Efficient zero-knowledge arguments in the discrete log setting, revisited (Full version)

Max Hoffmann[1], Michael Klooß[2], and Andy Rupp[2]

[1]Ruhr-University Bochum, max.hoffmann@rub.de
[2]Karlsruhe Institute for Technology, {michael.klooss, andy.rupp}@kit.edu

October 25, 2019

Abstract

This work revisits zero-knowledge proofs in the discrete logarithm setting. First, we identify and carve out basic techniques (partly being used implicitly before) to optimise proofs in this setting. In particular, the *linear combination of protocols* is a useful tool to obtain zero-knowledge and/or reduce communication. With these techniques, we are able to devise zero-knowledge variants of the logarithmic communication arguments by Bootle et al. (EUROCRYPT '16) and Bünz et al. (S&P '18) thereby introducing almost no overhead. We then construct a conceptually simple commit-and-prove argument for satisfiability of a set of *quadratic equations*. Unlike previous work, we are not restricted to rank 1 constraint systems (R1CS). This is, to the best of our knowledge, the first work demonstrating that general quadratic constraints, not just R1CS, are a natural relation in the dlog (or ideal linear commitment) setting. This enables new possibilities for optimisation, as, e.g., *any* degree n^2 polynomial $f(X)$ can now be "evaluated" with at most $2n$ quadratic constraints.

Additionally, we take a closer look at quantitative measures, e.g. the efficiency of an extractor. For this, we formalise *short-circuit extraction*, which allows us to give tighter bounds on the efficiency of an extractor.

1 Introduction

Zero-knowledge arguments (of knowledge) (ZKAoK) allow a party $\mathcal{P}$, the prover, to convince another party $\mathcal{V}$, the verifier, of the truth of a statement (and knowledge of a witness) without revealing any other information. For example, one may prove knowledge of a valid signature on some message, without revealing the signature. The ability to ensure *correctness* without compromising *privacy* makes zero-knowledge arguments a powerful tool, which is ubiquitous in theory and application of cryptography. Since the first *practical* construction of succinct non-interactive arguments of knowledge (SNARK) [25], and their application to Blockchain and related areas, research in theory and applications of efficient ZKAoKs has progressed significantly, see the works [2, 8, 13, 17, 21, 24, 25, 26, 46] to name a few.

In this paper, we revisit a line of works [13, 16, 28] in the setting of groups of prime order. From an abstract point of view, in terms of [33], one part of our work is in the world of ideal linear commitments (ILC). That is, our verifier can do "matrix-vector queries" on a committed value $\boldsymbol{w}$, e.g. request an opening for a matrix-vector product $\boldsymbol{\Gamma w}$. A priori, this is more powerful than other settings like PCP or IOP, where the verifier's queries are restricted to point or inner-product queries[33]. Nonetheless, the ILC-arguments in [13, 16, 28] only work for the language R1CS "natively", which is also covered by more restricted verifiers. We show that with ILC, one can directly handle *systems of quadratic equations*, of which R1CS is a special case.

However, another part of this work treats proofs of knowledge of preimages of group homomorphisms. For example, one can prove knowledge of the decryption of an ElGamal ciphertext like this. This does not fit into the ILC setting, hence we do not use the ILC abstractions.

1.1 Basic techniques

We identify and present basic design principles which underly most existing works on efficient zero-knowledge arguments in the group setting.

In the following, we use implicit representation for group elements, see Section 2. Let us recall (a slight variant of) the standard Σ-Protocol (Σ_{std}) for proving knowledge of a preimage $\boldsymbol{w}$ for $[\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$ for $[\boldsymbol{A}] \in \mathbb{G}^{m \times n}$. This proof covers a large class of statements, including dlog relations, knowing the opening of a commitment, etc. The protocol works as follows:

- Prover: Pick $\boldsymbol{r} \leftarrow \mathbb{F}_p^n$, let $[\boldsymbol{a}] := [\boldsymbol{A}]\boldsymbol{r}$, send $[\boldsymbol{a}]$.
- Verifier: Pick and send $\boldsymbol{x} = (x_1, x_2) \leftarrow \mathbb{F}_p^2$ (with $x_2 \neq 0$).
- Prover: Send $\boldsymbol{z} := x_1\boldsymbol{w} + x_2\boldsymbol{r}$.
- Verifier: Accept iff $[\boldsymbol{A}]\boldsymbol{z} = x_1[\boldsymbol{t}] + x_2[\boldsymbol{a}]$.

Intuitively, this is zero-knowledge since $\boldsymbol{r}$ completely masks $\boldsymbol{w}$ in $\boldsymbol{z} = x_1\boldsymbol{w} + x_2\boldsymbol{r}$ (since $x_2 \neq 0$), and finding $\boldsymbol{r}$ from $[\boldsymbol{a}]$ is hard. It is extractable, since two linearly independent challenges $\boldsymbol{x}_1, \boldsymbol{x}_2$ with answers $\boldsymbol{z}_1, \boldsymbol{z}_2$ (for fixed $[\boldsymbol{a}]$) allow to reconstruct $\boldsymbol{w}, \boldsymbol{r}$. But Protocol Σ_{std} is not particularly communication-efficient, as it sends the full masked witness $\boldsymbol{z} \in \mathbb{F}_p^n$ as well as $[\boldsymbol{a}] \in \mathbb{G}^m$. Using probabilistic verification, one can often improve this.

1.1.1 Probabilistic verification.

The underpinning of *efficient* arguments of knowledge (without zero-knowledge) is probabilistic verification of the claim. For instance, instead of verifying $[\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$ directly, the verifier could send a random $y \leftarrow \mathbb{F}_p$. Both parties compute $\boldsymbol{y} = (y^i)_i \in \mathbb{F}_p^m$ and prove (resp. verify) $[\widehat{\boldsymbol{A}}]\boldsymbol{w} = [\widehat{t}]$ for $[\widehat{\boldsymbol{A}}] = \boldsymbol{y}^\top[\boldsymbol{A}] \in \mathbb{G}^{1 \times n}$ and $[\widehat{t}] = \boldsymbol{y}^\top[\boldsymbol{t}] \in \mathbb{G}$ instead. This would result in a communication complexity which is independent of m as $[\widehat{\boldsymbol{a}}] = [\widehat{\boldsymbol{A}}]\boldsymbol{r} \in \mathbb{G}$.

Not all probabilistic verifications are alike. To work well with zero-knowledge, we need "suitable" verification procedures, so that techniques to attain zero-knowledge can be applied. This essentially means that the verification should be *linear*, i.e. all tested equations should be linear. (Abstract groups only allow linear operations anyway.)

We define so-called *testing distributions* which are distributions over $\mathbb{F}_p^n$, yielding "random linear test maps". Given enough independent test maps and images, one can recover the "tested object". This allows to extract knowledge. Our definitions are tailored to our setting. See [48] for a possible generalisation.

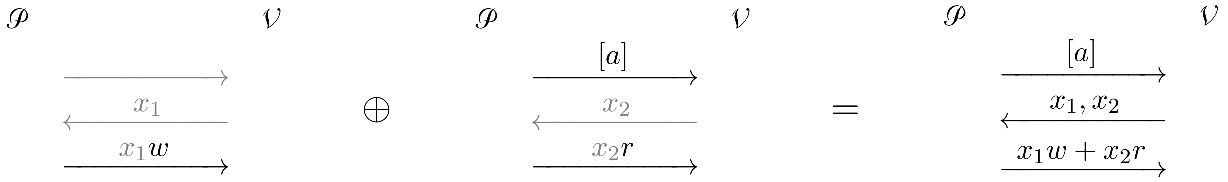

Figure 1: Linear Combination of Protocols. *Left:* The trivial proof of knowledge: Send the witness. *Middle:* Send a random statement. Then send the witness. *Grayed out:* Terms for linear combination. *Right:* The linear combination with verifier's randomness.

1.1.2 Linear combinations of protocols.

A core insight for achieving zero-knowledge (and reducing communication) in our setting *efficiently* is that protocols can often be linearly combined, see Fig. 1 for an illustration. This exploits the *linearity* of the computations and checks of verifier and prover in each round. By running an "umasked *non*-zero-knowledge argument" (Fig. 1, left) and linearly combining it with an argument for a "masking randomness" (middle), one can achieve zero-knowledge (right). All of our zero-knowledge compilations rely on this strategy. We typically consider *random* linear combinations of protocols, where the verifier picks the randomness (x_1, x_2 in Fig. 1), as this often achieves extractability. In fact, this kind of linear combination recovers the batch proofs of [44], see Appendix B. Non-randomised linear combinations are also used, e.g. Protocols 4.1 and 3.14 or [16].

1.1.3 Uniform(-or-unique) responses.

In our setting, for simulation it is typically enough to ensure that the prover's messages are distributed uniformly at random. More concretely, the responses should be either uniformly distributed (conditioned on all *later* messages, *not* previous messages), such as $\boldsymbol{z}$ in Protocol Σ_{std}. Or they should be uniquely determined and *efficiently* computable from the challenges and all *later* messages, such as $[\boldsymbol{a}]$ in Protocol Σ_{std}. This allows to construct a trivial simulator, which constructs the transcript *in reverse*: Starting with the final messages, and working its way towards the beginning, the simulator picks the uniformly distributed messages itself, and then computes the uniquely determined ones. All simulators in this paper work like this.

1.1.4 Kernels and redundancy.

Many interesting statements are non-linear. For example, for polynomial commitments [12], we want to show that $[\boldsymbol{c}] \in \mathbb{G}^m$ is a commitment to a polynomial $f \in \mathbb{F}_p[X]$ (of degree at most $d-1$) and $f(x) = t$, where $x \in \mathbb{F}_p$ is a random challenge. Naively, one commits to the coefficients of the polynomial with monomial basis X^i for $i = 0, \dots, n-1$. Suppose we have a (linear) protocol which proves $f(x) = t$. We could hope that running a random linear combination as in Fig. 1 should give us uniform-or-unique responses (and hence zero-knowledge). However, we are in a predicament: For random $g \in \mathbb{F}_p[X]$, we have $(f+g)(x) \neq f(x)$ and thus we have to let $\mathcal{V}$ know $y = g(x)$ somehow. To ensure the prover does not send arbitrary y, we have to rely on a proof again! But if this proof leaks (too much) information,[1] we cannot use it to randomise the response. We can escape by having a way to randomise *without changing the statement.* In other words, we need some g with $g(x) = 0$ for all $x \in \mathbb{F}_p$. Clearly, that means $g = 0$, and there's nothing random anymore! Another dead end.

