadcasts $\pi_{t,1} := \Pi_{\text{low.comm}}(A_{i,1}, (C_{t,j})_{(t,j) \in I_{t,1}}, x_{i,1}, \gamma_{i,1}, (\rho_{t,j})_{(t,j) \in I_{t,1}})$, where $I \subseteq [T] \times [\ell]$ is the set of indices $(t, j)$ for which $f_{t,j} = ((i, k), O)$ for some $k \in [\nu_i]$ and $O \in ([t-1] \times [\ell])^M$, and $C_{t,j} := (0, \ldots, C_{t,ij}, \ldots, 0)$ is the vector of length $\nu \cdot M$ with $C_{t,j}$ at the $k$-th entry and zeros at all other entries. (i.e., such that $C_{t,j}^\nu = C_{t,j}^\nu$.
  - 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 $j \in \text{Out}_i$, the value $C_{T,j}$ will be decrypted to party $P_i$ as follows:
  - Party $P_i$ samples $r_j \bydef \mathcal{Z}_m$ and $\rho_j \bydef \mathcal{R}$, and broadcasts $C_{\text{Out},j} := \text{Enc}(pk, r_j; \rho_j)$ together with $\pi_{\text{Out}}, \leftarrow \Pi_{pk}(C_{\text{Out},j}; r_j, \rho)$.
  - If the proof verifies, for $j \in \text{Out}_i$, each party sends $(\text{sid}, i_t, (pk, sk_j), C_{T,j} \ast C_T)$ to $\mathcal{F}_{\text{DEC}}$ and receives $\tilde{z}_{T,j} \in \mathbb{Z}_m$.
  - Party $P_i$ computes its output value $z_{T,j} := \left[\tilde{z}_{T,j} - r \mod m\right]$ for $j \in \text{Out}_i$. 

---

**Figure 7** Protocol $\Pi_{\text{RMS}}$ for Maliciously Secure Computation of vector-RMS program.
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 \( a_{i,k} \in \mathbb{Z}^M \) corresponding to the rows of their local (weighted) adjacency matrix and has global input values \( x_i^k \) the initial attribute values for \( k \in [M_i] \), together with an embedding \( \phi : [M_i] \rightarrow [M] \) of the local graph in the global graph. The corresponding vector-RMS program consists of \( T \) layers with \( t = M \) gates in each layer, where each gate \( j \) is of the form \( f_{t,j} = ((i,k), d_{t-1}) \) with \( d_{t-1} = ((t-1,1), \ldots, (t-1,M)) \) 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 \( C_{t,j} := C_{d_{t-1}}^{k} \ast \text{Enc}(\text{pk}, 0, \rho_{t,j}) \) for each \( k \in [M_i] \).

- \text{mode = low.comp:} \( P_i \) publishes \( M_i \) commitments \( A_{i,k} \). Moreover, \( I_{t,d_{t-1}} = ((k, \phi(k)) : k \in [M_i]) \), 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 \( a_{i,k} \in \mathbb{Z}^M \) will largely be lost after the amortization from Figure 3, since we are proving knowledge of a witness \( \sum_{k=1}^{M_i} e^k \cdot a_{i,k} \) for a random challenge \( c \). It is hard to predict how much we save by using sparse blinding compared to full blinding in this case.

- \text{mode = 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 \( j = \phi([M_i]) \). 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 \( c \)):

\[
\Psi(a_i; \gamma, \rho) := (\text{Com}(\text{pp}, a_i; \gamma), \prod_{k \in [M_i], t \in [T]} C_{t,\phi(k)}^{\sum_{j=1}^{M_i} e^k \cdot a_{i,k}} \ast \text{Enc}(\text{pk}, 0; \gamma))
\]

