Efficient Batch Zero-Knowledge Arguments for Low Degree Polynomials[⋆][⋆⋆]

Jonathan Bootle and Jens Groth

University College London, UK

Abstract. Bootle et al. (EUROCRYPT 2016) construct an extremely efficient zero-knowledge argument for arithmetic circuit satisfiability in the discrete logarithm setting. However, the argument does not treat relations involving commitments, and furthermore, for simple polynomial relations, the complex machinery employed is unnecessary.

In this work, we give a framework for expressing simple relations between commitments and field elements, and present a zero-knowledge argument which, by contrast with Bootle et al., is constant-round and uses fewer group operations, in the case where the polynomials in the relation have low degree. Our method also directly yields a batch protocol, which allows many copies of the same relation to be proved and verified in a single argument more efficiently with only a square-root communication overhead in the number of copies.

We instantiate our protocol with concrete polynomial relations to construct zero-knowledge arguments for membership proofs, polynomial evaluation proofs, and range proofs. Our work can be seen as a unified explanation of the underlying ideas of these protocols. In the instantiations of membership proofs and polynomial evaluation proofs, we also achieve better efficiency than the state of the art.

Keywords: Sigma-protocol, zero-knowledge argument, batch-verification, discrete logarithm assumption.

1 Introduction

Zero-knowledge proofs and arguments allow a prover to convince a verifier that a particular statement is true, without revealing anything beyond that fact. More formally, the statement is an element u from an NP-language $\mathcal{L}$, and the prover convinces the verifier that there exists a witness w to the fact that $u \in \mathcal{L}$. They are useful both in theory and in practice, as they can be used to construct

[⋆] The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 307937

[⋆⋆] ©IACR 2018. This article is the final version submitted by the authors to the IACR and to Springer-Verlag on January 8, 2018. The version published by Springer-Verlag is available at https://doi.org/10.1007/978-3-319-76581-5_19.

signature schemes, encryption schemes, anonymous credentials, and multi-party computation schemes with strong security guarantees.

Zero-knowledge arguments are computationally sound, meaning that cheating the verifier to accept when $u \notin \mathcal{L}$ reduces to breaking a computational intractability assumption. In this paper, we focus on the discrete logarithm assumption. There are many examples of zero-knowledge arguments based on the discrete logarithm assumption, for both general, NP-complete languages such as arithmetic circuit satisfiability [7], and for simpler languages such as range and membership arguments, shuffle arguments, and discrete logarithm relations.

While very efficient, arguments for general statements often make use of generic reductions and complex machinery, and fail to be as efficient as arguments specialised for a particular language.

1.1 Contributions

In this paper, we aim to bridge the gap between general and simple languages. We do this in three ways.

Framework for Low Degree Relations. We provide a framework to describe the types of languages commonly encountered. Protocols such as the 1-out-of-N membership argument of [28], and the polynomial evaluation argument of [2] prove membership in languages where the witnesses are zeroes of low-degree polynomial relations. In other words, the statement is an arithmetic circuit of low degree, and part of the witness is a satisfying assignment for the circuit. We give a general relation which allows us to recover specific protocols by instantiating with concrete polynomial relations. By separating the task of developing more efficient ways to perform the zero knowledge proof, and the task of designing better relations to describe a given language, we can explain the logic behind past optimisations of membership proofs in [28,6], and produce new optimisations for membership proofs and polynomial evaluation proofs.

Common Construction Techniques. We unify the approaches used in [28,2,6] to construct zero-knowledge proofs for membership and polynomial evaluation, which can all be viewed as employing the same construction method. The constructions of zero-knowledge arguments for low degree polynomial relations in these works proceed by masking an input variable u as $f_u = ux + u_b$, using a random challenge x and a random blinder u_b. During the proof, the polynomial or circuit from the statement is computed with f_u in place of u, so that the original relation appears in the leading x coefficient. The communication and computational complexity of the resulting arguments is determined by the degree of the polynomial relation and the number of inputs. By contrast, the complexity of general arithmetic circuit protocols is determined by the number of gates. In the case of [7], the authors embed a polynomial evaluation argument for a polynomial of degree N into a low degree polynomial with $\log N$ inputs and degree $\log N$, obtaining a protocol with $O(\log N)$ communication using 3 moves, and requiring $O(\log N)$ exponentiations in a suitably chosen cryptographic group.

On the other hand, a polynomial of degree N requires N multiplication gates to evaluate in general, so the best arithmetic circuit protocol [7] can only achieve $O(\log N)$ communication in $O(\log N)$ moves, and uses $O(N)$ group exponentiations. In particular, since the cost of computing group exponentiations is much higher than that of computing finite-field multiplications in the discrete logarithm setting, computing $O(\log N)$ group exponentiations rather than $O(N)$ leads to a significant performance advantage when considering implementation on constrained devices.

Bayer [1] gives two efficient batch proofs for multiplication and polynomial evaluation, which achieve a square-root communication overhead in the number of proofs to be batched. The key to achieving square-root overhead in [1] is to use Lagrange interpolation to embed many instances of the same relation into a single field element. This technique can be applied more generally to produce efficient batch proofs for the low-degree relations described above. Furthermore, by combining this with the polynomial commitment subprotocol in section 3, we improve the communication cost of the batched proof from $\sqrt{t}c$ to $\sqrt{tc}$, where c is the communication cost of the original non-batched proof, and t is a large number representing the number of proofs to be batched together.

Efficient Protocols for Applications. We exhibit a general protocol in our framework, and give an efficient batch protocol for proving and verifying t instances of the same relation simultaneously. We then show how to recover protocols of previous works with some optimisation. More specifically, we give new 1-out-of-N membership arguments and polynomial evaluation arguments. Our new instantiations simultaneously decrease communication costs and reduce prover and verifier computation, while retaining the conceptual clarity and simple 3-move structure of the originals. As an example, we obtain the most communication efficient Σ-protocols for membership or non-membership of a committed value in a public list, in the discrete logarithm setting. We also include an argument for range proofs, which captures the folklore method for performing range proofs and demonstrates the expressivity of our general relation. Our arguments all possess the following desirable properties:

- Perfect completeness and perfect special honest verifier zero-knowledge.
- Computational soundness based on the discrete logarithm assumption.
- Simple 3-move public coin structure.
- Common reference strings are formed from random group elements. They require no special structure.
- Prover and verifier both have efficient computation.

The discrete logarithm assumption is well-known, well-examined, and widely used in cryptography. Our protocols rely on the discrete logarithm assumption in groups with prime order p. The assumption is believed to hold in suitable subgroups of elliptic-curve groups. The best algorithms for finding discrete logarithms in such elliptic curve groups are still generic algorithms with complexity $\Omega(\sqrt{p})$. For these groups we therefore enjoy lower parameter sizes than protocols based on RSA groups that are subject to sub-exponential attacks.

The discrete logarithm assumption is also believed to hold in well-chosen multiplicative sub-groups of finite fields. Finite fields of prime order should have moduli of $\frac{\lambda^3}{\mathrm{polylog}\lambda}$ bits in order to achieve λ bits of security against the best known attacks. This makes protocols communicating large numbers of group elements highly impractical in this setting. As an improvement on previous works, in the case where $t = 1$ and we have a single relation, our protocols can be tuned so that they only require a constant number of group elements, resulting in much better efficiency when instantiated in finite fields of prime order, since the $\frac{\lambda^3}{\mathrm{polylog}\lambda}$ communication cost can then appear as a constant additive factor rather than a multiplicative one.

As a building block in our arguments, we also present an adaptation of the polynomial commitment sub-protocol appearing in [7], which allows the prover to commit to a polynomial so that the verifier can learn an evaluation of the polynomial in a secure manner.

1.2 Efficiency

Figure 1 compares the efficiency of our protocol with other works. One notable place where we improve communication efficiency over previous proofs is in our membership and polynomial evaluation proofs, which use a constant number of group elements, but have better communication efficiency regardless of whether the proofs are instantiated in elliptic curve groups or multiplicative subgroups of finite fields. Another is the polynomial evaluation argument with $O(\frac{\log N}{\log\log N})$ communication costs, which is an asymptotic improvement over the previous state-of-the-art, $O(\log N)$. Finally, our batch polynomial evaluation argument improves on [1] by putting the $\log N$ cost inside a square root.

1.3 Related Work

Zero Knowledge and Batching. There has been much work constructing efficient zero-knowledge arguments. For general statements, Kilian [34] gave the first zero-knowledge argument for circuit-satisfiability with poly-logarithmic communication complexity, but with high computational complexity. Bootle et al. [7] construct arguments with logarithmic communication complexity and linear computation costs based on the discrete logarithm assumption. Recent progress [8] yields zero-knowledge arguments with constant overhead for the prover, and square-root communication costs, though the large constants involved in the construction prevent it from being practical. For more specialised languages, such as range proofs, membership arguments, and polynomial evaluation arguments, there are numerous constructions [28,2], including some extremely simple Σ-protocols.

[1] We compare against the efficiency when [28] is instantiated using Pedersen commitments, and the prover and verifier know the openings of the list of commitments.

[2] We compare against the efficiency when [6] is instantiated using Pedersen commitments rather than Elgamal ciphertexts.

Protocol	Reference	Communication		Prover Computation		Verifier Computation	
		$\mathbb{G}$	$\mathbb{Z}_p$	$(\mathbb{G}, exp)$	$(\mathbb{Z}_p, \times)$	$(\mathbb{G}, exp)$	$(\mathbb{Z}_p, \times)$
Membership Proof	[7]	$4 \log N + 8$	$2 \log N + 7$	$12N$	$O(N)$	$4N$	$O(N)$
Membership Proof [1]	[28]	$4 \log N$	$3 \log N + 1$	$O(\log N)$	$O(N \log N)$	$O(\log N)$	$O(N)$
Membership Proof [2]	[6]	$\log N + 12$	$\frac{3}{2} \log N + 6$	$O(\log N)$	$O(N \log N)$	$O(\log N)$	$O(N)$
Membership Proof	This Work, 5.1	7	$4 \log N + 4$	$O(\frac{\log N}{\log \log N})$	$O(N \log N)$	$O(\frac{\log N}{\log \log N})$	$O(N)$
Membership Proof	This Work, 5.1	$2.7\sqrt{\log N}$ $+5$	$1.9 \log N +$ $2.7\sqrt{\log N} + 4$	$O(\frac{\log N}{\log \log N})$	$O(N \log N)$	$O(\frac{\log N}{\log \log N})$	$O(N)$
Batch Membership Proof	This Work, 5.1	$4.1\sqrt{t \log N}$	$4.1\sqrt{t \log N}$	$O(t \log tN)$	$O(tN \log tN)$	$O(\sqrt{t} \log tN)$	$O(tN)$
Polynomial Evaluation	[7]	$4 \log N + 8$	$2 \log N + 7$	$12N$	$O(N)$	$4N$	$O(N)$
Polynomial Evaluation	[2]	$4 \log N + 2$	$3 \log N + 3$	$O(\log N)$	$O(N \log N)$	$O(\log N)$	$O(N)$
Polynomial Evaluation	This Work, 5.2	7	$3 \log N + 4$	$O(\frac{\log N}{\log \log N})$	$O(N \log N)$	$O(\frac{\log N}{\log \log N})$	$O(N)$
Polynomial Evaluation	This Work, 5.2	$O(\frac{\log N}{\log \log N})$	$O(\frac{\log N}{\log \log N})$	$O(\frac{\log N}{\log \log N})$	$O(N \log N)$	$O(\frac{\log N}{\log \log N})$	$O(N)$
Batch Polynomial Evaluation	[1]	$O(\sqrt{t} \log N)$	$O(\sqrt{t} \log N)$	$O(t \log N)$	$O(tN \log N)$	$O(\sqrt{t} \log N)$	$O(tN)$
Batch Polynomial Evaluation	This Work, 5.2	$2.8\sqrt{t \log N}$	$2.8\sqrt{t \log N}$	$O(t \log tN)$	$O(tN \log tN)$	$O(\sqrt{t} \log tN)$	$O(tN)$
Range Proof	This Work, 5.3	7	$3 \log N + 4$	$O(\log N)$	$O(\log N)$	$O(\log N)$	$O(\log N)$
Range Proof	This Work, 5.3	$O(\frac{\log N}{\log \log N})$	$O(\frac{\log N}{\log \log N})$	$O(\log N)$	$O(\log N)$	$O(\log N)$	$O(\log N)$
Batch Range Proof	This Work, 5.3	$2.8\sqrt{t \log N}$	$2.8\sqrt{t \log N}$	$O(t \log N)$	$O(t \log N)$	$O(t \log N)$	$O(t \log N)$

Fig. 1. Efficiency Comparisons. N is the instance-size, t is the number of batched instances, $\mathbb{G}$ means the number of group elements transmitted, $\mathbb{Z}_p$ means the number of field elements transmitted, $(\mathbb{G}, exp)$ means the number of group exponentiations and $(\mathbb{Z}_p, \times)$ means the number of field multiplications. In the membership proofs, N is the number of items in the list that we wish to prove membership for. In the polynomial evaluation proofs, N is the degree of the polynomial. In the range proofs, N is the width of the range that we consider.