One solution is to add *redundancy*, which does not "influence" soundness: Here, we artificially create a non-trivial kernel of the "evaluate at x"-map. We can do so by representing $f(X)$ as $\sum_i (\alpha_i + \beta_i) X^i$ and commit to all α_i and β_i. Now we can mask with $g(X)$ where $\alpha_i \leftarrow \mathbb{F}_p$ and $\beta_i = -\alpha_i$. Thus, we successfully injected randomness into the response. Generally, adding just enough redundancy to achieve uniformly random responses is our goal.

1.1.5 Composition of arguments systems.

For completeness, we recall that, by committing to (intermediate) results, and sharing these commitments in multiple argument systems, one can combine the most efficient arguments for each task.

Example 1.1. In our logarithmic communication zero-knowledge inner product argument $\mathsf{IPA}_{\text{almZK}}$ for $\exists \boldsymbol{x}, \boldsymbol{y} \colon \langle \boldsymbol{x}, \boldsymbol{y} \rangle = t$, we randomise as $\langle \boldsymbol{x} + \boldsymbol{r}, \boldsymbol{y} + \boldsymbol{s} \rangle = t$ so that $\langle \boldsymbol{r}, \boldsymbol{y} \rangle = \langle \boldsymbol{r}, \boldsymbol{s} \rangle = \langle \boldsymbol{x}, \boldsymbol{s} \rangle = 0$ with only logarithmically many (specially chosen) random components in $\boldsymbol{r}, \boldsymbol{s}$. This is an application of the "redundancy/kernel" technique. The "uniform-or-unique" guideline ensures that it is enough that each response is random. Hence a logarithmic number of (well-chosen) random components in $\boldsymbol{r}$, $\boldsymbol{s}$ does suffice.

On the other hand, our logarithmic communication linear map preimage argument $\mathsf{LMPA}_{\mathsf{ZK}}$ for $\exists \boldsymbol{w} \colon [\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$ uses a linear combination of a *non*-zero-knowledge argument for $[\boldsymbol{A}]$, plus a similar argument for a *different* $[\boldsymbol{A}]$ and $[\boldsymbol{t}]$ (of the same size). Finally, for our logarithmic communication shuffle argument Π_{shuffle} (Appendix C), we compose $\mathsf{QESA}_{\mathsf{ZK}}$ (our quadratic equation argument) and $\mathsf{LMPA}_{\mathsf{ZK}}$ by sharing a commitment to the witness.

1.2 Contribution

To the best of our knowledge, there is no work which presents these techniques, in particular linear combination of protocols, as *unifying* guidelines. Implicitly, these techniques are used in many

[1] If it only leaks a little bit, then a linear combination with many g_i may work.

works [12, 13, 16, 28, 44]. We follow the above guidelines for constructing and explaining our zero-knowledge arguments.

1.2.1 Linear map preimage argument (LMPA).

We give in two steps an argument for $\exists \boldsymbol{w}\colon [\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$ for $[\boldsymbol{A}] \in \mathbb{G}^{m\times n}$ with communication $O(\log(n))$. The idea is to first use batch verification. Essentially, $\mathsf{LMPA}_{\mathrm{batch}}$ multiplies the equation with a random vector $\boldsymbol{y} \in \mathbb{F}_p^m$ from the left to obtain $[\widehat{\boldsymbol{A}}] = \boldsymbol{y}^\top[\boldsymbol{A}] \in \mathbb{G}^{1\times n}$ and $[\widehat{t}] = \boldsymbol{y}^\top[\boldsymbol{t}] \in \mathbb{G}$. Thus, communication is independent of m. Now, we prove $\exists \boldsymbol{w}\colon [\widehat{\boldsymbol{A}}]\boldsymbol{w} = [\widehat{t}]$ using $\mathsf{LMPA}_{\mathsf{ZK}}$. Protocol $\mathsf{LMPA}_{\mathsf{ZK}}$ is derived from [13]. It is enhanced with zero-knowledge at the cost of constant communication overhead and logarithmic computational overhead (in n).

1.2.2 Quadratic equation commit-and-prove.

First of all, we derive a (almost) zero-knowledge inner product argument $\mathsf{IPA}_{\mathrm{almZK}}$ from [13, 16], again with constant communication and logarithmic computational overhead compared to [13, 16]. From $\mathsf{IPA}_{\mathrm{almZK}}$ we obtain an argument for proving $\exists \boldsymbol{w}\colon \forall i\colon \langle \boldsymbol{w}, \boldsymbol{\Gamma}_i\boldsymbol{w}\rangle = 0$, where $\boldsymbol{\Gamma}_i \in \mathbb{F}_p^{n\times n}$ and $\boldsymbol{w}$ is committed to. For efficiency, we carry out a batch proof, i.e. we prove $\langle \boldsymbol{w}, \boldsymbol{\Gamma}\boldsymbol{w}\rangle$ with $\boldsymbol{\Gamma} := \sum_i r_i\boldsymbol{\Gamma}_i$ for random $r_i \in \mathbb{F}_p$. The resulting argument, $\mathsf{QESA}_{\mathsf{ZK}}$ for short, is "adaptive commit-and-prove", i.e. the statement $\boldsymbol{\Gamma}_i$ may be chosen after the commitment to $\boldsymbol{w}$.

The commit-and-prove system $\mathsf{QESA}_{\mathsf{ZK}}$ is conceptually simple and can be efficiently combined with other arguments. We leave as an open question whether its strategies can be adapted by linear IOPs or whether they are unique to ILC.

1.2.3 Sets of quadratic equations.

Being able to prove arbitrary quadratic equations instead of R1CS equations, i.e. equations $(\sum a_i x_i)(\sum b_i x_i) + \sum c_i x_i = 0$, gives much flexibility. To the best of our knowledge, expressing the quadratic equation $\langle \boldsymbol{x}, \boldsymbol{x}\rangle = \sum x_i^2 = t$ as R1CS requires n equations: $y_i = x_i^2$ $(i = 1, \dots, n-1)$ and $x_n^2 = t - \sum_i y_i$, where y_i are additionally introduced variables. Requiring n equations is surprising for [13, 16] which build on an *inner product argument*. Obviously, $\mathsf{QESA}_{\mathsf{ZK}}$ needs one (quadratic) equation to express $\langle \boldsymbol{x}, \boldsymbol{x}\rangle = t$.

Using general quadratic equations, one can evaluate any (univariate) polynomial $f(X) = \sum_{i=0}^{d^2-1} a_i X^i$ of degree $d^2 - 1$ with $2d$ equations and intermediate variables. Concretely, let $y_i = x^i = y_{i-1}x$, $z_i = x^{di} = z_1 z_{i-1}$, for $i = 2, \dots d-1$ and $z_1 = y_{d-1}x$ and $z_0 = 1$. Then $f(x) = \sum_{i,j=0}^{d} a_{i+jd} y_i z_j$. Using this, one can speed up "table lookups", which are typically encoded as polynomial evaluation. We are compelled to note that using composition of protocols, more efficient (batch) subproofs for polynomial evaluation may be possible.

For S(N)ARK-friendly cryptography [36], supporting quadratic equations is very useful. Matrix-vector multipliciations are efficient even when both matrix and vector are *secret*. "Embedding" an elliptic curve (see Jubjub [47]), is also more efficient than for R1CS. For general point addition in a (twisted) Edwards curve, we need 5 instead of 8 constraints per bit.

1.2.4 Correctness of a shuffle.

By instantiating the shuffle proof of Bayer and Groth [5] with $\mathsf{LMPA}_{\mathsf{ZK}}$ and $\mathsf{QESA}_{\mathsf{ZK}}$ as subprotocols, we obtain an argument Π_{shuffle} for correctness of a shuffle (of ElGamal ciphertexts). To the best of our knowledge,[a] this is the first efficient argument with proof size $O(\log(N))$. Our computational efficiency is comparable to [5], which has proof size $O(\sqrt{N})$. More concretely, we (very roughly) estimate at worst twice the computation.

[a] Addendum: We note that (concretely less efficient) shuffles of *commitments* are present in [16], and we have been informed of an improvement similar to ours, see Remark C.1 or [3]. Our protocol works in the setting of [5] with ElGamal *ciphertexts*.

1.2.5 Knowledge errors, tightness and short-circuit extraction.

From a quantitative perspective, our notion of testing distributions[2] and their soundness errors, are useful to separate study of knowledge errors and extraction in the setting of special soundness. Testing distributions have associated soundness errors, which (up to technical difficulties we state as open problems) translate to knowledge errors of the protocol. Explicit knowledge errors achieve tunable levels of soundness, e.g. 2^{-120} instead of 2^{-256}, which impacts runtime positively.

Short-circuit extraction. We give a definition of *short-circuit extraction.* This treats extraction assertions such as "Ext either finds a witness or it solves a hard problem". It formalises the (common) behaviour of an extractor to either find a witness with *few* transcripts, or solve the hard problem (e.g. equivocating a commitment). Without distinguishing these cases, the bounds on the necessary number of transcripts for extraction is much higher. For example, we show that the extractor for the $\mathsf{LMPA}_{\mathsf{ZK}}$ and $\mathsf{IPA}_{\mathrm{almZK}}$ (and also [13, 16]) needs a tree of transcripts of size $O(\log(n)n)$ in the worst case. For $\mathsf{QESA}_{\mathsf{ZK}}$, extracting a proof for N quadratic equations in n variables requires $O(\log(n)nN)$ transcripts.[3] The extractor in [16] needs $O(n^3N)$ transcripts, which for $n, N \approx 2^{20}$ implies a security loss of $\approx 2^{80}$ instead of $\approx 2^{45}$.

In Appendix D, we give a conjectured relation between communication efficiency and extraction efficiency, which implies that extraction from $O(\frac{n}{\log(n)})$ transcripts would be optimal. We also elaborate on a loophole in above security estimates, namely how to *efficiently obtain* the transcripts.

1.2.6 Dual testing distributions.

Dual testing distributions are a technical tool which allow us to sample a "new" commitment key from a given one, such that knowledge (e.g. commitment opening) cannot be transferred. This turns out to be more communication efficient than letting the verifier send a new commitment key. To the best of our knowledge, this is a new technique.