The computational complexity of the compressed protocol scales linearly in the evaluation time of \( \Psi \), 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 \( a_i \) 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. Since the second component of \( \Psi \) 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 \( \text{AHE} \) be a threshold additively homomorphic encryption scheme with message space \( \mathbb{Z}_m \) with threshold \( \tau \) as defined in Section 2.2 and which satisfies the zero-opening condition of Definition 3. Furthermore, let \( \mathcal{F}_{\text{KD}} \) be an ideal key distribution functionality which takes input \( (\text{sid}, i, d, m) \) from party \( P_i \) for each \( i \in [n] \), and outputs \( (\text{pk}, \text{sk}_i) \) to party \( P_i \) for \( i \in [n] \). Let \( \mathcal{F}_{\text{DEC}} \) be an ideal distributed decryption functionality, which takes as input \( (\text{sid}, \text{pk}, \text{sk}_i, C) \) from at least \( \tau \) honest parties \( P_i \) and outputs the set \( \{\text{Dec}(\text{pk}, \text{sk}_1, \ldots, \text{sk}_n, c) : c \in C\} \) to all parties. Finally, let \( \mathcal{F}_{\text{COM-Gen}} \) be an ideal functionality that takes as input \( (\text{sid}, \text{pp}, M, m) \) from each party \( P_i \), where \( \text{pp} \) are public parameters of a single-value commitment scheme \( \text{COM}' \) over \( \mathbb{Z}_m \), and outputs \( \text{pp} := (\text{pp}', g_1, \ldots, g_M) \) 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 \( \mathcal{F}_{\text{RMS}} \) 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 \( \Pi_{\text{RMS}} \) from Figure 7 realizes the functionality \( \mathcal{F}_{\text{RMS}} \) against at most \( n - \tau \) maliciously corrupted parties with identifiable abort in the random oracle, \( \mathcal{F}_{\text{KD}}, \mathcal{F}_{\text{DEC}}, \mathcal{F}_{\text{COM-Gen}} \)-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 $A$ maliciously corrupts an arbitrary subset $c \subset \{1, \ldots, n\}$ of at most $n - \tau$ parties $(P_i)_{i \in c}$. The simulator $Sim$ interacts with the corrupted parties $(P_i)_{i \in c}$ and the ideal functionality $F_{RMS}$ on behalf of $(P_i)_{i \in c}$, takes the role of $F_{KD}, F_{DEC}, F_{COM-Gen}$ and simulates the random oracle, as follows.

- To handle the setup, the simulator can run $Gen(1^\lambda) \rightarrow (pk, sk_1, \ldots, sk_n)$ and distribute the secret key shares $sk_i$ to the malicious parties $i \in c$. It can subsequently simulate $F_{COM-Gen}$ 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 $F_{RMS}$ and receives their outputs $z_{T,j}$ for $j \in \text{Out}_i$ for each $i \in c$.
- At the final layer, upon receiving the ciphertexts $C_T$ 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 $C_{T,j} \ast C_T$ for a $j \in \text{Out}_i$ for some $i \in c$ and the corresponding ciphertext $C_T$ the simulator received at the precious step, and if the correct $pk, sk_i$ are provided, and aborts if this is not the case. Otherwise, the simulator returns $\tilde{z}_{T,j} := [z_{T,j} + r \mod m]$ to all of the parties.

\[ \square \]

### 7 Evaluation of Sparse Blinding Technique

Since the sparse blinding technique does not yield an asymptotic improvement, we have implemented the compressed $\Sigma$-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 $\Sigma$-protocol for proving knowledge of a pre-image for a homomorphism of the form $\Psi(x; \gamma) = (\text{Com}(x; \gamma), L(x))$ with $x \in \mathbb{Z}_n^m$ the secret input vector, $\gamma$ 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.$^1$

**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 $x \in \mathbb{Z}_n^m$ that has Hamming weight $k \leq n$, i.e., the parameter $k$ defines the sparsity of $x$. 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. 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:

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

![Figure 8: Prover complexity (in number of exponentiations) and prover runtime (in seconds), respectively, 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.](image)

**Performance.** We measure the performance of the implementation for vectors $x$ of size $|x| = 2^{14}$ 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.$^1$

$^1$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.
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 [vEDvdB]

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 in Figure 8, similar savings in runtime 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 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


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 \((x, γ)\) satisfies some linear constraint \(L(x) = y \in \mathbb{Z}_n\), where \(L : \mathbb{Z}_n \rightarrow \mathbb{Z}_n\) is a linear form, by letting \(Ψ_ε(x; γ) := (\text{Com}(pp, x; γ), L(x))\). As in the reduction in [AC20, Prot. 3], it is possible to reduce proving knowledge of a witness for the morphism \(Ψ_ε(x; γ) := (\text{Com}(pp, x; γ), L(x))\) to proving knowledge of a witness for the morphism \(Ψ_ε(x; γ) := \text{Com}(pp, (x, e \cdot L(x)); γ)\) for a random challenge \(e \leftarrow \mathcal{E}\) 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 \(Ψ_ε(x; γ) := (\text{Com}(pp, x; γ), L_1(x), \ldots, L_k(x))\) 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 \(L_ε(x) := \sum_{i=1}^k e_i \cdot L_i(x)\). 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 \(Ψ_ε(x; γ) := (\text{Com}(pp, x; γ), L(x))\) does not satisfy the indistinguishability condition as defined in Section 4 but the morphism \(Ψ_ε(x; γ) := \text{Com}(pp, (x, e \cdot L(x)); γ)\) does. Another option would be to define the morphism as \(Ψ(x; γ, ρ) := (\text{Com}(pp, x; γ), L(x) + ρ)\) with \(ρ \in \mathbb{Z}_n\) 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 \((c_1, c_2) := (g^x, h^{−x} \cdot g^s)\), where \(G = \{g\}\) 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 \(c_1^{-x} \cdot c_2\) 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.