Camenisch and Stadler [15] provide a well-known symbolic notation for describing statements for zero-knowledge arguments of knowledge, and constructing protocols more easily from simple building blocks. By contrast, our general relation aims to describe languages defined by low degree polynomials and produce protocols for this case.

The idea of embedding many statements into a single polynomial using Lagrange interpolation polynomials in a challenge x originates in the quadratic arithmetic programs of Gennaro et al. [26]. It was used in the context of interactive zero-knowledge arguments by Bayer [1]. The technique was originally applied to construct a Hadamard product argument and batched polynomial evaluation argument. We show here that the same technique can be applied to our general relation. Earlier work by Gennaro et al. [25] batches Schnorr proofs using simple powers of x.

Other batch arguments in the literature use methods from [3] and multiply different instances of the proof by small exponents before compressing the proofs together. This approach may be used to trade soundness for efficiency. Our batching process proves and verifies the logical AND of many statements simultaneously. There are also batch proofs for OR statements [44], and k-out-of-N batch proofs [29]. Finally, Henry and Goldberg [29] define a notion of conciseness to characterise batch proofs.

Polynomial Commitments. Our polynomial commitment protocol is a key part of our zero-knowledge argument, and builds on the polynomial commitment protocol presented in [7]. Polynomial commitments were first introduced by Kate et al. [33], who give a construction using bilinear maps. The original construction has also been extended to the multivariate case [41,46]. Libert et al. [37] also gave a construction relying on much simpler pairing-based assumptions. Our polynomial commitment protocol gives square-root communication complexity based on the discrete logarithm assumption.

Applications. In a membership argument [11,10], a prover demonstrates that a secret committed value λ is an element of a list $\mathcal{L} = \{\lambda_0, \ldots, \lambda_{N-1}\}$, without revealing any other information about λ.

In a polynomial evaluation argument [23,10], a prover demonstrates that a secret committed value v is the evaluation of a public polynomial $h(U)$ at another secret committed value u.

In a range proof [9,38], a prover demonstrates that a secret committed value a is an element of the interval $[A; B]$.

One approach to constructing protocols for these applications is to design an arithmetic circuit which captures the desired conditions on the witness, and then apply existing zero-knowledge protocols for proving satisfiability in general circuits. There are currently several efficient arguments in the discrete logarithm setting. The methods of Cramer et al. [18] lead to arguments with communication complexity linear in the size of the circuit. The best interactive zero-knowledge protocol based on the discrete logarithm assumption for arithmetic circuits [7] yields a logarithmic communication complexity, but requires a non-constant number of rounds.

There are existing protocols for all three applications in the discrete logarithm setting that do not rely on general Circuit Satisfiability protocols. Cramer et al. [19] give techniques for composing sigma-protocols, producing proofs for AND composition, OR composition, and 1-out-of-many statements using sigma protocols for the individual statements. These techniques can be applied in a straightforward manner to produce sigma-protocols with linear communication complexity for the mentioned applications.

The goals of membership arguments are related to those of zero-knowledge sets [39]. Membership arguments allow a prover to commit to a secret value and show that it lies in a public set, without leaking information on the value. On the other hand, zero-knowledge sets allow the prover to commit to a secret set, and handle membership and non-membership queries in a verifiable manner, without leaking information on the set.

Herranz constructs attribute-based signatures [30] using what is essentially a set membership argument for multiple values. Like this work, the argument relies only on the discrete logarithm assumption, but the communication complexity is much higher; linear in the size of the set. Camenisch et al. [12] also provide set membership proofs with logarithmic communication complexity, and Fauzi et al. [22] construct constant size arguments for more complex relations between committed sets. The latter two works both rely on pairing-based assumptions.

Range arguments can be seen as a special case of membership arguments, where $\mathcal{L}$ is simply a list of consecutive integers. Many are based on the strong RSA assumption, and use Lagrange's Four-Square Theorem. Couteau et al. show that this assumption can be replaced by an RSA-variant which is much closer to the standard RSA assumption [17]. Examples are [27,38]. The work [16] gives an argument with sub-logarithmic communication complexity in the size of the list, which is comparable to the efficiency we achieve, and also relies on the hardness of the discrete logarithm problem, but uses pairings for verification.

Membership arguments also generalise arguments that a committed value lies in a linear subspace such as [31,32,35], which all make use of pairings. Peng [43] achieves a square-root complexity. Some existing protocols [2], [28] even achieve logarithmic communication complexity. Our single-value membership proof is an extension of the latter works where we reduce the number of commitments from logarithmic to constant.

Cryptographic accumulators,[4,40,13,14], can also be used to give membership proofs. The members of a set are absorbed into a constant-size accumulated value. Witnesses for set-membership can then be generated and verified using the accumulated value. Efficient instantiations of accumulators exist and often rely on the Strong RSA assumption or pairing-based assumptions. An RSA modulus has to be $\frac{\lambda^3}{\text{polylog}\lambda}$ bits to provide security against factorisation using the General Number Field Sieve. Security of pairing-based schemes with constant embedding degree scale similarly due to sub-exponential algorithms for attacking the discrete logarithm problem in the target group. Furthermore, such schemes require a trusted setup. By contrast, we only require random group elements of size $O(\lambda)$ bits for security against discrete logarithm attacks in elliptic curve groups.

Some of the schemes can be adapted to give zero-knowledge arguments for non-membership, from a variety of settings. For example, [2,43] also give non-membership arguments in the discrete logarithm setting. Accumulators that support non-membership arguments have been constructed, based on both pairing assumptions ([21]) and the strong RSA assumption ([36]).

1.4 Outline

Section 2 contains preliminary definitions needed to understand our protocols. Section 3 gives an adaptation of the polynomial commitment scheme used in [5]. Section 4 gives a general batched witness relation and efficient batched argument. Finally, Section 5 gives concrete choices of parameters to obtain zero knowledge arguments for several useful languages.

2 Preliminaries

Write $y = A(x;r)$ when the algorithm A outputs y on input x with randomness r. We write $y \leftarrow A(x)$ to mean selecting r at random and setting $y = A(x;r)$. We write $y \leftarrow S$ for sampling y uniformly at random from a set S. We define $[n]$ to be the set of integers $1, \ldots, n$.

Let $\lambda \in \mathbb{N}$ be a security parameter, usually provided to the algorithms in unary form 1^λ. We say that $f : \mathbb{N} \mapsto [0,1]$, is negligible if for every positive polynomial p, we have $f(\lambda) \leq \frac{1}{p(\lambda)}$ for $\lambda \gg 0$. We write $f(\lambda) \approx g(\lambda)$ if $|f(\lambda) - g(\lambda)|$ is negligible. We say that f is overwhelming if $f(\lambda) \approx 1$.

2.1 Assumptions

The results in this paper rely on the Discrete Logarithm Assumption. Let $\mathcal{G}$ be a probabilistic polynomial time algorithm that takes input 1^λ and outputs $gk = (\mathbb{G}, p, g)$. Here, $\mathbb{G}$ is a cyclic group of order p, which has efficient polynomial time algorithms for deciding membership and for computing group operations and inverses. The prime p has λ bits. The group is generated by the element g.

Definition 1 (Discrete Logarithm Assumption). *The discrete logarithm assumption holds relative to $\mathcal{G}$ if for all probabilistic polynomial time algorithms $\mathcal{A}$*

$$\Pr\left[gk = (\mathbb{G}, p, g) \leftarrow \mathcal{G}(1^\lambda); x \leftarrow \mathbb{Z}_p : x \leftarrow \mathcal{A}(gk, g^x)\right] \approx 0$$

2.2 Homomorphic Commitment Schemes

A commitment scheme allows a sender to commit to a secret value. Later on, the sender may open the commitment and reveal the value to another party, who can check that the value matches what was committed to. Commitment schemes should be hiding so that information about the secret value is not revealed prematurely, and binding so that the sender cannot reveal a different value to the one committed.

A non-interactive commitment scheme consists of two probabilistic polynomial time algorithms $(\mathrm{Gen}, \mathrm{Com})$. The first algorithm creates a commitment key $ck \leftarrow \mathrm{Gen}(1^\lambda)$. The key specifies a message space $\mathcal{M}_{ck}$, a commitment space $\mathcal{C}_{ck}$ and a randomiser space $\mathcal{R}_{ck}$. The sender commits to $m \in \mathcal{M}_{ck}$ by selecting $r \leftarrow \mathcal{R}_{ck}$ and computing the commitment $c = \mathrm{Com}_{ck}(m; r) \in \mathcal{C}_{ck}$.

Definition 2 (Hiding). *A commitment scheme* $(\mathrm{Gen}, \mathrm{Com})$ *is (computationally) hiding if for all probabilistic polynomial time stateful algorithms* $\mathcal{A}$

$$\Pr\left[ck \leftarrow \mathrm{Gen}(1^\lambda); (m_0, m_1) \leftarrow \mathcal{A}(ck); b \leftarrow \{0,1\}; c \leftarrow \mathrm{Com}_{ck}(m_b) : \mathcal{A}(c) = b\right] \approx \frac{1}{2}$$

If we have equality above then we say that the commitment scheme is perfectly hiding.

Definition 3 (Binding). *A commitment scheme is (computationally) binding if for all probabilistic polynomial time adversaries* $\mathcal{A}$

$$\Pr\left[\begin{array}{c} ck \leftarrow \mathrm{Gen}(1^\lambda); (m_0, r_1, m_1, r_1) \leftarrow \mathcal{A}(ck) : \\ \\ m_0 \neq m_1 \wedge \mathrm{Com}_{ck}(m_0; r_0) = \mathrm{Com}_{ck}(m_1; r_1) \end{array}\right] \approx 0$$

If we have equality above then we say that the commitment scheme is perfectly binding.

Suppose further that $(\mathcal{M}_{ck}, +)$, $(\mathcal{R}_{ck}, +)$ and $(\mathcal{C}_{ck}, \cdot)$ are groups.

Definition 4 (Homomorphic Commitment Scheme). *We call the commitment scheme homomorphic if for all* $\lambda \in \mathbb{N}$ *and for all* $ck \leftarrow \mathrm{Gen}(1^\lambda)$ *the commitment function* $\mathrm{Com} : \mathcal{M}_{ck} \times \mathcal{R}_{ck} \to \mathcal{C}_{ck}$ *is a group-homomorphism, i.e., for all* $m, m' \in \mathcal{M}_{ck}$ *and all* $r, r' \in \mathcal{R}_{ck}$

$$\mathrm{Com}_{ck}(m + m'; r + r') = \mathrm{Com}_{ck}(m; r) \cdot \mathrm{Com}_{ck}(m'; r')$$

Pedersen commitments. Our zero-knowledge arguments can be instantiated with any homomorphic, perfectly hiding and computationally binding commitment scheme. For concreteness, we will focus on the Pedersen commitment scheme [42] to multiple values. The generator outputs a description of a group of prime order p and a set of random group elements $ck = (p, \mathbb{G}, g_1, \ldots, g_n, h)$. The message space is $\mathbb{Z}_p^n$, the randomness space is $\mathbb{Z}_p$ and the commitment space is $\mathbb{G}$. To commit to a vector $\boldsymbol{m} = (m_1, \ldots, m_n)$ pick $r \leftarrow \mathbb{Z}_p$ and return the commitment $c = \mathrm{Com}_{ck}(\boldsymbol{m}; r) = h^r \prod_{i=1}^n g_i^{m_i}$. The Pedersen commitment scheme is homomorphic, perfectly hiding and computationally binding under the discrete logarithm assumption.