1.2.7 Efficiency and comparison to [16].

In Table 1, we compare our argument systems with related work in the group setting. In Table 2, we give precise efficiency measures for $\mathsf{LMPA}_{\mathsf{ZK}}$ and $\mathsf{QESA}_{\mathsf{ZK}}$. In any case, $n = |\boldsymbol{w}|$ is the size of the witness $\boldsymbol{w} \in \mathbb{F}_p^n$. Since it is statement dependent, we ignore that QE is more powerful than R1CS, possibly allowing smaller witness size (as seen in the example $\langle \boldsymbol{x}, \boldsymbol{x} \rangle = t$ above). Since statement size N is typically a small mulitple of witness size, we ignore its influence. In Table 2, we omit the verifier's computation, since after optimisations [16], both are almost identical.[4] Generally, optimisations applicable to [16] are applicable to our protocols as well. For the prover, we do not optimise (e.g. we use no multi-exponentiations), and are not aware of non-generic optimisation. Although $\mathsf{QESA}_{\mathsf{ZK}}$ covers general quadratic equations, it compares favorably to Bulletproofs [16] which only cover R1CS. In Section 5, we compare our implementations of (aggregate) range proofs.

1.2.8 Comparison with other proof systems.

It is hard to make a fair comparison of proof systems. There are many relevant parameters, such as setup, assumptions, quantum resistance, native languages, etc. beyond mere proof size and performance. See Section 1.3 for a high-level discussion. To draw (non-trivial) conclusions from comparisons on an implementation level, one should compare fully optimised implementations. Thus, we restrict ourselves to a comparison with Bulletproofs (which we reimplemented with the same optimisation level as our proof systems). For somewhat concrete numbers regarding (implementation) performance, as well as other factors relevant to the comparison of proof systems, we refer to [8, Figure 2]. Our proof systems are similar enough to Bulletproofs for these comparisons to still hold.

[2] We concede that the notion of "testing distribution" is too narrow, one should work with more general definitions.

[3] The *average* number of necessary transcripts could be as low as $O(n)$.

[4] For completness are $4n$ resp. $6n$ exponentiations for $\mathsf{QESA}_{\mathsf{ZK}}$ resp. [16]. The verifier in $\mathsf{LMPA}_{\mathsf{ZK}}$ needs $\approx mn$ group exponentiations.

	Setup	Ass.	Moves	Comm.	Comp. $\mathscr{P}$	Comp. $\mathscr{V}$	Nat. $\mathscr{R}$
SNARG [25]	✗	KoE	1	$O(1)$	$O(n)$	$\leq \lvert\boldsymbol{w}\rvert$	R1CS
Bulletproofs[16]	✓	dlog	$O(\log(n))$	$O(\log(n))$	$O(n)$	$\lvert\boldsymbol{w}\rvert$	R1CS
This work	✓	dlog	$O(\log(n))$	$O(\log(n))$	$O(n)$	$\lvert\boldsymbol{w}\rvert$	QE

Table 1: **Setup:** Is a common *random* string sufficient? **Ass(umption):** Underlying security assumption. Knowledge of exponents (KoE); Hardness of dlogs. **Moves:** The number of messages sent. **Comm(unication):** The number of group elements sent. **Comp:** Computation of $\mathscr{P}$ resp. $\mathscr{V}$ in number of exponentiations. **Nat(ive)** $\mathscr{R}$**:** "Native" relation proven.

	Comm. $\mathbb{G}$	Comm. $\mathbb{F}_p$	Comp. $\mathscr{P}$	$\mathscr{R}$
$\mathsf{LMPA}_{\mathrm{ZK}}$	$\approx 2km\log_k(n)$	$2km$	$\approx (k+2)mn$	LMP
$\mathsf{QESA}_{\mathrm{ZK}}(k=2)$	$2\lceil\log(n+2)\rceil + 3$	2	$\approx 8n$	QE
Bulletproofs[16]	$2\lceil\log(n)\rceil + 8$	5	$\approx 12n$	R1CS

Table 2: Detailed comparisons in terms of group operations. By "$\approx$" we denote upper bounds up to logarithmic (or constant) additive terms, i.e. $\approx f$ is $f + O(\log(f))$. Note that k is a tunable parameter but $k = 2$ is the sweet spot. We assume all random exponents are full sized and do not count multi-exponentiations.

1.2.9 Implementation.

In Section 5, we compare our implementations of (aggregate) range proofs. The theoretical prediction of 0.75× prover runtime compared to [16] is close to measurements, which suggest 0.7×. Using 140bit exponents, we experimentally attain $\approx 0.63\times$ compared to [16] on the same platform. As an important remark, we compare the dedicated range proofs of [16] with our generic instantiation of $\mathsf{QESA}_{\mathrm{ZK}}$.

1.3 Related work

Due to space constraints, we only elaborate on the most important concepts and related work. We refer to [33] for an overview and a general taxonomy.

The dlog setting and ILC. Very closely related works are [12, 13, 16, 28], which are efficient proofs in the dlog setting. Many zero-knowledge proofs in the group setting are instantiations of [19, 40]. The possibilities of our setting, namely ability to apply linear transformations to a committed witness has been abstracted in the ideal linear commitment model [14]. (Our techniques for $\mathsf{QESA}_{\mathrm{ZK}}$ are amenable to ILC.)

Knowledge assumptions. Another line of work [11, 21, 25, 29, 30, 39] gives non-interactive arguments using knowledge of exponent assumptions. They attain *constant size* proofs for arithmetic circuits and sublinear verification costs. However, they require a trusted setup.

PCPs, IOPs, MPC-in-the-head. Techniques, such as probabilistically checkable proofs (PCP), MPC-in-the-head [35], interactive oracle proofs (IOP) and more, construct efficient zero-knowledge proofs without relying on public key primitives. The possible performance gain (and quantum resistance) is interesting from a practical point of view. There is much progress on improving these techniques [2, 8, 17, 24, 46], which until recently suffered from relatively large proof size or unacceptable constants. In [8], Ben-Sasson et al. present a logarithmic communication IOP for R1CS. Still according to [8], proof sizes for R1CS statements of size $N = 10^6$ are about 130kb whereas our proofs, like Bulletproofs, stay well below 2kb. For combining proofs in the "symmetric key" setting with efficient proofs for "public key" algebraic statements, [1] can be used. Our proofs can be directly combined with algebraic statements over the same group $\mathbb{G}$.

2 Preliminaries

For a set S and probability distribution χ on S we write $s \leftarrow \chi$ for drawing s according to χ. We write $s \leftarrow S$ for a uniformly random element. We also write $y \leftarrow \mathcal{A}(x; r)$ for running an algorithm $\mathcal{A}$ with randomness r and $y \leftarrow \mathcal{A}(x)$ for running $\mathcal{A}$ with (uniformly) random $r \leftarrow R$ (where R is the randomness space). We let κ denote the security parameter and note that almost all objects are *implicitly* parameterised by it. By negl we denote some (fixed) negligible function, i.e. a function with $\lim_{\kappa\to\infty} \kappa^c \mathsf{negl}(\kappa) = 0$ for any $c \in \mathbb{N}$. We assume we can sample uniformly random from any $\{1, \dots, n\}$. The number $p \in \mathbb{N}$ will always denote a prime, $\mathbb{F}_p := \mathbb{Z}/p\mathbb{Z}$, and $\mathbb{G}$ is a (cyclic abelian) group of order p. We use additive implicit notation for $\mathbb{G}$ as introduced in [23]. That is, we write $[1]$ for some (fixed public) generator associated with $\mathbb{G}$ and $[x] := x[1]$. We extend this notation to vectors and matrices, i.e. for compatible $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ over $\mathbb{F}_p$, we write $\boldsymbol{A}[\boldsymbol{B}]\boldsymbol{C} = [\boldsymbol{ABC}]$. Matrices are bold, e.g. $[\boldsymbol{a}]$, components not, e.g. $[a_i]$. By $\boldsymbol{e}_i$ we denote the i-th standard basis vector. We write $\operatorname{diag}(\boldsymbol{M}_1, \dots, \boldsymbol{M}_n)$ for a block-diagonal matrix. By id_n we denote the $n \times n$ identity matrix.

2.1 Matrix kernel assumptions and Pedersen commitments

Instead of discrete logarithm assumptions, the generalisation of hard (matrix) kernel assumptions [41], but for right-kernels, better suits our needs.

Definition 2.1. Let $\mathbb{G} \leftarrow \mathsf{GrpGen}(1^\kappa)$ be a group generator (we let $[1]$ and p be implicitly given by $\mathbb{G}$). Let $\mathcal{D}_{m,n}$ be a (efficiently samplable) distribution over $\mathbb{G}^{m\times n}$ (where m and n may depend on κ). We say $\mathcal{D}_{m,n}$ has a **hard kernel assumption** if for all efficient adversaries $\mathcal{A}$, we have

$$\mathbb{P}(\mathbb{G} \leftarrow \mathsf{GrpGen}(1^\kappa); [\boldsymbol{A}] \leftarrow \mathcal{D}_{m,n}; \boldsymbol{x} \leftarrow \mathcal{A}(1^\kappa, \mathbb{G}, [\boldsymbol{A}])\colon [\boldsymbol{A}]\boldsymbol{x} = 0 \ \wedge\ \boldsymbol{x} \neq 0) \leq \mathsf{negl}(\kappa)$$

For simplicity, we will often only implicitly refer to $\mathcal{D}_{m,n}$ and just say $[\boldsymbol{A}]$ has hard kernel assumption. Note that kernel assumptions generalise discrete log assumptions: Finding a non-trivial kernel element of $[h, 1] \in \mathbb{G}^2$ immediately yields the discrete logarithm h of $[h]$.