Throughout the paper, we make use of commitments for vectors of different sizes. We can use the same commitment key for this and just append the vectors with enough zeros to get length n.

2.3 Σ-Protocols

A Σ-protocol is a 3-move public-coin interactive protocol that enables a prover to convince a verifier that a particular statement is true. First, the prover sends an initial message to the verifier. The verifier sends back a randomly selected challenge. The prover responds to the challenge. Finally, the verifier decides whether or not to accept the proof based on the conversation.

We assume a probabilistic polynomial time algorithm $\mathcal{G}$ that generates a common reference string crs known to all parties. In this paper crs consists of the key for a homomorphic commitment scheme. For Pedersen commitments, this is just a list of random group elements.

Let R be a polynomial-time decidable relation. We call w a witness for statement u if $(crs, u, w) \in R$. A Σ-protocol for R is a collection of stateful probabilistic polynomial time algorithms $(\mathcal{G}, \mathcal{P}, \mathcal{V})$. The algorithm $\mathcal{G}$ provides a common reference string (which in our paper will be a commitment key as described above). Algorithms $\mathcal{P}, \mathcal{V}$ function as shown in Figure 2. The challenge space $\mathcal{X}$ is implicitly given by the common reference string. Intuitively, $\mathcal{V}$ outputs 1 if accepting the proof and 0 if rejecting.

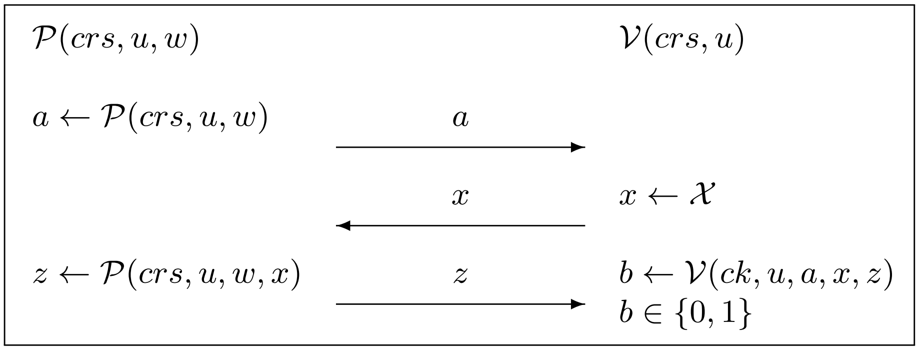
$\mathcal{P}(crs, u, w)$
$\mathcal{V}(crs, u)$
$a \leftarrow \mathcal{P}(crs, u, w)$
a
x
$x \leftarrow \mathcal{X}$
$z \leftarrow \mathcal{P}(crs, u, w, x)$
z
$b \leftarrow \mathcal{V}(ck, u, a, x, z)$
$b \in \{0, 1\}$

Fig. 2. A General Σ-Protocol

Algorithms $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ are a Σ-protocol if they satisfy completeness, special soundness, and special honest verifier zero-knowledge:

Definition 5 (Perfect Completeness). $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ *is perfectly complete if for all probabilistic polynomial time algorithms* $\mathcal{A}$*, we have*

$$\Pr\left[\begin{matrix} crs \leftarrow \mathcal{G}(1^\lambda); (u, w) \leftarrow \mathcal{A}(crs); a \leftarrow \mathcal{P}(crs, u, w); x \leftarrow \mathcal{X}; z \leftarrow \mathcal{P}(x) : \\ (crs, u, w) \notin R \text{ or } \mathcal{V}(crs, u, a, x, z) = 1 \end{matrix}\right] = 1$$

Definition 6 (n-Special Soundness). $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ *is* n*-special sound if there exists a probabilistic polynomial time algorithm* χ *that uses* n *accepting transcripts* *with the same initial message a and distinct challenges to compute the witness. For all probabilistic polynomial time algorithms $\mathcal{A}$*

$$\Pr\left[\begin{array}{c} crs \leftarrow \mathcal{G}(1^\lambda); (u, a, x_1, z_1, \dots, x_n, z_n) \leftarrow \mathcal{A}(crs); \\ w \leftarrow \chi(u, a, x_1, z_1, \dots, x_n, z_n) : \\ (crs, u, w) \in R \text{ or } \exists i \in [n] \text{ such that } \mathcal{V}(crs, u, a, x_i, z_i) \neq 1 \end{array}\right] \approx 1,$$

where the adversary outputs distinct $x_1, \dots, x_n$.

If the above holds with equality, then we say that $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ has perfect n-special soundness.

Definition 7 (Special Honest Verifier Zero Knowledge (SHVZK)). *We say that $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ has SHVZK if there exists a probabilistic polynomial time simulator $\mathcal{S}$ such that for all interactive probabilistic polynomial time algorithms $\mathcal{A}$*

$$\begin{aligned} &\Pr\left[crs \leftarrow \mathcal{G}(1^\lambda); (u, w, x) \leftarrow \mathcal{A}(crs); a \leftarrow \mathcal{P}(crs, u, w); z \leftarrow \mathcal{P}(x) : \mathcal{A}(crs, a, z) = 1\right] \\ \approx &\Pr\left[crs \leftarrow \mathcal{G}(1^\lambda); (u, w, x) \leftarrow \mathcal{A}(crs); (a, z) \leftarrow \mathcal{S}(crs, u, x) : \mathcal{A}(crs, a, z) = 1\right] \end{aligned}$$

If the above holds with equality, then we say that $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ has perfect SHVZK.

Full zero-knowledge. In real life applications special honest verifier zero-knowledge may not suffice since a malicious verifier may give non-random challenges. However, it is easy to convert an SHVZK argument into a full zero-knowledge argument secure against *arbitrary* verifiers in the common reference string model using standard techniques. The conversion can be very efficient and only costs a small additive overhead. Details of conversion methods can be found in [27,24,20].

2.4 Relations

In this section, we describe the relations for our zero-knowledge proofs. The prover's witness is a secret vector $\boldsymbol{a}$ satisfying some conditions, and an opening to a commitment C which is computed from $\boldsymbol{a}$.

This type of relation could be modelled using a relation with a polynomial $\boldsymbol{P}$ to impose conditions on $\boldsymbol{a}$, and another polynomial $\boldsymbol{Q}$ to compute the opening to C. The value r is the randomness used to make the commitment.

$$\mathbf{P}(\mathbf{a}) = \mathbf{0}, \quad C = \mathrm{Com}(\mathbf{Q}(\mathbf{a}); r)$$

For example, $\boldsymbol{a} = (a_0, a_1, a_2)$ could be a secret vector of bits, imposed by $\boldsymbol{P}(\boldsymbol{a}) = \boldsymbol{a} \circ (\mathbf{1} - \boldsymbol{a})$, and $\boldsymbol{Q}(\boldsymbol{a}) = a_0 + 2a_1 + 4a_2$ could compute the integer represented by the bits.

We also incorporate a public vector $\boldsymbol{b}$, which can be seen as a 'tweak' and allows modification of the statement. For example, setting $\boldsymbol{Q}(\boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{a} \cdot \boldsymbol{b}$, we can recover the range proof above by using $\boldsymbol{b} = (1, 2, 4)$. We can also get relations about other knapsacks by using a different value of $\boldsymbol{b}$.

More formally, let $\mathbf{P}(\mathbf{a}, \mathbf{b}), \mathbf{Q}(\mathbf{a}, \mathbf{b})$ be length ℓ_P, ℓ_Q vectors of polynomials of degrees d_P, d_Q respectively. Let C be a commitment. Let $\mathbf{b} \in \mathbb{Z}_p^{\ell_b}$ be a public vector of field elements. The prover gives a zero-knowledge argument of knowledge of $\mathbf{a} \in \mathbb{Z}_p^{\ell_a}$ and $r \in \mathbb{Z}_p$ such that

$$\mathbf{P}(\mathbf{a}, \mathbf{b}) = \mathbf{0}, \quad C = \mathrm{Com}(\mathbf{Q}(\mathbf{a}, \mathbf{b}); r)$$

We give more general batched proofs which can handle t instances at once. Let $C_1, \ldots, C_m$ be commitments. Let $t = mn$. Let $\mathbf{b}_{1,1}, \ldots, \mathbf{b}_{m,n} \in \mathbb{Z}_p^{\ell_b}$ be public vectors of field elements. The $\mathbf{b}_{i,j}$ values allow a single instance to capture some variation in the statement. The batched argument is an argument of knowledge of values $\{\mathbf{a}_{i,j}\}_{i,j=1}^{m,n}$ and $\{r_i\}_{i=1}^{m}$, such that $\mathbf{P}(\mathbf{a}_{i,j}, \mathbf{b}_{i,j}) = \mathbf{0}$ for $i \in [m], j \in [n]$, and the prover knows commitment openings

$$\begin{aligned}
C_1 &= \mathrm{Com}(\mathbf{Q}(\mathbf{a}_{1,1}, \mathbf{b}_{1,1}), \quad \mathbf{Q}(\mathbf{a}_{1,2}, \mathbf{b}_{1,2}), \quad \ldots, \mathbf{Q}(\mathbf{a}_{1,n}, \mathbf{b}_{1,n}); \quad r_1) \\
C_2 &= \mathrm{Com}(\mathbf{Q}(\mathbf{a}_{2,1}, \mathbf{b}_{2,1}), \quad \mathbf{Q}(\mathbf{a}_{2,2}, \mathbf{b}_{2,2}), \quad \ldots, \mathbf{Q}(\mathbf{a}_{2,n}, \mathbf{b}_{2,n}); \quad r_2) \\
&\vdots \\
C_m &= \mathrm{Com}(\mathbf{Q}(\mathbf{a}_{m,1}, \mathbf{b}_{m,1}), \mathbf{Q}(\mathbf{a}_{m,2}, \mathbf{b}_{m,2}), \ldots, \mathbf{Q}(\mathbf{a}_{m,n}, \mathbf{b}_{m,n}); r_m)
\end{aligned}$$

When $m = n = 1$, we have $t = 1$ and recover the relation for a zero-knowledge argument of knowledge for a single instance.

The idea is that $\boldsymbol{Q}$ allows the prover to prove things about parts of the witness that were included as commitments in the statement for the zero-knowledge proof. Then $\boldsymbol{P}$ deals with parts of the witness that were not included as commitments in the statement. Therefore, by choosing $\boldsymbol{P}$ and $\boldsymbol{Q}$ appropriately, we can easily deal with applications where the evaluation of a polynomial is known, and applications where it is in committed form.

We can easily generalise to the case where multiple polynomials $\boldsymbol{Q}_1(\boldsymbol{a}, \boldsymbol{b}), \ldots, \boldsymbol{Q}_k(\boldsymbol{a}, \boldsymbol{b})$ are given in separate commitments.

2.5 Lagrange Polynomials

Let $z_1, \ldots, z_m$ be distinct points in some field. The Lagrange polynomials $l_1(X), \ldots, l_m(X)$ are the unique polynomials of degree $m - 1$ such that $l_i(z_j) = \delta_{i,j}$, where $\delta_{i,j}$ is the Kronecker-delta. In cryptography, Lagrange polynomials have been used for secret-sharing [45].

For $j \in [m]$, $l_j(X)$ can be computed as

$$\ell_j(X) := \prod_{\substack{0 \le m \le k \\ m \ne j}} \frac{X - z_m}{z_j - z_m} = \frac{(X - z_0)}{(z_j - z_0)} \cdots \frac{(X - z_{j-1})}{(z_j - z_{j-1})} \frac{(X - z_{j+1})}{(z_j - z_{j+1})} \cdots \frac{(X - z_k)}{(z_j - z_k)}$$

3 Polynomial commitment schemes

We present a protocol which allows a prover to commit to a polynomial in the discrete-logarithm setting, using a homomorphic commitment scheme. The prover may then later reveal to the verifier an evaluation of the polynomial in a specific point x chosen by the verifier and prove the evaluation is correct. Bootle et al. [7] dealt with a similar problem for Laurent polynomials with constant term zero, whose coefficients were single field elements. We use the same techniques and generalise to the case of vector coefficients. We treat only positive powers and ignore the condition on the constant term since this suffices for our needs. However, the case of Laurent polynomials is straightforward and similar to [7].

3.1 Definition

A polynomial commitment scheme $(\mathrm{Gen}, \mathrm{PolyCommit}, \mathrm{PolyEval}, \mathrm{PolyVerify})$ enables a prover to commit to a secret vector of polynomials $\mathbf{h}(X) \in \mathbb{Z}_p^l[X]$ of some known degree N. Later on the prover may choose to evaluate the committed polynomial in a point $x \in \mathbb{Z}_p$ and send an opening to the verifier.

$\mathrm{Gen}(1^\lambda) \to ck$**:** Gen is a probabilistic polynomial time algorithm that returns a commitment key ck. The commitment key specifies among other things a prime p of size $|p| = \lambda$.

$\mathrm{PolyCommit}(ck, \mathbf{h}(X)) \to (\mathsf{msg}_1, \mathsf{st})$**:** PolyCommit is a probabilistic polynomial time algorithm that given a commitment key ck and a vector of degree N polynomials returns a commitment message msg_1 and a state st.

$\mathrm{PolyEval}(\mathsf{st}, x) \to \mathsf{msg}_2$**:** PolyEval is a deterministic polynomial time algorithm that given a state and a point $x \in \mathbb{Z}_p$ returns an evaluation message msg_2.

$\mathrm{PolyVerify}(ck, \mathsf{msg}_1, \mathsf{msg}_2, x) \to \bar{\mathbf{h}}$**:** PolyVerify is a deterministic polynomial time algorithm that given a commitment key, a commitment message, an evaluation message and a point $x \in \mathbb{Z}_p$ returns $\perp$ if it rejects the input, or a purported evaluation of the committed vector of polynomials in x.

A polynomial commitment scheme should be complete, $(m + 1)$-special sound and special honest verifier zero-knowledge as defined below.

The definition of completeness simply guarantees that if PolyCommit and PolyVerify are carried out honestly, then PolyVerify will return the correct polynomial evaluation $\mathbf{h}(x)$.

Definition 8 (Perfect Completeness).
$(\mathrm{Gen}, \mathrm{PolyCommit}, \mathrm{PolyEval}, \mathrm{PolyVerify})$ *has perfect completeness if for all* $\lambda \in \mathbb{N}$*, for all* $ck \leftarrow \mathrm{Gen}(1^\lambda)$*, and all* $\mathbf{h}(X) \in \mathbb{Z}_p^l[X]$ *of degree* N*, and all* $x \in \mathbb{Z}_p$

$$\Pr\left[\begin{array}{ll} (\mathsf{msg}_1, \mathsf{st}) \leftarrow \mathrm{PolyCommit}(ck, \mathbf{h}(X)) & \\ \mathsf{msg}_2 \leftarrow \mathrm{PolyEval}(\mathsf{st}, x) & : \bar{\mathbf{h}} = \mathbf{h}(x) \\ \bar{\mathbf{h}} \leftarrow \mathrm{PolyVerify}(ck, \mathsf{msg}_1, \mathsf{msg}_2, x) & \end{array}\right] = 1.$$

The definition of $(m+1)$-Special Soundness guarantees that given $m+1$ accepting evaluations for different evaluation points, but from the same polynomial commitment message msg_1, then it is possible to extract a polynomial $\mathbf{h}(X)$ that is consistent with the evaluations produced. Furthermore, any other accepting evaluations for the same commitment will also be evaluations of $\mathbf{h}(X)$.

Definition 9 (Computational $(m+1)$-Special Soundness).
$(\mathrm{Gen}, \mathrm{PolyCommit}, \mathrm{PolyEval}, \mathrm{PolyVerify})$ *is $(m+1)$-special sound if there exists a probabilistic polynomial time algorithm χ that uses $m+1$ accepting transcripts with the same commitment message msg_1 to compute the committed polynomial $\mathbf{h}(X)$. For all probabilistic polynomial time adversaries $\mathcal{A}$ and all $L \geq m$*

$$\Pr\left[\begin{array}{l} ck \leftarrow \mathrm{Gen}(1^\lambda) \\ (\mathsf{msg}_1, x^{(0)}, \mathsf{msg}_2^{(0)}, \ldots, x^{(L)}, \mathsf{msg}_2^{(L)}) \leftarrow \mathcal{A}(ck) \\ \mathbf{h}(X) \leftarrow \chi(ck, \mathsf{msg}_1, x^{(0)}, \mathsf{msg}_2^{(0)}, \ldots, x^{(m)}, \mathsf{msg}_2^{(m)}) \\ \bar{\mathbf{h}}_i \leftarrow \mathrm{PolyVerify}(ck, \mathsf{msg}_1, \mathsf{msg}_2^{(i)}, x^{(i)}) \end{array} : \begin{array}{l} \textit{There is a } \bar{\mathbf{h}}_i = \bot \\ \textit{or all } \bar{\mathbf{h}}_i = \mathbf{h}(x^{(i)}) \end{array}\right] \approx 1,$$

where the adversary outputs distinct points $x^{(0)}, \ldots, x^{(L)} \in \mathbb{Z}_p$ and the extractor returns a degree N vector of polynomials.

Perfect special honest verifier zero-knowledge means that given any evaluation point x and an evaluation $\boldsymbol{h}(x)$, it is possible to simulate $\mathsf{msg}_1, \mathsf{msg}_2$ that are distributed exactly as in a real execution of the protocol, in a way that is consistent with the evaluation $\boldsymbol{h}(x)$.

Definition 10 (Perfect Special Honest Verifier Zero Knowledge).
$(\mathrm{Gen}, \mathrm{PolyCommit}, \mathrm{PolyEval}, \mathrm{PolyVerify})$ *has* perfect special honest verifier zero knowledge (SHVZK) *if there exists a probabilistic polynomial time simulator $\mathcal{S}$ such that for all stateful probabilistic polynomial time adversaries $\mathcal{A}$*

$$\Pr\left[\begin{array}{l} ck \leftarrow \mathrm{Gen}(1^\lambda); (\mathbf{h}(X), x) \leftarrow \mathcal{A}(ck) \\ (\mathsf{msg}_1, \mathsf{st}) \leftarrow \mathrm{PolyCommit}(ck, \mathbf{h}(X)) \\ \mathsf{msg}_2 \leftarrow \mathrm{PolyEval}(\mathsf{st}, x) \end{array} : \mathcal{A}(\mathsf{msg}_1, \mathsf{msg}_2) = 1\right]$$
$$= \Pr\left[\begin{array}{l} ck \leftarrow \mathrm{Gen}(1^\lambda); (\mathbf{h}(X), x) \leftarrow \mathcal{A}(ck) \\ (\mathsf{msg}_1, \mathsf{msg}_2) \leftarrow \mathcal{S}(ck, x, \mathbf{h}(x)) \end{array} : \mathcal{A}(\mathsf{msg}_1, \mathsf{msg}_2) = 1\right]$$

3.2 Construction

In the following, we will build a polynomial commitment scheme on top of a perfectly-hiding, homomorphic commitment scheme $(\mathrm{Gen}, \mathrm{Com})$ to vectors in $\mathbb{Z}_p^{nl}$. Let us first give some intuition about how the construction will work.

Let $\mathbf{h}(X) = \sum_{i=0}^{N} \mathbf{h}_i X^i$ be a polynomial of degree $N = (n+1)m - 1$ with coefficients that are row-vectors in $\mathbb{Z}_p^l$. Define an $m \times (n+1)l$ matrix

$$\begin{pmatrix} \mathbf{h}_{0,0} & \mathbf{h}_{0,1} & \cdots & \mathbf{h}_{0,n} \\ \mathbf{h}_{1,0} & \mathbf{h}_{1,1} & \cdots & \mathbf{h}_{1,n} \\ \vdots & \vdots & \ddots & \\ \mathbf{h}_{m-1,0} & \mathbf{h}_{m-1,1} & \cdots & \mathbf{h}_{m-1,n} \end{pmatrix} = \begin{pmatrix} \mathbf{h}_0 & \mathbf{h}_m & \cdots & \mathbf{h}_{nm} \\ \mathbf{h}_1 & \mathbf{h}_{m+1} & \cdots & \mathbf{h}_{nm+1} \\ \vdots & \vdots & \ddots & \\ \mathbf{h}_{m-1} & \mathbf{h}_{2m-1} & \cdots & \mathbf{h}_N \end{pmatrix}$$

With this matrix we have $\mathbf{h}(X) = \sum_{j=0}^{n}(\sum_{i=0}^{m-1} \mathbf{h}_{i,j} X^i) X^{mj}$. In the polynomial commitment scheme, the prover commits to each row of the matrix with commitments $\{H_i\}_{i=0}^{m-1}$. After receiving a point x from the verifier, the prover computes for each column $\bar{\mathbf{h}}_j = \sum_{i=0}^{m} \mathbf{h}_{i,j} x^i$ and sends them to the verifier as part of openings of the commitment $\prod_{i=0}^{m-1} H_i^{x^i}$. The verifier can use the homomorphic property of the commitments to check that the $\bar{\mathbf{h}}_j$ values are correctly formed and compute $\mathbf{h}(x) = \sum_{j=0}^{n} \bar{\mathbf{h}}_j x^{jm}$.

While the main idea we have sketched above gives the verifier assurance that the committed polynomial has been correctly evaluated, the prover may not be happy. The problem is that the solution gives away information about the coefficients of $\mathbf{h}(X)$. We will therefore introduce some random blinding vectors to ensure no information is leaked about the committed coefficients except the evaluation of the polynomial. We will also adjust the protocol to handle an arbitrary polynomial degree $N = mn + d$ for $0 \leq d < m$ by shifting the first column of the matrix.

We pick random blinders $\mathbf{b}_1, \ldots, \mathbf{b}_n \leftarrow \mathbb{Z}_p^l$ and define an $(m+1) \times (n+1)l$ matrix $\{\mathbf{h}_{i,j}\}_{i=0,j=0}^{m,n}$ as follows:

$$\begin{pmatrix}
\mathbf{h}_0 & \mathbf{b}_1 & \cdots & \mathbf{b}_{n-1} & \mathbf{b}_n \\
\mathbf{h}_1 & \mathbf{h}_{d+1} & \cdots & \mathbf{h}_{(n-2)m+d+1} & \mathbf{h}_{(n-1)m+d+1} \\
 & & & & \vdots \\
\mathbf{h}_d - \mathbf{b}_1 & \vdots & \ddots & & \mathbf{h}_{nm} \\
0 & & & & \mathbf{h}_{nm+1} \\
\vdots & & & & \vdots \\
0 & \mathbf{h}_{m+d-1} & \cdots & \mathbf{h}_{(n-2)m+d-1} & \mathbf{h}_{N-1} \\
0 & \mathbf{h}_{m+d} - \mathbf{b}_2 & \cdots & \mathbf{h}_{(n-2)m+d} - \mathbf{b}_n & \mathbf{h}_N
\end{pmatrix}$$

We can therefore rewrite the polynomial as

$$\mathbf{h}(X) = \sum_{i=0}^{m} \mathbf{h}_{i,0} X^i + \sum_{j=1}^{n} \left(\sum_{i=0}^{m} \mathbf{h}_{i,j} X^i \right) X^{(j-1)m+d}.$$

In the polynomial commitment scheme, the prover commits to each row of the matrix with commitments $\{H_i\}_{i=0}^{m}$. After receiving a point x from the verifier, the prover computes for each column $\bar{\mathbf{h}}_j = \sum_{i=0}^{m} \mathbf{h}_{i,j} x^i$ and sends them to the verifier as part of an opening of the commitment $\prod_{i=0}^{m} H_i^{x^i}$. The verifier can use the opening to check that the $\bar{\mathbf{h}}_j$ values are correct and compute $\mathbf{h}(x) = \bar{\mathbf{h}}_0 + \sum_{j=1}^{n} \bar{\mathbf{h}}_j x^{(j-1)m+d}$. We describe the full polynomial commitment scheme below.