If $\mathcal{D}_{m,n}$ is a matrix distribution with hard kernel assumption, then $[\boldsymbol{A}] \leftarrow \mathcal{D}_{m,n}$ is a (Pedersen) commitment key ck. Commit to $\boldsymbol{x} \in \mathbb{F}_p^n$ via $\mathsf{Com}_{\mathrm{ck}}(\boldsymbol{x}) = [\boldsymbol{c}] \in \mathbb{G}^m$. Breaking the binding property of the commitment is equivalent to finding non-trivial elements in $\ker([\boldsymbol{A}])$. The common case will be $[\boldsymbol{g}] \in \mathbb{G}^{1\times(n+1)}$ drawn uniformly as commitment key ck. Breaking the hard kernel assumption for $[\boldsymbol{g}]$ is tightly equivalent to breaking the dlog assumption in $\mathbb{G}$. Write $\boldsymbol{x} = (r_{\boldsymbol{w}}, \boldsymbol{w})$ with $r_{\boldsymbol{w}} \in \mathbb{F}_p$, $\boldsymbol{w} \in \mathbb{F}_p^n$. If $r_{\boldsymbol{w}} \leftarrow \mathbb{F}_p$ is drawn uniformly, it is evident that $[c] = [\boldsymbol{g}]\boldsymbol{x}$ perfectly hides $\boldsymbol{w}$, i.e. $[c]$ is uniformly distributed in $\mathbb{G}$.

Remark 2.2. It would be convenient to consider *hard kernel assumptions with prior knowledge* $V \leq \mathbb{F}_p^n$, where the subvector space V is some (leaked) knowledge about $\ker([\boldsymbol{B}])$ for $[\boldsymbol{B}] \leftarrow \mathcal{D}_{m,n}$. The reason being that we construct matrices $[\boldsymbol{B}]$ from $[\boldsymbol{A}]$ (where $[\boldsymbol{A}]$ has a standard hard kernel assumption) in such a way that some kernel elements are known, e.g. because $[\boldsymbol{B}]$ has some zero columns. However, to keep the overhead low, we deal with these cases explicitly.

2.2 Interactive arguments, extractability and zero-knowledge

Our setting will be the common reference string model, i.e. there is some CRS crs, typically a commitment key, set up by a trusted party. In the following $\mathcal{R}$ denotes a binary relation for which $(st, w) \in \mathcal{R}$ is efficiently decidable. We call st the statement and w the witness. ($\mathcal{R}$ does depend on crs, i.e. actually we consider (crs, st, w) tuples, but we suppress this.) The (NP-)language $\mathcal{L}$ defined by $\mathcal{R}$ is the language of statements in $\mathcal{R}$, i.e. $\mathcal{L} = \{st \mid \exists w\colon (st, w) \in \mathcal{R}\}$.

Definition 2.3. An **(interactive) argument system** for a relation $\mathcal{R}$ is a protocol between two parties, a **prover** $\mathcal{P}$ and a **verifier** $\mathcal{V}$. We use the name **(interactive) proof system** interchangably.[5] The transcript of the interaction of $\mathcal{P}$ and $\mathcal{V}$ on inputs x and y is denoted $\langle \mathcal{P}(x), \mathcal{V}(y)\rangle$ where both parties have a final "output" message. We write $b = \langle \mathcal{P}(x), \mathcal{V}(y)\rangle$, for the bit b indicating whether an (honest) verifier accepts the argument, where $b = 1$ means accept.

[5] More precise usage would only use the term *proof* and *proof system* if soundness against unbounded adversaries is guaranteed.

Definition 2.4 (Completeness). An interactive argument system for $(st, w) \in \mathcal{R}$ is (computationally) **complete** if for all efficient adversaries $\mathcal{A}$, we have

$$\mathbb{P}(\mathrm{crs} \leftarrow \mathsf{GenCRS}(1^\kappa); (st, w) \leftarrow \mathcal{A}(\mathrm{crs})\colon (st, w) \notin \mathcal{R} \text{ or } \langle \mathcal{P}(st, w), \mathcal{V}(st) \rangle = 1) \leq 1 - \mathsf{negl}(\kappa)$$

for a negligible function negl. It is **perfectly complete** if $\mathsf{negl} = 0$.

In Appendix D, we give a definition of witness-extended emulation [13, 31] with extraction error (i.e. knowledge error). It turns out that preserving a good extraction error over multiple rounds is non-trivial. See Sections 2.3 and 2.4.

Definition 2.5 (Public coin). An interactive argument system for $\mathcal{R}$ is **public coin** if all of the verifier's challenges are independent of any other messages or state (essentially $\mathcal{V}$ makes his random coins public). Furthermore, $\mathcal{V}$'s verdict is a function $\mathsf{Verify}(tr)$ of the transcript.

Definition 2.6. Let $(\mathcal{P}, \mathcal{V})$ be an interactive argument system for $\mathcal{R}$. We call $(\mathcal{P}, \mathcal{V})$ **(ε-statistical) honest-verifier zero-knowledge** (HVZK), if there exists an expected polynomial-time simulator Sim such that for all expected polynomial-time $\mathcal{A}$ the probabiliy distributions of $(\mathrm{crs}, tr, state)$, where

- $\mathrm{crs} \leftarrow \mathsf{GenCRS}(1^\kappa); (st, w) \leftarrow \mathcal{A}(\mathrm{crs}); tr \leftarrow \langle \mathcal{P}(st, w), \mathcal{V}(st) \rangle$
- $\mathrm{crs} \leftarrow \mathsf{GenCRS}(1^\kappa); (st, w) \leftarrow \mathcal{A}(\mathrm{crs}); tr \leftarrow \mathsf{Sim}(st, \rho);$

are indistinguishable (have statistical distance at most ε), assuming $tr := \perp$ if $(st, w) \notin \mathcal{R}$.

Remark 2.7. We focus on HVZK, not *special* HVZK, The latter states that even if the adversary chooses statement, witness and the verifier's *randomness* (ρ in Definition 2.6), the *special* simulator will "succeed". Our security proofs make use of *honest* challenges. Different (more complex) security proofs may be possible.

2.2.1 Full-fledged zero-knowledge.

To obtain security against dishonest verifiers, i.e. full-fledged zero-knowledge, simple transformations exist [18, 20, 27, 37] for public coin HVZK arguments. The most straightforward one is to use an equivocable coin toss between prover and verifier to generate the challenge.

2.2.2 The Fiat–Shamir heuristic.

In the random-oracle model (ROM), public coin arguments can be converted to non-interactive arguments by computing the (verifier's) challenges as the hash of the transcript (and relevant "context") up to that point. The statement of the argument should be part of the "context" [10].

2.3 Testing distributions

Intuitively, testing distributions are a special form of probabilistic verification where one can *efficiently* recover the "tested" value given enough "tests". Thus, they are used to recover the witness in proofs of knowledge. We only define testing distributions over $\mathbb{F}_p^m$.

Example 2.8. To test if a vector $[\boldsymbol{c}] \in \mathbb{G}^m$ is $[\boldsymbol{0}]$, test if $\boldsymbol{x}^\top [\boldsymbol{c}] \overset{?}{=} [0]$ for random $\boldsymbol{x} \in \mathbb{F}_p^m$. The soundness error is $1/p$.

Definition 2.9 (Subdistribution). Let χ be a distribution on $\mathbb{F}_p^m$. We call a distribution ψ on $\mathbb{F}_p^m$ a **subdistribution of χ weight ε** if

- there exists a **subdensity** $\rho_\psi\colon \mathbb{F}_p^m \to [0, 1]$. (It is important that $\rho_\psi(x) \leq 1$.)
- $\varepsilon = \sum_{x \in \mathbb{F}_p^n} \rho_\psi(x) \chi(x)$, and
- ψ has probability $\psi(x) = \frac{1}{\varepsilon} \rho_\psi(x)$ to pick x. (That is, ψ has density $\frac{1}{\varepsilon} \rho_\psi$ w.r.t. χ.)

The definition of a subdistribution is constructed to deal with adversaries. As a concrete example consider extraction by rewinding: It may happen that the adversary does not correctly answer a challenge. Thus, the challenges which are answered are a subset, or more generally a subdistribution. An adversary with success probability ε must succeed on a subdistribution of weight ε.

Definition 2.10. A **testing distribution** χ_m for $\mathbb{F}_p^m$ with soundness $\delta_{\mathsf{snd}}(\kappa)$ is a distribution over $\mathbb{F}_p^m$ with following property: For all subdistributions ψ of χ_m with weight $\varepsilon \geq \delta_{\mathsf{snd}} := \delta_{\mathsf{snd}}(\chi_m)$, we have

$$\mathbb{P}(\boldsymbol{x}_i \leftarrow \psi, \boldsymbol{X} = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_n) \colon \det(\boldsymbol{X}) = 0) \leq \tfrac{1}{\varepsilon}\delta_{\mathsf{snd}}.$$

We write $\delta_{\mathsf{snd}}(\chi_m)$ for some (fixed) soundness error of χ_m.

Note that $\det(\boldsymbol{X}) \neq 0$, is equivalent to all $\boldsymbol{x}_i$ being linearly independent, and to $\bigcap_{i=1}^m \ker(\boldsymbol{x}_i^\top) = \{0\}$. These interpretations allow to generalise in different directions. For more about testing distributions, see Appendix E. Typically, we want that $\delta_{\mathsf{snd}}(\chi)$ is very small, e.g. 2^{-100} in practice.

Our examples need a slight generalisation of the lemma of Schwartz–Zippel.

Lemma 2.11 (Schwartz–Zippel). *Let $f \in \mathbb{F}_p[X_1, \dots, X_n]$ be a* non-zero *polynomial of (total) degree d. Let χ be a distribution on $\mathbb{F}_p$. Let $p_\infty(\chi) := \sup_{x \in \mathbb{F}_p} \chi(x)$, where $\chi(x) := \mathbb{P}(x = y \mid x \leftarrow \chi)$. Then $\mathbb{P}(f(\boldsymbol{x}) = 0) \leq dp_\infty(\chi)$ for $\boldsymbol{x} \leftarrow \chi^n$ (i.e. $x_i \leftarrow \chi$). Moreoever, since $p_\infty(\psi) \leq \frac{1}{\varepsilon}p_\infty(\chi)$ for any subdistribution ψ of weight ε, we get $\mathbb{P}_{\boldsymbol{x} \leftarrow \psi^n}(f(\boldsymbol{x}) = 0) \leq \frac{dp_\infty(\chi)}{\varepsilon}$.*

Proof. We follow the standard proof. For $n = 1$ this is easy: Consider $f \neq 0$. Then f has at most d zeroes $y_1, \dots, y_n$. From $\chi(\{y_1, \dots, y_n\}) = \sum_{i=1}^d \chi(y_i) \leq dp_\infty(\chi)$, where $\chi(S) := \mathbb{P}(x \leftarrow \chi \colon x \in S)$, the claim follows

For degree n, write f as $f(X_1, \dots, X_n) = \sum_{i=0}^d X_1^i g_i(X_2, \dots, X_n)$. Note that $\deg(g_i) \leq d - i$. Let j be maximal with $g_j \neq 0$. The probability that we obtain 0 after a random evaluation at $\boldsymbol{x} \leftarrow \chi^n$ is smaller (by union bound) than the probability that $g_j(x_2, \dots, x_n) = 0$ plus the probability that $f(X_1, x_2, \dots, x_n)$ evaluated at x_1 is 0. Writing $\eta := p_\infty(\chi)$, we find by induction hypothesis that $\mathbb{P}(f(\boldsymbol{x}) = 0) \leq (d - j)\eta + j\eta = d\eta$, as claimed. □

Example 2.12 (Polynomial testing). The distribution induced by $\boldsymbol{x} = (x^0, \dots, x^{m-1})$, where $x \leftarrow \mathbb{F}_p$, is a testing distribution. This follows from the fact that $\boldsymbol{X}$ is a Vandermonde matrix, hence invertible except if the same x was chosen twice. It is easy to see that $\delta_{\mathsf{snd}}(\chi) \leq \frac{m}{p}$.