Common Input: ck

PolyCommit$(ck, \mathbf{h}(X)) \to (\mathsf{msg}_1, \mathsf{st})$**:** The prover randomly selects $\mathbf{b}_1, \dots, \mathbf{b}_n \leftarrow \mathbb{Z}_p^l$ and arranges them into a matrix with entries $\{\mathbf{h}_{i,j}\}_{i=0,j=0}^{m,n}$ as follows:

$$\begin{pmatrix} \mathbf{h}_0 & \mathbf{b}_1 & \cdots & \mathbf{b}_{n-1} & \mathbf{b}_n \\ \mathbf{h}_1 & \mathbf{h}_{d+1} & \cdots & \mathbf{h}_{(n-2)m+d+1} & \mathbf{h}_{(n-1)m+d+1} \\ & & & & \vdots \\ \mathbf{h}_d - \mathbf{b}_1 & \vdots & \ddots & & \mathbf{b}_n \\ 0 & & & & \mathbf{h}_{nm+1} \\ \vdots & & & & \vdots \\ 0 & \mathbf{h}_{m+d-1} & \cdots & \mathbf{h}_{(n-2)m+d-1} & \mathbf{h}_{N-1} \\ 0 & \mathbf{h}_{m+d} - \mathbf{b}_2 & \cdots & \mathbf{h}_{(n-2)m+d} - \mathbf{b}_n & \mathbf{h}_N \end{pmatrix}$$

For $0 \leq i \leq m$, the prover randomly selects $r_i \leftarrow \mathbb{Z}_p$ and computes a commitment H_i to the ith row of the matrix using randomness r_i.

$$\mathsf{msg}_1 = (\{H_i\}_{i=0}^m), \qquad \mathsf{st} = \big(\mathbf{h}(X), \{\mathbf{b}_j\}_{j=1}^n, \{r_i\}_{i=0}^m\big)$$

The prover sends msg_1 to the verifier.

PolyEval(st, x)**:** $\to (\mathsf{msg}_2)$**:** For $0 \leq j \leq n$, the prover computes

$$\bar{\mathbf{h}}_j = \sum_{i=0}^m \mathbf{h}_{i,j} x^{i+1}.$$

The prover also computes $\bar{r} = \sum_{i=0}^m r_i x^i$.
Set $\mathsf{msg}_2 = \big(\{\bar{\mathbf{h}}_j\}_{j=0}^n, \bar{r}\big)$.
The prover sends msg_2 to the verifier.

PolyVerify$(ck, \mathsf{msg}_1, \mathsf{msg}_2, x)$**:** $\to (\mathsf{cmt})$**:** The verifier checks whether

$$\mathrm{com}(\bar{\mathbf{h}}_0, \dots, \bar{\mathbf{h}}_n; \bar{r}) = \prod_{i=0}^m H_i^{x^i}.$$

Return $\perp$ if this fails.
After accepting the commitment opening, the verifier returns

$$\bar{\mathbf{h}} = \sum_{i=0}^m \mathbf{h}_{i,0} x^i + \sum_{j=1}^n \left(\sum_{i=0}^m \mathbf{h}_{i,j} x^i \right) x^{(j-1)m+d}.$$

Lemma 1. *The polynomial commitment protocol given above has perfect completeness, computational* $(m+1)$*-special-soundness, and perfect special honest verifier zero-knowledge.*

Proof. By inspection, it follows that when the prover is honest, the verifier always recovers $\bar{\mathbf{h}} = \mathbf{h}(x)$.

Given x and $\mathbf{h}(x)$, we describe an efficient simulator to prove special honest verifier zero knowledge. The simulator first picks random $\bar{\mathbf{h}}_1, \ldots, \bar{\mathbf{h}}_n \leftarrow \mathbb{Z}_p^l$ and then computes $\bar{\mathbf{h}}_0 = \mathbf{h}(x) - \sum_{j=1}^{n} \bar{\mathbf{h}}_j x^{(j-1)m+d}$. In other words, the $\boldsymbol{h}_j$ are chosen uniformly at random, conditional on giving the correct evaluation $\boldsymbol{h}(x)$. The simulator also picks at random $\bar{r} \in \mathbb{Z}_p$ and $r_1, \ldots, r_m \leftarrow \mathbb{Z}_p$ and sets $H_i = \mathrm{Com}_{ck}(\mathbf{0}; r_i)$. Finally, it computes $H_0 = \mathrm{Com}_{ck}(\bar{\mathbf{h}}_0, \ldots, \bar{\mathbf{h}}_n; \bar{r}) \prod_{i=1}^{m} H_i^{-x^i}$.

This is a perfect SHVZK simulation. First, because the commitment scheme is perfectly hiding, the commitments $H_1, \ldots, H_m$ are identically distributed in real proofs and simulated proofs. The values $\bar{\mathbf{h}}_1, \ldots, \bar{\mathbf{h}}_n$ and $\bar{r}$ are also independently and uniformly at random in real proofs due to the choices of $\mathbf{b}_1, \ldots, \mathbf{b}_n$ and r_0, just as in the simulated proofs. Finally, given these random values both real and simulated proofs, the matching H_0 and $\bar{\mathbf{h}}_0$ are uniquely determined. This means we have identical distributions of real and simulated proofs which are consistent with the evaluation $\boldsymbol{h}(x)$.

Finally, we prove $(m+1)$-special soundness. Suppose that we are given msg_1 and $x^{(0)}, \ldots, x^{(m)}, \mathsf{msg}_2^{(0)}, \ldots, \mathsf{msg}_2^{(m)}$ which are all accepting, and where the $x^{(i)}$ are distinct. Consider the Vandermonde matrix:

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ x^{(0)} & x^{(1)} & \cdots & x^{(m)} \\ \vdots & \vdots & \ddots & \vdots \\ \left(x^{(0)}\right)^m & \left(x^{(1)}\right)^m & \cdots & \left(x^{(m)}\right)^m \end{pmatrix}$$

This matrix is invertible, meaning that for any $0 \leq k \leq m$, we can take linear combinations of the columns to obtain $(0, \ldots, 0, 1, 0, \ldots, 0)^T$, where the kth entry is 1. We may take the same linear combinations of the verification equation $\mathrm{com}(\bar{\mathbf{h}}_0, \ldots, \bar{\mathbf{h}}_n; \bar{r}) = \prod_{i=0}^{m} H_i^{x^i}$ in order to find openings to each H_k. We now have that $H_0, \ldots, H_m$ are commitments to known row vectors $(\mathbf{h}_{i,0}, \ldots, \mathbf{h}_{i,n})$ with known randomness r_i. We define the extracted vector of polynomials to be $\mathbf{h}(X) = \sum_{i=0}^{m} \mathbf{h}_{i,0} X^i + \sum_{j=1}^{n} \left(\sum_{i=0}^{m} \mathbf{h}_{i,j} X^i \right) X^{(j-1)m+d}$, which is a vector of degree N polynomials.

By the binding property of the commitment scheme, for each accepting transcript, we have

$$\bar{\mathbf{h}}_k = \sum_{i=0}^{m} \mathbf{h}_{i,0} (x^{(k)})^i + \sum_{j=1}^{n} \left(\sum_{i=0}^{m} \mathbf{h}_{i,j} (x^{(k)})^i \right) (x^{(k)})^{(j-1)m+d}.$$

Therefore, all openings are consistent with the extracted polynomial $\mathbf{h}(X)$. □

Communication. The prover must send $m+1$ group elements and $l(n+1)+1$ field elements to the verifier.

Computation. Prover computation is dominated by $m+1$ multi-exponentiations of width $l(n+1)+1$ costing approximately $\frac{lmn}{\log ln} + \frac{l}{\log l}$ exponentiations. Verifier computation is dominated by a multi-exponentiation of width $l(n+1)+m+1$ costing approximately $\frac{ln+m}{\log(ln+m)}$ exponentiations.

4 Batch Protocol for Low Degree Relations

We give an argument of knowledge of values $\{\mathbf{a}_{i,j}\}_{i\in[m],j\in[n]}$ and $\{r_i\}_{i\in[m]}$, such that $\mathbf{P}(\mathbf{a}_{i,j}, \mathbf{b}_{i,j}) = \mathbf{0}$ for $i \in [m], j \in [n]$, and the prover knows commitment openings

$$\begin{aligned}
C_1 &= \mathrm{com}(\mathbf{Q}(\mathbf{a}_{1,1}, \mathbf{b}_{1,1}), \quad \mathbf{Q}(\mathbf{a}_{1,2}, \mathbf{b}_{1,2}), \quad \dots, \mathbf{Q}(\mathbf{a}_{1,n}, \mathbf{b}_{1,n}); \quad r_1)\\
C_2 &= \mathrm{com}(\mathbf{Q}(\mathbf{a}_{2,1}, \mathbf{b}_{2,1}), \quad \mathbf{Q}(\mathbf{a}_{2,2}, \mathbf{b}_{2,2}), \quad \dots, \mathbf{Q}(\mathbf{a}_{2,n}, \mathbf{b}_{2,n}); \quad r_2)\\
&\vdots\\
C_m &= \mathrm{com}(\mathbf{Q}(\mathbf{a}_{m,1}, \mathbf{b}_{m,1}), \mathbf{Q}(\mathbf{a}_{m,2}, \mathbf{b}_{m,2}), \dots, \mathbf{Q}(\mathbf{a}_{m,n}, \mathbf{b}_{m,n}); r_m)
\end{aligned}$$

The protocol we design will be more efficient than repeating $t = mn$ instances of the basic protocol in parallel, as the communication depends on $\sqrt{t}$ rather than t.

In the following we will refer to the parameters $\ell_a, \ell_b, \ell_P, d_P, \ell_Q, d_Q$ such that $\mathbf{a}_{i,j} \in \mathbb{Z}_p^{\ell_a}$, $\mathbf{b}_{i,j} \in \mathbb{Z}_p^{\ell_b}$, $\mathbf{P}$ is a vector of ℓ_P $(\ell_a + \ell_b)$-variate polynomials of total degree d_P, and $\mathbf{Q}$ is a vector of ℓ_Q $(\ell_a + \ell_b)$-variate polynomials of total degree d_Q.

4.1 Intuition behind Protocol

The protocol embeds multiple instances of the same polynomial equality into a single polynomial by using Lagrange interpolation polynomials, inspired by [26,1]. To recover a single instance, simply evaluate the polynomial in one of the interpolation points.

More concretely, let $z_1, \dots, z_m$ be distinct points in $\mathbb{Z}_p$, and let $l_1(X), \dots, l_m(X)$ be their associated Lagrange polynomials such that $l_i(z_j) = \delta_{i,j}$. Let $l_0(X) = \prod_{i=1}^m (X - z_i)$. The prover produces the following commitments.

$$\begin{aligned}
A_0 &= \mathrm{com}(\mathbf{a}_{0,1}, \ \mathbf{a}_{0,2}, \ \dots, \mathbf{a}_{0,n} \ ; r_0 \ )\\
A_1 &= \mathrm{com}(\mathbf{a}_{1,1}, \ \mathbf{a}_{1,2}, \ \dots, \mathbf{a}_{1,n} \ ; r_1 \ )\\
A_2 &= \mathrm{com}(\mathbf{a}_{2,1}, \ \mathbf{a}_{2,2}, \ \dots, \mathbf{a}_{2,n} \ ; r_2 \ )\\
&\vdots\\
A_m &= \mathrm{com}(\mathbf{a}_{m,1}, \mathbf{a}_{m,2}, \dots, \mathbf{a}_{m,n} \ ; r_m \ )
\end{aligned}$$

Here, the values $\mathbf{a}_{0,1}, \dots, \mathbf{a}_{0,n} \in \mathbb{Z}_p^{l_a}$, where the value of the first index is 0, are blinding values chosen uniformly at random. These are completely unrelated to the values of the witness, which are $\boldsymbol{a}_{1,1}, \dots, \boldsymbol{a}_{m,n}$, where the first index has a value strictly greater than 0. After receiving a random challenge x from the verifier, the prover sends $\bar{\mathbf{a}}_j = \sum_{i=0}^m \mathbf{a}_{i,j} l_i(x)$ to the verifier for each $j \in [n]$.