Example 2.13. For the special case $m = 2$, and testing distribution with $\boldsymbol{x} = (\alpha, 1)$ where $\alpha \leftarrow \mathcal{S}$ for some $\mathcal{S} \subseteq \mathbb{F}_p$ we write $\chi^{(\beta)}$ and $\alpha \leftarrow \chi^{(\beta)}$. If $\mathcal{S} \subseteq \mathbb{F}_p^\times$, i.e. $\alpha \neq 0$, we write $\chi^{(\beta \neq 0)}$.

Example 2.14 (Random testing). The uniform distribution over $\mathbb{F}_p^m$ is a testing distribution. The Lemma of Schwartz–Zippel immediately yields $\delta_{\mathsf{snd}}(\chi) \leq \frac{m}{p}$. Moreover, one can resort to a set $\mathcal{S}$ of "small exponents", i.e. draw from $\mathcal{S} = \{0, \dots, \ell - 1\}$ and still have soundness $\delta_{\mathsf{snd}}(\chi) \leq \frac{m}{\ell}$.

Example 2.15 (Pseudo-random testing). The verifier can replace truly random choices, e.g. $\boldsymbol{x} \leftarrow \mathbb{F}_p^m$ as above, by pseudorandom choices, e.g. $\boldsymbol{x} \leftarrow \mathsf{PRG}(s)$ for $s \leftarrow \{0,1\}^\kappa$. This allows the verifier to compress such challenges to a random seed s.

It is heuristically plausible, that any non-pathological PRG has distribution with soundness error (negligibly close) to that of the respective uniform distributions. In fact, for a PRG which is secure against *non-uniform* adversaries, this is easy to see. However, this is a strong assumption and there are distributions χ which are pseudorandom (under plausible assumptions), but where the soundness error $\delta_{\mathsf{snd}}(\chi)$ is large, e.g. greater than $\frac{1}{2}$. This motivates some *computational* notion of soundness error, which is discussed in Appendix E.

Note that soundness of testing distributions is a combinatorial property. No pseudorandomness property is required, as illustrated by Example 2.12. Thus, there may be better options to use "small exponents" than (pseudo)random testing.

2.3.1 Dual testing distributions.

Testing distributions are essentially a stronger (and simplified) form of the general concept of probabilistic verification with efficient extraction. They allow to test if an element in $\mathbb{F}_p^n$ is 0. By dualising, we find another concept, for which an intuitive description seems harder. Instead of a distribution on $\boldsymbol{x}^\top \in \mathbb{F}_p^{1 \times m}$ satisfying with high probability $\bigcap_{i=1}^m \ker(\boldsymbol{x}^\top) = \{\boldsymbol{0}\}$, we consider a distribution on $\boldsymbol{M} \in \mathbb{F}_p^{m \times m-1}$, satisfying with high probability $\bigcap_{i=1}^m \operatorname{im}(\boldsymbol{M}) = \{\boldsymbol{0}\}$, In a sense, $\boldsymbol{M}$ guarantees that for any $\mathbf{0} \neq \boldsymbol{z} \in \mathbb{F}_p^m$, $\boldsymbol{z} \notin \operatorname{im}(\boldsymbol{M})$ with high probability. Hence, we can use it to *enforce* $\boldsymbol{z} = \mathbf{0}$, instead of testing for it.

More concretely, we use this to ensure that for a Pedersen commitment $[c] = [\boldsymbol{G}|\boldsymbol{H}](\begin{smallmatrix}\boldsymbol{w}\\ \boldsymbol{z}\end{smallmatrix})$ the adversary must have $\boldsymbol{z} = \mathbf{0}$. We do so by constructing $[\boldsymbol{H}]$ as $[\boldsymbol{H}] \coloneqq [\boldsymbol{Q}]\boldsymbol{M}$. Intuitively, knowledge of some $[c'] = [\boldsymbol{G}|\boldsymbol{Q}](\begin{smallmatrix}\boldsymbol{w}\\ \boldsymbol{y}\end{smallmatrix})$ cannot be transferred to $[\boldsymbol{G}|\boldsymbol{H}]$ because we must have $\boldsymbol{z} = \boldsymbol{M}\boldsymbol{y}$, i.e. $\boldsymbol{z} \in \operatorname{im}(\boldsymbol{M})$, which is unlikely (except for $\boldsymbol{z} = \mathbf{0}$ or if $\mathscr{A}$ breaks the binding property). Thus, we can provably "zero" a part of a commitment without an (expensive) argument. Generally, this allows to derive "fresh" commitment keys. Using this is more communication efficient than picking and sending a fresh $[\boldsymbol{H}] \leftarrow \mathbb{G}^m$.

Morally, dual testing *enforces* $\boldsymbol{z} = \mathbf{0}$, while "normal" testing *verifies* $\boldsymbol{z} = \mathbf{0}$.

Definition 2.16. An (arbitrary) **dual testing distribution** $\chi_m^\vee$ is a distribution on $\mathbb{F}_p^{m \times (m-1)}$. The soundess error $\delta_{\mathsf{snd}}(\chi_m^\vee)$ is defined as before, but using $\mathbb{P}(\cap_{i=1}^m \operatorname{im}(\boldsymbol{M}_i) \neq \{\mathbf{0}\})$.

Let χ_m be a testing distribution on $\mathbb{F}_p^m$ such that $\boldsymbol{x} \leftarrow \chi_m$ *always* has $x_1 = 1$. Then $\chi_m^\vee$ defined as follows is a dual testing distribution: To pick $\boldsymbol{M} \leftarrow \chi_m^\vee$, pick $\boldsymbol{x}^\top = (1, \boldsymbol{x}')^\top \leftarrow \chi_m$ and let $\boldsymbol{M} \coloneqq \boldsymbol{M}_{\boldsymbol{x}} \coloneqq \left(\begin{smallmatrix}\boldsymbol{x}'\\ -\operatorname{id}_{m-1}\end{smallmatrix}\right)$. By construction $\ker(\boldsymbol{x}^\top) = \operatorname{im}(\boldsymbol{M}_{\boldsymbol{x}})$, and consequently $\delta_{\mathsf{snd}}(\chi^\vee) = \delta_{\mathsf{snd}}(\chi_m)$.

Note that by construction, $\boldsymbol{M}_{\boldsymbol{x}}$ is the (parity) check matrix for the linear code with generator $\boldsymbol{x}$. In particular, $\boldsymbol{x}^\top \boldsymbol{M}_{\boldsymbol{x}} = \mathbf{0}$. For simplicity, we only consider dual testing distributions associated to some testing distribution.

2.4 Special soundness

In the main body, we only consider special soundness and give extractors which produce a witness given a suitable tree of accepting transcripts, see also [13].

Definition 2.17 (μ-special soundness (over $\mathbb{F}_p$)). Let $(\mathsf{GenCRS}, \mathscr{P}, \mathscr{V})$ be a public coin argument system for $\mathscr{R}$. Suppose the verifier sends n challenges and $\mu = (\mu_0, \ldots, \mu_{n-1}) \in \mathbb{N}^n$. Furthermore, suppose the challenges are vectors in $\mathbb{F}_p^{n_i}$. Then the protocol is μ-special sound if there exists an extractor Ext such that given any good μ-tree tree_μ of transcripts, $\mathsf{Ext}(\mathit{st}, \mathit{tree}_\mu)$ returns a witness w with $(\mathit{st}, w) \in \mathscr{R}$. A μ-tree of transcripts is a (directed) tree where nodes of depth i have μ_i children, with edges labelled with the i-th challenge, and nodes labelled with the prover's i-th answer, and every path along the tree constitutes an *accepting* transcript. We call a μ-tree *good* if for every node, all its challenges (i.e. outgoing edges) are in general position.[6]

Caution 2.18. The choice of *general position* instead of just linear independence may not generalise well.[7]

Given a $\mathsf{TreeFind}$ algorithm, which produces good μ-trees (with oracle access to a successful prover), and an extractor as above, one obtains witness extended emulation by plugging the tree into the extractor. To be able to speak about the security of the resulting protocol, one needs *success* and *runtime guarantees* of $\mathsf{TreeFind}$. We do not deal with this here as it is a separate issue. See [13] for a $\mathsf{TreeFind}$er and [48] for further generalisations, more details are in Appendix D.

2.4.1 Short-circuit extraction.

In this section, we assume $\mathsf{TreeFind}$ produces the tree's nodes and leaves on demand, and Ext queries $\mathsf{TreeFind}$ *as an oracle*, and *traverses the tree in depth-first order.* Moreover, we are in a situation where Ext either extracts a witness for some statement, or a solution to a (supposedly) hard problem, or both. Concretely, we have statements like "we extract $\boldsymbol{w}$ such that either $[\boldsymbol{g}]\boldsymbol{w} = [c]$ is a valid commitment opening, or $[\boldsymbol{g}]\boldsymbol{w} = [0]$ breaks the hard kernel assumption for $[\boldsymbol{g}]$."

[6] Vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathbb{F}_p^n$ are in *general position* if any n of them are linearly independent. E.g. in the case $n = 1$, this is equivalent to all x's being different (and non-zero).

[7] We (only) need this in Lemma 3.10 (3), and (implicitly) its descendants, namely all of Section 4, because we use it to guarantee that the partial commitments $[\boldsymbol{u}_\ell]$ are all extractable. However, we only (state and) prove Lemma 3.10 for *special challenges* where general position has strong implications. We fear that the proof, as is, may not work in general, and additional properties are required for (generalised) testing distributions. Since we believe the instance of Lemma 3.10 to be optimal, c.f. Remark 3.11, and since this only affects extraction of $[\boldsymbol{u}_\ell]$, not the overall proof-of-knowledge property, we opt to keep this "bad definition" for now. Those who whish to use the Definition 2.17, be warned.

Definition 2.19. Consider the situation in Definition 2.17. Suppose $\mathcal{R}$ is $\mathsf{OR}(\mathcal{R}_1,\mathcal{R}_2)$, i.e. $\mathcal{R} = \{((st_1, st_2),(w_1,w_2)) \mid (st_i, w_i) \in \mathcal{R}_i \text{ for } i = 1 \text{ or } i = 2\}$.

Suppose there is some $\mu' \leq \mu$ (i.e. $\mu'_i \leq \mu_i$ for all i) such that extractor Ext has following property. For any good μ-tree $tree_\mu$, $\mathsf{Ext}(st, tree_\mu)$ we have either:

- Ext finishes after exploring a μ'-subtree of $tree_\mu$ and returns a witness for st_1. We call this *quick-extraction.*
- If in layer ℓ of the tree, Ext must explore more than μ'_ℓ children of some node, then after exploring all μ_ℓ children, Ext returns a witness for st_2 (and perhaps st_1). (That is, Ext *short-circuits* in layer ℓ.)

We say that such an Ext has **short-circuit** extraction with $\mu' \leq \mu$ for finding a witness to st_1 or to st_2. (Note that order of the statements matters!)

Our definition is ad-hoc and tailored to our needs. We leave a solid definition and precise treatment of short-circuit extraction for future work.

Corollary 2.20. *If* Ext *as in Definition 2.19 traverses a good tree* $tree_\mu$ *in depth-first order, we have following "runtime" guarantees: Let* $\mu' = (\mu'_0, \ldots, \mu'_{n-1}) \leq (\mu_0, \ldots, \mu_{n-1}) = \mu$. *In case of quick-extraction, at most* $\prod_{i=0}^{n-1} \mu'_i$ *leaves are explored. In case of short-circuit extraction, at most* $s_0 + 1$ *leaves are explored, where* $s_0 = \sum_{i=0}^{n-1}(\mu_i - 1)\prod_{j=i+1}^{n-1}\mu'_j$. *In particular,* $s_0 \leq (\sum_{i=0}^{n-1}\mu_i)(\prod_{i=0}^{n-1}\mu'_i)$.

Proof. Let t_i denote the *maximal number of leaves* necessary to ensure a $\mu|_i$-subtree, where $\mu|_i = (\mu_i, \ldots, \mu_{n-1})$, is extractable. We find $t_{n-1} = \mu_{n-1}$. Recursively, we find $t_i = (\mu_i - 1)\prod_{j=i+1}^{n-1}\mu'_j + t_{i-1}$, which yields our formula. (Note that $s_i = t_i - 1$ is the maximum number of leaves, such that one additional leaf guarantees extraction.)

We argue as follows: In the worst case, layer i short-circuits. If that happens, we have to extract all μ_i nodes. A subtree (in layer $i+1$) quick-extracts after exploring $\prod_{j=i+1}^{n-1}\mu'_j$ leaves. In case of failure, the subtree must short-circuit, requiring at most t_{i+1} nodes. In the worst case, the first $\mu_i - 1$ nodes in layer i quick-extract, and the last node, i.e. the μ_i-th node, again short-circuits. Thus, we again pay the costs[8] for a short-circuit extraction, now in layer $i+1$, which is bounded by t_{i+1}. Hence, at most t_{i+1} nodes are explored. The claim follows. □

We note that since the tree $tree_\mu$ is randomised (or Ext might explore children in random order), the above worst-case analysis is very conservative.

3 HVZK arguments for $[\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$

Let ck $:= [\boldsymbol{g}] = [g_0, \overline{\boldsymbol{g}}] \leftarrow \mathbb{G}^{1 \times n+1}$ be a Pedersen commitment key, where $[g_0] \in \mathbb{G}$ and $[\overline{\boldsymbol{g}}] \in \mathbb{G}^n$. Define $\mathsf{Com}_{\boldsymbol{g}}(\boldsymbol{w}; r) := [g_0]r + [\overline{\boldsymbol{g}}]\boldsymbol{w}$ for $r \in \mathbb{F}_p$, $\boldsymbol{w} \in \mathbb{F}_p^n$. In the whole section, we work with matrices $[\boldsymbol{A}] \in \mathbb{G}^{m \times n}$, and vectors $\boldsymbol{w} \in \mathbb{F}_p^n$ and $[\boldsymbol{t}] \in \mathbb{G}^m$. The dimensions are as above, unless otherwise specified. Our witness relation $\mathcal{R}$ is $st = ([\boldsymbol{A}], [\boldsymbol{t}])$ and $w = \boldsymbol{w}$ such that $(st, w) \in \mathcal{R}$ if and only if $[\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$.

3.1 Intuition

In this section, we devise communication efficient public-coin HVZK arguments for knowledge of a preimage of a linear map, i.e. $\exists \boldsymbol{w}\colon [\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$. We follow two principles: "Use probabilistic (batch) verification to check many things at once" and "If messages are too long, replace them by a shorter proof (of knowledge)." For this, we use shrinking commitments, to keep the messages small.

Our strategy is as follows: First, we recall the well-known general HVZK protocol [19, 40] for proving $\exists \boldsymbol{w}\colon [\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$ where $[\boldsymbol{A}] \in \mathbb{G}^{m \times n}$. Then, we show how to apply batch verification to reduce the argument for $([\boldsymbol{A}], [\boldsymbol{t}])$ to another an argument for some $([\boldsymbol{B}], [\boldsymbol{u}])$ with $[\boldsymbol{B}] \in \mathbb{G}^{2 \times n}$. This makes communication independent of the number m of rows of $[\boldsymbol{A}]$. After this, we revisit the arguments from [13] which recursively batch statement and witness, i.e. they reduce the number n of columns of

[8] Since $\mu' \leq \mu$, short-circuit extraction is never cheaper.

$[\boldsymbol{A}]$. Unlike [13, 16], we need a zero-knowledge version of these arguments. We provide a very efficient conversion with constant communication and logarithmic computational overhead. Taken together, we can prove knowledge of $\boldsymbol{w}$ in communication $O(\log(n))$ now.

3.2 Step 0: A standard Σ-protocol for $[\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$

Here, we recall the prototypical Σ-protocol in a group setting [19, 40].

Protocol 3.1 (Σ_{std}). The following is a protocol to prove $\exists \boldsymbol{w}\colon [\boldsymbol{t}] = [\boldsymbol{A}]\boldsymbol{w}$, using testing distribution $\chi^{(\beta)}$ for challenges, c.f. Example 2.13. Common input is $([\boldsymbol{A}], [\boldsymbol{t}]) \in \mathbb{G}^{m\times n} \times \mathbb{G}^{n}$. The prover's witness is some $\boldsymbol{w} \in \mathbb{F}_p^n$.

- $\mathscr{P} \to \mathscr{V}$: Pick $\boldsymbol{r} \leftarrow \mathbb{F}_p^n$ and compute $[\boldsymbol{a}] = [\boldsymbol{A}]\boldsymbol{r}$. Send $[\boldsymbol{a}] \in \mathbb{G}^m$.
- $\mathscr{V} \to \mathscr{P}$: Pick $\beta \leftarrow \chi^{(\beta)}$. Send $\beta \in \mathbb{F}_p$.
- $\mathscr{P} \to \mathscr{V}$: Compute $\boldsymbol{z} = \beta\boldsymbol{w} + \boldsymbol{r}$. Sends $\boldsymbol{z} \in \mathbb{F}_p^n$.
- $\mathscr{V}$: Check $[\boldsymbol{A}]\boldsymbol{z} \overset{?}{=} \beta[\boldsymbol{t}] + [\boldsymbol{a}]$. (Accept/reject if true/false.)

It is straightforward to show that any $(x_1, x_2) \leftarrow \chi_2$ can be used instead of $\chi^{(\beta)}$, as long as $x_2 \neq 0$, so that $x_1\boldsymbol{w} + x_2\boldsymbol{r}$ is uniformly distributed, c.f. Section 1.1.

Lemma 3.2. *Protocol* Σ_{std} *is a* HVZK-*PoK for* $\exists w\colon [\boldsymbol{t}] = [\boldsymbol{A}]\boldsymbol{w}$. *It is perfectly complete, has perfect* HVZK *and is* 2*-special sound.*

Proof. **Completeness:** is straightforward to verify.

Extraction: We are given two accepting transcripts $([\boldsymbol{a}], \beta, \boldsymbol{z})$, and $([\boldsymbol{a}], \beta', \boldsymbol{z}')$ with $\beta - \beta' \neq 0$. Due to the final check of the verifier, we obtain $\frac{1}{\beta-\beta'}[\boldsymbol{A}](\boldsymbol{z} - \boldsymbol{z}') = [\boldsymbol{t}]$. Consequently, $\boldsymbol{w} \coloneqq \frac{1}{\beta-\beta'}(\boldsymbol{z} - \boldsymbol{z}')$ is a witness.

HVZK: Pick $\beta \leftarrow \chi^{(\beta)}$ and $\boldsymbol{z} \leftarrow \mathbb{F}_p^m$. Then $[\boldsymbol{a}] \coloneqq [\boldsymbol{A}]\boldsymbol{z} - \beta[\boldsymbol{t}]$ is uniquely defined. Since the distribution of β and $\boldsymbol{z}$ is as in an honest execution, this yields a perfect simulation. □

Now, we improve communication efficiency. We do this in two steps. First, we make the communication independent of the number m of equations, using batch-verification. Then we make it logarithmic in the size n of the witness, using techniques from [13, 16]. We apply all techniques mentioned in the introduction, using shrinking commitments to keep messages small. Composition of proof systems is implicit due the following remark.