The verifier now checks the received $\bar{\mathbf{a}}_j$ against the commitments A_i. This proves knowledge of the $\boldsymbol{a}$ values. It remains to demonstrate that $\mathbf{a}_{i,j}, \mathbf{b}_{i,j}$ satisfy the polynomial relations in the statement. Let $\bar{\mathbf{b}}_j = \sum_{i=1}^m \mathbf{b}_{ij} l_i(x)$. The verifier evaluates $\mathbf{P}$, $\mathbf{Q}$ using $\bar{\mathbf{a}}_j$ and $\bar{\mathbf{b}}_j$ for each j. By definition of $\bar{\mathbf{a}}_j$ and $\bar{\mathbf{b}}_j$, when evaluating at an interpolation point z_i, we obtain the single evaluation of the original polynomial, $\mathbf{P}(\mathbf{a}_{i,j}, \mathbf{b}_{i,j})$. This implies, for example, that $\mathbf{P}(\bar{\mathbf{a}}_j, \bar{\mathbf{b}}_j) \equiv \mathbf{0} \mod l_0(x)$, or in other words, that $\mathbf{P}(\bar{\mathbf{a}}_j, \bar{\mathbf{b}}_j)$ is a multiple of $l_0(X)$ for each j. The prover must commit to the coefficients of $\mathbf{P}(\bar{\mathbf{a}}_j, \bar{\mathbf{b}}_j)/l_0(x)$ in advance (as a polynomial in x), and uses the polynomial commitment scheme to achieve this for every j simultaneously.

Finally, the prover needs to convince the verifier that the commitments C_i contain commitments to $\mathbf{Q}(\mathbf{a}_{i,j}, \mathbf{b}_{i,j})$. This is done in a similar way to the $\mathbf{P}$ polynomial, except here we build up polynomial equalities over committed values. The full protocol can be found below.

Common Reference String: $crs = (ck, z_1, \ldots, z_m)$ where $ck \leftarrow \mathrm{Gen}(1^\lambda)$ and $z_1, \ldots, z_m$ are distinct points in $\mathbb{Z}_p$ defining Lagrange polynomials $l_1(X), \ldots, l_m(X)$ such that $l_i(z_j) = \delta_{i,j}$ and defining $l_0(X) = \prod_{j=1}^m (X - z_j)$.

Statement: $\{C_i\}_{i\in[m]}, \{\mathbf{b}_{i,j}\}_{i\in[m],j\in[n]}, \mathbf{P}, \mathbf{Q}$ polynomials.

Prover's Witness: $\{\mathbf{a}_{i,j}\}_{i\in[m],j\in[n]}, \{r_i\}_{i\in[m]}$ such that

$$\mathbf{P}(\mathbf{a}_{i,j}, \mathbf{b}_{i,j}) = \mathbf{0} \text{ for } i \in [m], j \in [n]$$

$$C_i = \mathrm{com}(\mathbf{Q}(\mathbf{a}_{i,1}, \mathbf{b}_{i,1}), \mathbf{Q}(\mathbf{a}_{i,2}, \mathbf{b}_{i,2}), \ldots, \mathbf{Q}(\mathbf{a}_{i,n}, \mathbf{b}_{i,n}); r_i) \text{ for } i \in [m]$$

P → V: Pick $r_0, s_0, \ldots, s_m \leftarrow \mathbb{Z}_p$ and $\mathbf{a}_{0,1}, \ldots, \mathbf{a}_{0,n} \leftarrow \mathbb{Z}_p^{\ell_a}$ and $\mathbf{c}_1, \ldots, \mathbf{c}_n \leftarrow \mathbb{Z}_p^{\ell_Q}$. Compute

$$C_0 = \mathrm{Com}_{ck}(\mathbf{c}_1, \ldots, \mathbf{c}_n; r_0) \text{ and } A_i = \mathrm{Com}_{ck}\left(\mathbf{a}_{i,1}, \ldots, \mathbf{a}_{i,n}; s_i\right) \text{ for } i \in \{0\}\cup[m].$$

Define

$$\bar{\mathbf{a}}_j(X) = \textstyle\sum_{i=0}^m \mathbf{a}_{i,j} l_i(X) \qquad \bar{\mathbf{b}}_j(X) = \textstyle\sum_{i=1}^m \mathbf{b}_{i,j} l_i(X)$$

$$\mathbf{P}_j^*(X) = \frac{\mathbf{P}\big(\bar{\mathbf{a}}_j(X), \bar{\mathbf{b}}_j(X)\big)}{l_0(X)} \qquad \mathbf{Q}_j^*(X) = \mathbf{c}_j + \frac{\sum_{i=1}^m \mathbf{Q}(\mathbf{a}_{i,j}, \mathbf{b}_{i,j}) l_i(X) - \mathbf{Q}\big(\bar{\mathbf{a}}_j(X), \bar{\mathbf{b}}_j(X)\big)}{l_0(X)}$$

Run $\mathrm{PolyCommit}(ck, \{\mathbf{P}_j^*(X)\}_{j\in[n]}) \rightarrow (\mathsf{msg}_{P,1}, \mathsf{st}_P)$.

Run $\mathrm{PolyCommit}(ck, \{\mathbf{Q}_j^*(X)\}_{j\in[n]}) \rightarrow (\mathsf{msg}_{Q,1}, \mathsf{st}_Q)$.

The prover sends $\{A_i\}_{i\in[m]}$ and $\mathsf{msg}_{P,1}, \mathsf{msg}_{Q,1}$ to the verifier.

P ← V: Send the challenge $x \leftarrow \mathbb{Z}_p \setminus \{z_1, \ldots, z_m\}$ to the prover.

P → V: Run

$$\mathrm{PolyEval}(\mathsf{st}_P, x) \rightarrow \mathsf{msg}_{P,2} \qquad \mathrm{PolyEval}(\mathsf{st}_Q, x) \rightarrow \mathsf{msg}_{Q,2}.$$

Compute

$$\bar{\mathbf{a}}_j = \bar{\mathbf{a}}_j(x) \qquad \bar{r} = \sum_{i=0}^m r_i l_i(x) \qquad \bar{s} = \sum_{i=0}^m s_i l_i(x).$$

The prover sends $\{\bar{\mathbf{a}}_j\}_{j\in[n]}, \bar{r}, \bar{s}, \mathsf{msg}_{P,2}, \mathsf{msg}_{Q,2}$ to the verifier.

V: Run

$$\text{PolyVerify}(ck, \mathsf{msg}_{P,1}, \mathsf{msg}_{P,2}, x) \rightarrow \bar{\mathbf{p}} = (\bar{\mathbf{p}}_1, \ldots, \bar{\mathbf{p}}_n)$$

and

$$\text{PolyVerify}(ck, \mathsf{msg}_{Q,1}, \mathsf{msg}_{Q,2}, x) \rightarrow \bar{\mathbf{q}} = (\bar{\mathbf{q}}_1, \ldots, \bar{\mathbf{q}}_n).$$

Return 0 if $\bar{\mathbf{p}} = \perp$ or $\bar{\mathbf{q}} = \perp$.
Check

$$\text{Com}_{ck}(\bar{\mathbf{a}}_1, \ldots, \bar{\mathbf{a}}_n; \bar{s}) = \prod_{i=0}^{m} A_i^{l_i(x)}.$$

Compute $\bar{\mathbf{b}}_j = \bar{\mathbf{b}}_j(x)$ and check for all $j \in [n]$ that

$$\mathbf{P}(\bar{\mathbf{a}}_j, \bar{\mathbf{b}}_j) = \bar{\mathbf{p}}_j l_0(x).$$

Check that

$$\text{Com}_{ck}(\{\bar{\mathbf{q}}_j l_0(x) + \mathbf{Q}(\bar{\mathbf{a}}_j, \bar{\mathbf{b}}_j)\}_{j \in [n]}; \bar{r}) = \prod_{i=0}^{m} C_i^{l_i(x)}.$$

If all checks are satisfied, then the verifier outputs 1, and otherwise 0.

Lemma 2. *The batch protocol has perfect completeness, m_s-special-soundness, and perfect special honest verifier zero-knowledge, where $m_s = (m \max(d_P, d_Q) + 1)$.*

Proof. Perfect completeness of the protocol follows by perfect completeness of the PolyCommit sub-protocol, and by careful inspection.

For perfect special honest verifier zero knowledge, we provide an efficient simulator for the protocol. The simulator selects $z_1, \ldots, z_m$ as the prover. She then selects $\bar{\mathbf{a}}_j \leftarrow \mathbb{Z}_p^{\ell_a}$, $\bar{r}, \bar{s} \leftarrow \mathbb{Z}_p$, $\bar{\mathbf{q}}_j \leftarrow \mathbb{Z}_p^{\ell_Q}$, and $A_1, \ldots, A_m$ as uniformly random commitments to 0. All these values are distributed exactly as in a real protocol, where they are also uniformly random.

She then simulates the polynomial commitment and evaluation messages $\mathsf{msg}_{P,1}, \mathsf{msg}_{P,2}, \mathsf{msg}_{Q,1}, \mathsf{msg}_{Q,2}$ using the evaluation point x and evaluations $\bar{\boldsymbol{p}}$ and $\bar{\boldsymbol{q}}$, which are determined by the values already simulated. By the perfect SHVZK of the polynomial commitment scheme, the simulated values have identical distribution to the real proofs. Furthermore, since the polynomial commitment simulator takes the polynomial evaluation as input, the simulated polynomial commitments are consistent with the rest of the simulated values in the outer protocol.

In both the real and simulated protocols, the verification equations now determine the values of A_0 and C_0 uniquely, and the simulator can easily compute the correct values by rearranging the equations. The entire simulated proof therefore has the same distribution as a real proof.

Finally, we prove special soundness. Suppose that we have $m_s = (m \max(d_P, d_Q) + 1)$ accepting transcripts for the same first message, and distinct challenges x.

Pick any $m+1$ of the challenges, and note that the matrix

$$M = \begin{pmatrix} l_0(x^{(1)}) & l_1(x^{(1)}) & \cdots & l_m(x^{(1)}) \\ l_0(x^{(2)}) & l_1(x^{(2)}) & \cdots & l_m(x^{(2)}) \\ \vdots & \vdots & \ddots & \vdots \\ l_0(x^{(m+1)}) & l_1(x^{(m+1)}) & \cdots & l_m(x^{(m+1)}) \end{pmatrix}$$

is invertible. This follows from linear independence of the polynomials $l_0(X), \dots, l_m(X)$. If the determinant was zero, there would be a non-trivial linear dependence between the columns of the matrix. This would give a non-trival dependence relation between the polynomials.

Therefore, for each i, it is possible to take a linear combination of the rows to produce $(0, \dots, 0, 1, 0, \dots, 0)$, where the 1 is at the ith entry. By taking the same linear combinations of the left and right hand sides of the verification equation $\mathrm{Com}_{ck}(\bar{\mathbf{a}}_1, \dots, \bar{\mathbf{a}}_n; \bar{s}) = \prod_{i=0}^{m} A_i^{l_i(x)}$ for $m+1$ different transcripts, we can for each $i \in \{0\} \cup [m]$ extract an opening $\{\mathbf{a}_{i,j}\}_{j\in[n]}$ and s_i of A_i. By the binding property of the commitment scheme, we now have in each transcript that $\bar{\mathbf{a}}_j$ is correctly formed as a polynomial determined by the openings of the A_i evaluated in x.

By the special soundness of the polynomial commitment protocols, we extract polynomials $\mathbf{P}_j^*(X)$ of degree $(d_P - 1)m$, and $\mathbf{Q}_j^*(X)$ of degree $(d_Q - 1)m$ such that in each transcript, $\bar{\mathbf{p}}_j = \mathbf{P}_j^*(x)$ and $\bar{\mathbf{q}} = \mathbf{Q}_j^*(x)$ for the challenge x appearing in that transcript.