Remark 3.3. AND-proofs for statements of the form $\exists \boldsymbol{w}\colon [\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$ are trivial. Namely, to prove $\exists \boldsymbol{w}\colon [\boldsymbol{A}_1]\boldsymbol{w} = [\boldsymbol{t}_1] \wedge [\boldsymbol{A}_2]\boldsymbol{w} = [\boldsymbol{t}_2]$, it suffices to define $[\boldsymbol{A}] = \left[\begin{smallmatrix}\boldsymbol{A}_1\\ \boldsymbol{A}_2\end{smallmatrix}\right]$ and $[\boldsymbol{t}] = \left[\begin{smallmatrix}\boldsymbol{t}_1\\ \boldsymbol{t}_2\end{smallmatrix}\right]$ and prove $\exists \boldsymbol{w}\colon [\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$. This AND-compilation technique will be used without explicit mention. Evidently, many trivial optimisations are possible, e.g. removing duplicate rows.

3.3 Step 1: Batching all equations together

In this step, we devise a HVZK-AoK for $\exists \boldsymbol{w}\colon [\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$, where $\mathscr{P}$'s communication is independent of m, the "number of equations". Thus, we have to shrink the message $[\boldsymbol{a}] \in \mathbb{G}^m$ somehow. We would like to batch all m linear equations (given by $[\boldsymbol{A}]$) into a single linear equation, i.e. replace $[\boldsymbol{A}]$ by a random linear combination of its rows. We do not know whether this is sound or not, c.f. Question 3.7. Nevertheless, if $\mathscr{P}$ has explicitly committed to the witness $\boldsymbol{w}$ (or $[\boldsymbol{a}]$), the statement — excluding the commitment — can be batched, as $\mathscr{P}$ cannot change his mind anymore.

Note that the value $[\boldsymbol{t}]$ is in general *not* a commitment, since the adversary may supply (parts of) $[\boldsymbol{A}]$ in the soundness experiment. Thus, he may know dlogs and generate preimages of $[\boldsymbol{t}]$ freely. By adding a commitment to $\boldsymbol{w}$, we get around this problem.

By using a shrinking commitment to $\boldsymbol{w}$, we ensure that the communication is small. Now the verifier can send batching randomness, and a HVZK-AoK for the batched statement is carried out. We directly apply AND-compilation in the protocol. We use general testing distributions, but the reader may want to imagine the familiar setting of polynomial testing with $\boldsymbol{x} = (x^0, \ldots, x^m)$ first.

Protocol 3.4 (Protocol $\mathsf{LMPA}_{\text{batch}}$). The following is a protocol to prove $\exists \boldsymbol{w} \colon [\boldsymbol{t}] = [\boldsymbol{A}]\boldsymbol{w}$. Let χ_m and $\chi^{(\beta)}$ be testing distributions. Common input is $([\boldsymbol{A}], [\boldsymbol{t}]) \in \mathbb{G}^{m \times n} \times \mathbb{G}^m$. The prover's witness is some $\boldsymbol{w} \in \mathbb{F}_p^n$.

- $\mathcal{P} \to \mathcal{V}$: Pick $r_{\boldsymbol{w}} \leftarrow \mathbb{F}_p$, and compute $[c_{\boldsymbol{w}}] = [g_0]r_{\boldsymbol{w}} + [\overline{\boldsymbol{g}}]^\top \boldsymbol{w} = \mathsf{Com}(\boldsymbol{w}; r_{\boldsymbol{w}})$. Send $[c_{\boldsymbol{w}}]$.
- $\mathcal{V} \to \mathcal{P}$: Pick $\boldsymbol{x} \leftarrow \chi_m$. Send $\boldsymbol{x}$.
  Let $[\widehat{\boldsymbol{A}}] = \boldsymbol{x}^\top [\boldsymbol{A}] \in \mathbb{G}^{1 \times n}$ and $[\widehat{t}] = \boldsymbol{x}^\top [\boldsymbol{t}] \in \mathbb{G}$ be the batched statement (for both $\mathcal{P}$ and $\mathcal{V}$). Let $[\boldsymbol{B}] := \begin{bmatrix} g_0 & \overline{\boldsymbol{g}} \\ 0 & \widehat{\boldsymbol{A}} \end{bmatrix}$ and let $\exists (\boldsymbol{w}, r_{\boldsymbol{w}}) \colon [\boldsymbol{B}] \begin{pmatrix} r_{\boldsymbol{w}} \\ \boldsymbol{w} \end{pmatrix} = \begin{bmatrix} c_{\boldsymbol{w}} \\ \widehat{t} \end{bmatrix} =: [\boldsymbol{u}]$ be the new (AND-type) statement.
- $\mathcal{P} \leftrightarrow \mathcal{V}$: Engage in Protocol Σ_{std} for $\exists \left(\begin{smallmatrix} r_{\boldsymbol{w}} \\ \boldsymbol{w} \end{smallmatrix} \right) \colon [\boldsymbol{B}] \left(\begin{smallmatrix} r_{\boldsymbol{w}} \\ \boldsymbol{w} \end{smallmatrix} \right) = [\boldsymbol{u}]$.

In words, Protocol $\mathsf{LMPA}_{\text{batch}}$ batches $[\boldsymbol{A}]$ to $[\widehat{\boldsymbol{A}}]$, and carries out an AND-proof for opening the commitment $[c_{\boldsymbol{w}}]$ and that the content $\boldsymbol{w}$ of $[c_{\boldsymbol{w}}]$ is preimage of $[\widehat{t}]$ under $[\widehat{\boldsymbol{A}}]$. This is proven via a subprotocol call to Protocol Σ_{std}.

Lemma 3.5. *Protocol* $\mathsf{LMPA}_{\text{batch}}$ *is a* 5*-move* HVZK*-AoK for* $\exists w \colon [\boldsymbol{t}] = [\boldsymbol{A}]\boldsymbol{w}$ *with* $(m, 2)$*-special soundness for finding a witness or a non-trivial kernel element for* $[\boldsymbol{g}]$*. It has* $(1, 2)$ *short-circuit extraction.*

Proof. **Completeness:** It is straightforward to see that completeness holds.

Zero-knowledge: The simulator picks $\beta, \boldsymbol{x}$ according to the distributions. The simulator proceeds in two steps. First, simulate the Protocol Σ_{std}, i.e. the final three rounds. Since those are now simulated independently of $[c_{\boldsymbol{w}}]$, we pick $[c_{\boldsymbol{w}}] \leftarrow \mathbb{G}$ randomly. This gives a perfect HVZK simulation.

Extraction: Given a good $(m, 2)$-tree *tree*$_\mu$, we first extract the second layer (i.e. the subprotocol Σ_{std}). If not all of them yield the same $(r_{\boldsymbol{w}}, \boldsymbol{w})$, we found a non-trivial kernel element for $[\boldsymbol{g}]$ and are finished. To prove short-circuit extraction, we show that if this does not happen, $\boldsymbol{w}$ is a valid witness. Now, for all $\boldsymbol{x}_i$, we have $[\boldsymbol{B}_i] \left(\begin{smallmatrix} r_{\boldsymbol{w}} \\ \boldsymbol{w} \end{smallmatrix} \right) = \left[\begin{smallmatrix} c_{\boldsymbol{w}} \\ \widehat{t}_i \end{smallmatrix} \right]$, where the subscript i denotes the matrices of the i-th round. Then in particular,

$$\boldsymbol{x}_i^\top [\boldsymbol{A}]\boldsymbol{w} = [\widehat{\boldsymbol{A}}_i]\boldsymbol{w} = [\widehat{t}_i] = \boldsymbol{x}_i^\top [\boldsymbol{t}].$$

Arranging the m linear equations into a vector, we find with $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$,

$$\boldsymbol{X}^\top [\boldsymbol{A}]\boldsymbol{w} = \boldsymbol{X}^\top [\boldsymbol{t}] \quad \text{and hence} \quad [\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}].$$

Since *tree*$_\mu$ is *good*, $\boldsymbol{X}$ is invertible. Thus $\boldsymbol{w}$ is a valid witness. □

Remark 3.6 (Commitment extending). When working with adversarial $[\boldsymbol{A}]$ (and $[\boldsymbol{t}]$), one can not rely on any hardness assumptions. Extending $[\boldsymbol{A}]$ to some $[\boldsymbol{B}]$ which has hardness (as in Protocol 3.4) is one way to circumvent problems. For the sake of referencing, we call this *commitment extending* $[\boldsymbol{A}]$. If $[\boldsymbol{A}]$ already contains a commitment submatrix, there is an obvious adaption of Protocol $\mathsf{LMPA}_{\text{batch}}$. More concretely, randomly sum together all rows but those in the submatrix into a single row, as done before, and run the subprotocol on this statement instead.

Note that commitment extending $[\boldsymbol{A}]$ was not necessary for Protocol Σ_{std}, where extraction is unconditional. This raises following (to the best of our knowledge open) question.

Question 3.7. Is batch-verification without an (unbatched) commitment sound? That is, $\mathcal{V}$ initiates $\mathsf{LMPA}_{\text{batch}}$ and sends $\boldsymbol{x}$ immediately. Then $\exists \boldsymbol{w} \colon [\widehat{\boldsymbol{A}}]\boldsymbol{w} = [\widehat{t}]$ is proven. Since the statements are adversarially chosen, this is essentially an information-theoretic question. Partial results show that soundness holds at least in certain (very) special cases. The gist of this question recurs in different guises, and culminates in the question whether many of the presented arguments (and many in the literature) may in fact be *proofs* of knowledge.

3.4 Step 2: "Batching" the witness

In this section, we show how to "batch" the witness, i.e. proving $\exists \boldsymbol{w} \colon [\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$ for $[\boldsymbol{A}] \in \mathbb{G}^{m \times n}$ with communication sublinear in n. For the introduction, one may assume $m = 1$, e.g. $[\boldsymbol{A}] = [\boldsymbol{g}]$. (Using $\mathsf{LMPA}_{\text{batch}}$, we can reduce to $m \geq 2$.)

Remark 3.8. We can also reduce to $m = 1$ *conceptually*. Namely, let $\mathbb{H} := \mathbb{G}^m$. Then $[\boldsymbol{A}]$ and $[\boldsymbol{t}]$ can be interpreted as $[\boldsymbol{A}] \in \mathbb{H}^{1\times n}$, $[\boldsymbol{t}] \in \mathbb{H}$, and $[\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$. Using $\mathbb{H}$ means working over a vector space of *dimension* $m > 1$. This is a relevant difference, but mostly affects zero-knowledge.[9]

3.4.1 The general idea.

We present the technique of [13], but in our situation and notation. For the motivation, let us ignore zero-knowledge, and only construct an argument (of knowledge). We add zero-knowledge later.