Consider the verification equations $\mathbf{P}(\bar{\mathbf{a}}_j, \bar{\mathbf{b}}_j) = \bar{\mathbf{p}}_j l_0(x)$. By the binding property of the commitment scheme, we have that $\mathbf{P}(\bar{\mathbf{a}}_j(x), \bar{\mathbf{b}}_j(x)) = \mathbf{P}_j^*(x) l_0(x)$ holds for m_s different challenges x. Since m_s is larger than the degree of the polynomial this implies that we have an equality of polynomials. By evaluating the polynomial expression at a particular interpolation point z_i, and parsing the resulting vector correctly, we see that $\mathbf{P}(\mathbf{a}_{i,j}, \mathbf{b}_{i,j}) = \mathbf{P}_j^*(z_i) l_0(z_i) = \mathbf{0}$ for each i, j.

We can in a similar manner to the extraction of the A_i extract openings of all C_i to values $\mathbf{c}_{i,1}, \dots, \mathbf{c}_{i,n}$. The last verification equation tells us that for each $j \in [n]$

$$\bar{\mathbf{q}}_j l_0(x) + \mathbf{Q}(\bar{\mathbf{a}}_j, \bar{\mathbf{b}}_j) = \sum_{i=0}^{m} c_{i,j} l_i(x).$$

Since m_s is larger than the degree of the polynomials this implies that we have an equality of polynomials. By plugging in the evaluation points z_i, we get $\mathbf{Q}(\mathbf{a}_{i,j}, \mathbf{b}_{i,j}) = c_{i,j}$ for each $i \in [m], j \in [n]$. □

Communication Let k_1, k_2 be the dimensions of the matrix used in the PolyCommit subprotocol when committing to $\mathbf{P}^*$, and similarly, let t_1, t_2 be the dimensions of the matrix in the subprotocol for committing to $\mathbf{Q}^*$. The total communication cost of the protocol is $m + k_1 + t_1 + 4$ group elements and $\ell_a n + \ell_P n(k_2 + 1) + \ell_Q n(t_2 + 1) + 4$ field elements.

Single Proof Case When $t = mn = 1$ and the prover is proving a single relation, we may choose parameters so that the protocol only uses a constant number of group elements. Set $k_1 = t_1 = 1$, $k_2 = d_P - 1$, $t_2 = d_Q - 1$. Then the protocol has communication costs of 7 group elements plus $\ell_a + \ell_P d_P + \ell_Q d_Q + 4$ field elements. This minimises communication in the case where the protocol is instantiated over a multiplicative subgroup of a finite field, where group elements are much bigger than field elements.

In the case where the protocol is instantiated using an elliptic curve group, group elements and field elements have roughly the same size. Then, we can minimise the total communication costs by choosing $k_2 = \left\lceil \sqrt{\frac{d_P}{\ell_P}} \right\rceil$, $k_1 \approx \frac{d_P}{k_2}$. Set $t_2 = \left\lceil \sqrt{\frac{d_Q}{\ell_Q}} \right\rceil$, $t_1 \approx \frac{d_Q}{t_2}$. Then the protocol has costs $\sqrt{\ell_P d_P} + \sqrt{\ell_Q d_Q} + 5$ group elements and $\ell_a + \sqrt{\ell_P d_P} + \sqrt{\ell_Q d_Q} + 4$ field elements.

Batch Proof Case When t is large, we choose parameters so that the communication costs are proportional to $\sqrt{t}$ rather than t. Set $k_2 = \left\lceil \sqrt{\frac{d_P m}{\ell_P n}} \right\rceil$, $k_1 \approx \frac{d_P m}{k_2}$. Set $t_2 = \left\lceil \sqrt{\frac{d_Q m}{\ell_Q n}} \right\rceil$, $t_1 \approx \frac{d_Q m}{t_2}$. Finally, set $m \approx \sqrt{\ell_a t}, n \approx \frac{t}{m}$. Then the protocol has communication costs of roughly $\sqrt{\ell_a t} + \sqrt{d_P \ell_P t} + \sqrt{d_Q \ell_Q t}$ group elements and $\sqrt{\ell_a t} + \sqrt{d_P \ell_P t} + \sqrt{d_Q \ell_Q t}$ field elements.

Computation The prover's computational costs are dominated by

$$O\left(\frac{\ell_a t}{\log \ell_a n} + \frac{\ell_Q n}{\log \ell_Q n} + \frac{\ell_P d_P t}{\log \ell_P n k_2} + \frac{\ell_P d_P t}{\log \ell_P n t_2}\right)$$

exponentiations. Over $\mathbb{Z}_p$, the prover must perform

$$O((\ell_a + \ell_b + \ell_P) t d_P \log m d_P + (\ell_a + \ell_b + \ell_Q) t d_Q \log m d_Q) + t d_P \mathsf{Eval}_P + t d_Q \mathsf{Eval}_Q$$

multiplications. Here, Eval_P is the cost of evaluating P once, and similarly for Q. The vectors of polynomials $\mathbf{P}^*(X), \mathbf{Q}^*(X)$ are computed using FFT techniques.

The verifier's computational costs are dominated by

$$O\left(\frac{m + \ell_a n}{\log(m + \ell_a n)} + \frac{m + \ell_Q n}{\log(m + \ell_Q n)} + \frac{k_1 + \ell_P n k_2}{\log(k_1 + \ell_P n k_2)} + \frac{t_1 + \ell_Q n t_2}{\log(t_1 + \ell_Q n t_2)}\right)$$

exponentiations. Over $\mathbb{Z}_p$, the verifier must perform

$$O((\ell_P + \ell_Q) n) + n \mathsf{Eval}_P + n \mathsf{Eval}_Q$$

multiplications.

5 Applications

In this section, we specify concrete choices of relations for $\mathbf{P}, \mathbf{Q}$, which give rise to zero-knowledge arguments for several useful applications.

5.1 Membership Argument with Public List

In membership arguments [11,10], the prover wishes to convince the verifier that a commitment contains one of the values in a given list $\mathcal{L} = (\lambda_0, \ldots, \lambda_{N-1})$. Groth and Kohlweiss [28] give an efficient membership argument, which with minor tweaks fits into our framework. For simplicity, we will in the following assume N is a power of 2.

Statement: $(c, \lambda_0, \ldots, \lambda_{N-1})$
Witness: ℓ, r such that $c = \mathrm{Com}_{ck}(\lambda_\ell; r)$
Polynomial Encoding: Let $m = \log_2 N$ and let $(l_0, \ldots, l_{m-1})$ be the binary expansion of l, satisfying $l_j(1 - l_j) = 0$ for $0 \leq j \leq m-1$. Define $l_{j,1} := l_j$ and $l_{j,0} = 1 - l_j$. We have that

$$\sum_{i=0}^{N-1} \lambda_i \prod_{j=0}^{m-1} l_{j,i_j} = \lambda_l$$

where we write the binary expansion of i as $(i_0, \ldots, i_{m-1})$.
Parameter Choice: Writing $\circ$ for the entry-wise product of two vectors

- $\ell_a = \log_2 N, \ell_b = N, \ell_P = \log_2 N, d_P = 2, \ell_Q = 1, d_Q = \log_2 N$
- $\mathbf{a} = (l_0, \ldots, l_{m-1})$
- $\mathbf{b} = (\lambda_0, \ldots, \lambda_{N-1})$
- $\mathbf{P}(\mathbf{a}, \mathbf{b}) = \mathbf{a} \circ (\mathbf{1} - \mathbf{a})$
- $Q(\mathbf{a}, \mathbf{b}) = \sum_{i=0}^{N-1} \lambda_i \prod_{j=0}^{m-1} l_{j,i_j}$

An alternative construction was given in [6] that optimises the membership argument by using an n-ary representation of l. This alternative construction is captured by our framework as follows, this time assuming for simplicity that N is a power of n, using different polynomials $\mathbf{P}$ and $\mathbf{Q}$.

Polynomial Encoding: Let $m = \log_n N$ and let $(l_0, \ldots, l_{m-1})$ be the n-ary expansion of l. Let δ_{rs} be the Kronecker delta symbol, which is equal to 1 if $r = s$ and 0 otherwise. Consider the bit-string $(\delta_{l_0,0}, \delta_{l_0,1}, \ldots, \delta_{l_{m-1},n-1})$, each element satisfying $\delta_{i,j}(1 - \delta_{i,j}) = 0$, and with $\sum_{i=0}^{n-1} \delta_{l_j,i} = 1$ for each j. As described in [6], we have that

$$\sum_{i=0}^{N-1} \lambda_i \prod_{j=0}^{m-1} \delta_{j,i_j} = \lambda_l$$

where i_j the jth n-ary digit of i.
Parameter Choice:

- $\ell_a = n \log_n N,\ \ell_b = N,\ \ell_P = n \log_n N,\ d_P = 2,\ \ell_Q = 1,\ d_Q = \log_n N$
- $\mathbf{a} = (\delta_{l_0,1}, \ldots, \delta_{l_{m-1},n-1})$, not including $\delta_{j,0}$ for any j.
- $\mathbf{b} = (\lambda_0, \ldots, \lambda_{N-1})$.
- $\delta_{l_j,0} = 1 - \sum_{i=1}^{n-1} \delta_{l_j,i}$ for each j.
- $\mathbf{v} = \big(\delta_{l_0,0}, \ldots, \delta_{l_{m-1},n-1}\big)$, with the $\delta_{j,0}$ included.

- $P(\mathbf{a}, \mathbf{b}) = \mathbf{v} \circ (\mathbf{1} - \mathbf{v})$
- $Q(\mathbf{a}, \mathbf{b}) = \sum_{i=0}^{N-1} \lambda_i \prod_{j=0}^{m-1} \delta_{j,i_j} = \lambda_l$

When $t = 1$ and we are aiming for a constant number of group elements, the simple binary version of the argument gives the lowest communication costs. Otherwise, in the cases where t is large, or where $t = 1$ and we aim to minimise the total number of elements communicated, setting $n = 3$ gives the lowest communication costs. The protocol efficiency is reported in Table 1.

5.2 Polynomial Evaluation Argument

In a polynomial evaluation argument [23,10], we have a polynomial of degree N and commitments to a point and its purported evaluation in that point. The prover wants to convince the verifier that the committed evaluation of the polynomial is correct.

The most efficient discrete logarithm based polynomial evaluation argument was given by Bayer and Groth [2]. We will now use our framework of polynomial relations to capture their protocol.

Statement: $(c_u, c_v, h(X))$, where $h(X)$ is a polynomial of degree N.

Witness: u, η, v, ν such that $c_u = \mathrm{Com}_{ck}(u; \eta), c_v = \mathrm{Com}_{ck}(v, \nu)$, and $h(u) = v$.

Polynomial Encoding: Set $u_i = u^{2^i}$ for $0 \leq i \leq \log_2 N - 1$, so that $u_i = u_{i-1}^2$ for each i. If $h(X) = \sum_{i=0}^{N-1} h_i X^i$, then we can write $h(u) = \sum_{i=0}^{N-1} h_i \prod_{j=0}^{\log_2 N-1} u_j^{i_j}$.

Parameter Choice:

- $\ell_a = \log_2 N$, $\ell_b = N$, $\ell_P = \log_2 N - 1$, $d_P = 2$, $\ell_Q = 1$, $d_Q = \log_2 N$
- $\mathbf{a} = (u_0, \ldots, u_{\log N-1})$
- $\mathbf{b} = (h_0, \ldots, h_{N-1})$
- $\mathbf{P}(\mathbf{a}, \mathbf{b}) = (u_1 - u_0^2, \ldots, u_{\log N-1} - u_{\log N-2}^2)$
- $\mathbf{Q}(\mathbf{a}, \mathbf{b}) = \sum_{i=0}^{N-1} h_i \prod_{j=0}^{\log N-1} u_j^{i_j}$

With alternative choices of the matrices $\mathbf{P}, \mathbf{Q}$, we can improve the communication costs of their argument by switching to an n-ary encoding of the powers in the polynomial.

Polynomial Encoding: Set $u_i = u^{n^i}$ for $0 \leq i \leq \log_n N - 1$, so that $u_i = u_{i-1}^n$ for each i. If $h(X) = \sum_{i=0}^{N-1} h_i X^i$, then we can write $h(u) = \sum_{i=0}^{N-1} h_i \prod_{j=0}^{\log_n N-1} u_j^{i_j}$, where this time, i_j is the jth digit of the $n$ary representation of i. This gives rise to the efficiencies listed in Table 1.

Parameter Choice:

- $\ell_a = \log_n N$, $\ell_b = N$, $\ell_P = \log_n N$, $d_P = n$, $\ell_Q = 1$, $d_Q = \log_n N$
- $\mathbf{a} = (u_0, \ldots, u_{\log_n N-1})$
- $\mathbf{b} = (h_0, \ldots, h_{N-1})$
- $\mathbf{P}(\mathbf{a}, \mathbf{b}) = (u_1 - u_0^n, \ldots, u_{\log_n N-1} - u_{\log_n N-2}^n)$
- $\mathbf{Q}(\mathbf{a}, \mathbf{b}) = \sum_{i=0}^{N-1} h_i \prod_{j=0}^{\log_n N-1} u_j^{i_j}$

When $t = 1$ and we are aiming for a constant number of group elements, setting $n = 4$ gives the lowest communication costs. When $t = 1$ and we aim to minimise the total number of elements communicated, we set $n = \frac{\log_2 N}{\log_2 \log_2 N}$. Otherwise, in the cases where t is large, setting $n = 6$ gives the lowest communication costs. The protocol efficiency is reported in Table 1.

We note that [1] gives a batch argument for polynomial evaluation based on similar ideas. However, ours is more communication efficient.

Remark. The relations above arise from choices of a small set of powers of u which generate all powers from u to u^{N-1}. This is the same as choosing an additive basis for $[N-1]$. For certain parameter choices, we have found modest benefits to using more complex bases, such as generalised Zeckendorf bases, but these give only slight improvements, so are omitted for simplicity.

5.3 Range Proof

In range proofs [9,38], we have a commitment and a range $[A; B]$. The prover wants to convince the verifier that the committed value inside the commitment falls in the given range. A common strategy for constructing a range proof is to write the committed value in binary, prove all the bits are indeed 0 or 1, and that their weighted sum yields a number within the range. We now describe this type of range proof in our framework of polynomial relations, where we for simplicity focus on intervals $[0, N]$ with $N = 2^m - 1$.

Statement: (N, c)
Witness: a, r such that $c = \mathrm{Com}_{ck}(a; r), a \in [0, N]$.
Polynomial Encoding: Let $a_0, \ldots, a_{m-1}$ be the binary representation of a, so that $a_i(1 - a_i) = 0$ for $0 \leq i \leq m-1$. Then $a = \sum_{i=0}^{m-1} a_i 2^i$.
Parameter Choice:

- $\ell_a = m, \ell_b = m, \ell_P = m, d_P = 2, \ell_Q = 1, d_Q = m + 1$
- $\mathbf{a} = (a_0, \ldots, a_{m-1})$
- $\mathbf{b} = (2^0, 2^1, \ldots, 2^{m-1})$
- $\mathbf{P}(\mathbf{a}, \mathbf{b}) = \mathbf{a} \circ (\mathbf{1} - \mathbf{a})$
- $\mathbf{Q}(\mathbf{a}, \mathbf{b}) = \sum_{i=0}^{m-1} a_i 2^i$

With an alternative choice of $\mathbf{P}, \mathbf{Q}$, following [16], it is possible to improve the communication costs of the argument by using an n-ary base. This gives rise to the efficiencies listed in Table 1.

Polynomial Encoding: Let $N = n^m - 1$. Let $a_0, \ldots, a_{m-1}$ be the n-ary representation of a, so that $\prod_{k=0}^{n-1}(a_i - k) = 0$ for $0 \leq i \leq m-1$. Then $a = \sum_{i=0}^{m-1} a_i n^i$.
Parameter Choice:

- $\ell_a = m$, $\ell_b = m$, $\ell_P = m$, $d_P = n$, $\ell_Q = 1$, $d_Q = 1$
- $\mathbf{a} = (a_0, \ldots, a_{m-1})$
- $\mathbf{b} = (1, n, \ldots, n^{m-1})$

- $P(\mathbf{a}, \mathbf{b}) = \mathbf{a} \circ (\mathbf{a} - \mathbf{1}) \circ \ldots (\mathbf{a} - n + 1)$
- $Q(\mathbf{a}, \mathbf{b}) = \sum_{i=0}^{m-1} a_i n^i$

When $t = 1$ and we are aiming for a constant number of group elements, setting $n = 4$ gives the lowest communication costs. When $t = 1$ and we aim to minimise the total number of elements communicated, we set $n = \frac{\log_2 N}{\log_2 \log_2 N}$. Otherwise, in the cases where t is large, setting $n = 6$ gives the lowest communication costs. The protocol efficiency is reported in Table 1.

6 Conclusion

We have provided zero-knowledge arguments for simple polynomial relations, relying solely on the discrete logarithm assumption. When we only have one instance of the argument, $t = 1$, the single value membership arguments and polynomial evaluation arguments compiled within our framework improve on the state of the art both asymptotically and for practical parameters. When there are many instances, $t > 1$, we have a batch argument for polynomial relations, which is significantly more efficient than the naïve solution of repeating single instance arguments many times.

References

1. Stephanie Bayer. *Practical zero-knowledge Protocols based on the discrete logarithm Assumption*. PhD thesis, University College London, 2014.
2. Stephanie Bayer and Jens Groth. Zero-Knowledge Argument for Polynomial Evaluation with Application to Blacklists. In *EUROCRYPT*, pages 646–663, 2013.
3. Mihir Bellare, Juan A. Garay, and Tal Rabin. Batch Verification with Applications to Cryptography and Checking. In *EUROCRYPT*, pages 236–250, 1998.
4. Josh Benaloh and Michael de Mare. One-way accumulators: A decentralized alternative to digital signatures. *Advances in CryptologyEUROCRYPT'93*, 1994.
5. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, and Jens Groth. Foundations of Fully Dynamic Group Signatures. In *ACNS*, pages 117–136, 2016.
6. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, Jens Groth, and Christophe Petit. Short Accountable Ring Signatures Based on DDH. In *ESORICS*, pages 243–265, 2013.
7. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In *EUROCRYPT*, pages 327–357, 2016.
8. Jonathan Bootle, Andrea Cerulli, Essam Ghadafi, Jens Groth, Mohammad Hajiabadi, and Sune K. Jakobsen. Linear-time zero-knowledge proofs for arithmetic circuit satisfiability. Cryptology ePrint Archive, Report 2017/872, 2017. `http://eprint.iacr.org/2017/872`.
9. Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In *EUROCRYPT*, pages 431–444, 2002.
10. Stefan Brands, Lisa Demuynck, and Bart De Decker. A practical system for globally revoking the unlinkable pseudonyms of unknown users. In *ACISP*, pages 400–415, 2007.

11. Emmanuel Bresson and Jacques Stern. Efficient revocation in group signatures. In *PKC*, pages 190–206, 2001.
12. Jan Camenisch and Rafik Chaabouni. Efficient protocols for set membership and range proofs. *Advances in Cryptology-ASIACRYPT . . .*, 2008.
13. Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. *Public Key CryptographyPKC . . .*, 2009.
14. Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *CRYPTO*, pages 61–76, 2002.
15. Jan Camenisch and Markus Stadler. Proof systems for general statements about discrete logarithms. Technical Report 260, ETH Zurich, 1997.
16. Rafik Chaabouni, Helger Lipmaa, and Abhi Shelat. Additive combinatorics and discrete logarithm based range protocols. In *ACISP*, volume LNCS 6168, pages 336–351, 2010.
17. Geoffroy Couteau, Thomas Peters, and David Pointcheval. Removing the strong RSA assumption from arguments over the integers. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II*, pages 321–350, 2017.
18. Ronald Cramer and Ivan Damgård. Zero-knowledge proofs for finite field arithmetic, or: Can zero-knowledge be for free? In *CRYPTO*, pages 424–441, 1998.
19. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. *Advances in Cryptology . . .*, 839:174–187, 1994.
20. Ivan Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *EUROCRYPT*, pages 418–430, 2000.
21. Ivan Damgård and Nikos Triandopoulos. Supporting Non-membership Proofs with Bilinear-map Accumulators. IACR ePrint archive report 538, 2008.
22. Prastudy Fauzi, Helger Lipmaa, and Bingsheng Zhang. Efficient Non-Interactive Zero Knowledge Arguments for Set Operations. In *Financial Cryptography and Data Security*, pages 216–233, 2014.
23. Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *CRYPTO*, pages 16–30, 1997.
24. Juan A. Garay, Philip MacKenzie, and Ke Yang. Strengthening zero-knowledge protocols using signatures. *Journal of Cryptology*, 2006.
25. Roario Gennaro, Darren Leigh, Ravi Sundaram, and William Yerazunis. Batching Schnorr identification scheme with applications to privacy-preserving authorization and low-bandwidth communication devices. In *ASIACRYPT*, volume LNCS 3329, pages 276–292, 2004.
26. Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic Span Programs and Succinct NIZKs without PCPs. In *EUROCRYPT*, pages 626–645, 2013.
27. Jens Groth. *Honest verifier zero-knowledge arguments applied*. PhD thesis, Aarhus University, 2004.
28. Jens Groth and Markulf Kohlweiss. One-out-of-Many Proofs: Or How to Leak a Secret and Spend a Coin. In *EUROCRYPT*, pages 253–280, 2015.
29. Ryan Henry and Ian Goldberg. Batch proofs of partial knowledge. In *ACNS*, pages 502–517, 2013.
30. Javier Herranz. Attribute-based versions of schnorr and elgamal. *Appl. Algebra Eng. Commun. Comput.*, 27(1):17–57, 2016.

31. Charanjit Jutla and Arnab Roy. Shorter {Q}uasi-{A}daptive {NIZK} {P}roofs for {L}inear {S}ubspaces. In *ASIACRYPT*, volume LNCS 8269, pages 1–20, 2013.
32. Charanjit S. Jutla and Arnab Roy. Switching lemma for bilinear tests and constant-size NIZK proofs for linear subspaces. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8617 LNCS(PART 2):295–312, 2014.
33. Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, pages 177–194, 2010.
34. Joe Kilian. A note on efficient zero-knowledge proofs and arguments. In *STOC*, pages 723–732, 1992.
35. Eike Kiltz and Hoeteck Wee. Quasi-adaptive NIZK for linear subspaces revisited. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9057(339563):101–128, 2015.
36. Jiangtao Li, Ninghui Li, and Rui Xue. Universal Accumulators with Efficient Nonmembership Proofs. *Proceedings of the 5th international conference on Applied Cryptography and Network Security (ACNS)*, pages 253–269, 2007.
37. Benoît Libert, Somindu C. Ramanna, and Moti Yung. Functional commitment schemes: From polynomial commitments to pairing-based accumulators from simple assumptions. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 30:1–30:14, 2016.
38. Helger Lipmaa. On diophantine complexity and statistical zero-knowledge arguments. In *ASIACRYPT*, pages 398–415, 2003.
39. Silvio Micali, Michael O. Rabin, and Joe Kilian. Zero-knowledge sets. In *FOCS*, pages 80–91, 2003.
40. Lan Nguyen. Accumulators from bilinear pairings and applications to ID-based ring signatures and group membership revocation. In *CT-RSA*, pages 275–292, 2005.
41. Charalampos Papamanthou, Elaine Shi, and Roberto Tamassia. Signatures of correct computation. In *Theory of Cryptography - 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*, pages 222–242, 2013.
42. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO*, pages 129–140, 1991.
43. Kun Peng. A general, flexible and efficient proof of inclusion and exclusion. *Trusted Systems*, pages 33–48, 2012.
44. Kun Peng and Feng Bao. Batch ZK Proof and Verification of OR Logic. In *Inscrypt*, volume LNCS 5487, pages 141–156, 2008.
45. Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.
46. Yupeng Zhang, Daniel Genkin, Jonathan Katz, Dimitrios Papadopoulos, and Charalampos Papamanthou. vsql: Verifying arbitrary SQL queries over dynamic outsourced databases. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 863–880, 2017.