Let $k \in \mathbb{N}$ be the size-reduction we want to achieve. Assume for simplicity that $k|n$, i.e. $n/k \in \mathbb{N}$.[10] We will reduce the equation $[\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$ to $[\widehat{\boldsymbol{A}}]\widehat{\boldsymbol{w}} = [\widehat{\boldsymbol{t}}]$, where $[\widehat{\boldsymbol{A}}] \in \mathbb{G}^{m\times n/k}$, $\widehat{\boldsymbol{w}} \in \mathbb{F}_p^{n/k}$, $[\widehat{\boldsymbol{t}}] \in \mathbb{G}^m$. To do so, divide $[\boldsymbol{A}]$ and $\boldsymbol{w}$ into k equal blocks,[11] obtaining vectors/matrices of vectors/matrices i.e. $[\boldsymbol{A}] = [\boldsymbol{A}_1|\dots|\boldsymbol{A}_k] \in (\mathbb{G}^{m\times n/k})^{1\times k}$ with $[\boldsymbol{A}_i] \in \mathbb{G}^{m\times n/k}$, and likewise $\boldsymbol{w} = \left(\begin{smallmatrix}\boldsymbol{w}_1\\ \cdots\\ \boldsymbol{w}_k\end{smallmatrix}\right) \in (\mathbb{G}^{n/k})^k$. We want to prove

$$\sum_{i=1}^{k}[\boldsymbol{A}_i]\boldsymbol{w}_i = [\boldsymbol{t}].$$

Still, the techniques from Section 3.3 are not applicable, because $[\boldsymbol{t}] \in \mathbb{G}$ (if $m = 1$). The trick of [13] is to embed our problem into a different one which can be batch-verified. Namely, we exploit that the scalar product is the sum of the diagonal entries (i.e. the trace) of the outer product:

$$\begin{bmatrix}\boldsymbol{A}_1\\ \vdots\\ \boldsymbol{A}_k\end{bmatrix}(\boldsymbol{w}_1,\dots,\boldsymbol{w}_k) = \begin{bmatrix}\boldsymbol{A}_1\boldsymbol{w}_1 & \boldsymbol{A}_1\boldsymbol{w}_2 & \dots & \boldsymbol{A}_1\boldsymbol{w}_k\\ \boldsymbol{A}_2\boldsymbol{w}_1 & \boldsymbol{A}_2\boldsymbol{w}_2 & \dots & \boldsymbol{A}_2\boldsymbol{w}_k\\ \vdots & \vdots & \ddots & \\ \boldsymbol{A}_k\boldsymbol{w}_1 & \boldsymbol{A}_k\boldsymbol{w}_2 & \dots & \boldsymbol{A}_k\boldsymbol{w}_k\end{bmatrix} \in \mathbb{G}^{k\times k} \tag{3.1}$$

Now we can send all terms $[\boldsymbol{A}_i]\boldsymbol{w}_j$ to the verifier. Our probabilistic test has to map both $[\boldsymbol{A}]$ and $\boldsymbol{w}$ to a new (smaller) statement. We can do that by multiplying from the left by $\boldsymbol{x} \in \mathbb{F}_p^k$ and from the right by $\boldsymbol{y} \in \mathbb{F}_p^k$ where $\boldsymbol{x}, \boldsymbol{y} \leftarrow \chi_k$. Consequently, we obtain (from associativity)

$$\boldsymbol{x}^\top\left(\begin{bmatrix}\boldsymbol{A}_1\\ \vdots\\ \boldsymbol{A}_k\end{bmatrix}(\boldsymbol{w}_1,\dots,\boldsymbol{w}_k)\right)\boldsymbol{y} = \underbrace{\left(\boldsymbol{x}^\top\begin{bmatrix}\boldsymbol{A}_1\\ \vdots\\ \boldsymbol{A}_k\end{bmatrix}\right)}_{:=\sum_i x_i[\boldsymbol{A}_i]\,=:\,[\widehat{\boldsymbol{A}}]}\underbrace{((\boldsymbol{w}_1,\dots,\boldsymbol{w}_k)\boldsymbol{y})}_{:=\sum_i y_i\boldsymbol{w}_i=:[\widehat{\boldsymbol{w}}]} = \underbrace{\sum_{i,j}x_iy_j[\boldsymbol{A}_i]\boldsymbol{w}_j}_{=:[\widehat{t}]}$$

The prover thus sends the (purported) $[\boldsymbol{A}_i\boldsymbol{w}_j]$, denoted $[\boldsymbol{u}_{i,j}]$, and $\widehat{\boldsymbol{w}}$, the shrunk witness. The verifier checks $\sum_i[\boldsymbol{u}_{i,i}] \overset{?}{=} [\boldsymbol{t}]$ and $[\widehat{\boldsymbol{A}}]\widehat{\boldsymbol{w}} \overset{?}{=} [\widehat{t}] = \sum_{i,j}x_iy_j[\boldsymbol{u}_{i,j}]$.

If each $[\boldsymbol{A}_i]$ satisfies a hard kernel assumption, the prover is committed to $\boldsymbol{w}_1,\dots,\boldsymbol{w}_k$. It is not hard to see that given enough (linearly independent) challenges, one can extract $\boldsymbol{w}$ (or find non-trivial kernel elements.) We will show this for a more efficient special case. All in all, we reduced the statement $([\boldsymbol{A}],[\boldsymbol{t}])$ to $([\widehat{\boldsymbol{A}}],[\widehat{\boldsymbol{t}}])$ which is smaller by a factor of k. This can be applied recursively.

3.4.2 Refining the testing distribution.

It turns out, that by a good choice of testing distribution, we can reduce communication. Namely, we can pick testing distributions with $x_iy_j = z_{j-i}$ for all i, j. Then it is sufficient for the verifier to know the sum of the off-diagonals[12] i.e. $[\boldsymbol{u}_\ell] := \sum_{j-i=\ell}[\boldsymbol{A}_i]\boldsymbol{w}_j$ for $\ell = \pm1,\dots,\pm k$ (and $[\boldsymbol{u}_0] = [\boldsymbol{t}]$). This follows from $\sum_{j-i=\ell}x_iy_j[\boldsymbol{A}_i]\boldsymbol{w}_j = z_\ell\sum_{j-i=\ell}[\boldsymbol{A}_i]\boldsymbol{w}_j$. We denote the (purported) $\sum_{j-i=\ell}[\boldsymbol{A}_i]\boldsymbol{w}_j$, sent by the prover, as $[\boldsymbol{u}_\ell]$. Note that $[\boldsymbol{u}_0] = [\boldsymbol{t}]$ need not be sent. From the testing distribution $\widetilde{\chi}_{2k-1}$ we

[9] Drawing a random $[b] \leftarrow \mathbb{H}$ needs a *basis* $[h_i]$ of $\mathbb{H}$ and sets $[b] = \sum r_i[h_i]$ for $r_i \leftarrow \mathbb{F}_p$.

[10] Pad $[\boldsymbol{A}]$ and the witness with zeroes if necessary. Note the technical "difficulties" that arise, Remark 2.2.

[11] It may be helpful to think of the vector space $(\mathbb{F}_p^{n/k})^k$ as $\mathbb{F}_p^{n/k}\otimes\mathbb{F}_p^k$.

[12] Any diagonal which is "parallel" to the diagonal (i.e. $(M_{i,j})_{j-i=\ell}$ for some ℓ) is called off-diagonal.

require that $\boldsymbol{z} \leftarrow \widetilde{\chi}_{2k-1}$, belongs to a pair $(\boldsymbol{x}, \boldsymbol{y})$. We always implicitly consider $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ for $\widetilde{\chi}_{2k-1}$, as these values belong together.[13]

One testing distribution with this property comes from monomials X^i, e.g. $\boldsymbol{x} = (1, x, \dots, x^{k-1})$ and $\boldsymbol{y} = (1, x^{-1}, \dots, x^{-k+1})$.[14] In this case, $z_\ell = x^{-\ell}$.

For efficiency, picking $\boldsymbol{x}$ as above, but $\boldsymbol{y} = (x^{k-1}, \dots, x, 1)$ is also interesting, since this preserves small x^i. In this case, $z_\ell = x^{k-1-\ell}$.

Protocol 3.9 ($\mathsf{LMPA}_{\mathrm{noZK}}$). The following is a protocol to prove $\exists \boldsymbol{w} \colon [\boldsymbol{t}] = [\boldsymbol{A}]\boldsymbol{w}$. Let $\widetilde{\chi}_{2k-1}$ be a testing distributions with the properties described above. Common input is $([\boldsymbol{A}], [\boldsymbol{t}]) \in \mathbb{G}^{m \times n} \times \mathbb{G}^m$. We assume $n = k^d$. The prover's witness is some $\boldsymbol{w} \in \mathbb{F}_p^n$.

Recursive step. Suppose $n = k^d > k$.

- Notation: Let $[\vec{\boldsymbol{A}}^\top] := \begin{bmatrix} \boldsymbol{A}_1 \\ \vdots \\ \boldsymbol{A}_k \end{bmatrix} \in (\mathbb{G}^{m \times n/k})^k$ and $\vec{\boldsymbol{w}} := \begin{pmatrix} \boldsymbol{w}_1 \\ \vdots \\ \boldsymbol{w}_k \end{pmatrix} \in (\mathbb{F}_p^{n/k})^k$, where $[\boldsymbol{A}] = [\boldsymbol{A}_1, \dots, \boldsymbol{A}_k] \in (\mathbb{G}^{m \times n/k})^{1 \times k}$.
- $\mathcal{P} \to \mathcal{V}$: Compute $[\boldsymbol{u}_\ell] = \sum_{j-i=\ell} [\boldsymbol{A}_i]\boldsymbol{w}_j$. Send $[\boldsymbol{u}_\ell]$ for $\ell = \pm 1, \dots, \pm(k-1)$. ($[\boldsymbol{u}_0] = [\boldsymbol{t}]$ is known to the verifier.)
- $\mathcal{V} \to \mathcal{P}$: Pick $\boldsymbol{z} \leftarrow \widetilde{\chi}_{2k-1}$ with corresponding $\boldsymbol{x}, \boldsymbol{y}$. Send $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$.
- Both parties compute $[\widehat{\boldsymbol{A}}] = \boldsymbol{x}^\top [\vec{\boldsymbol{A}}^\top] = \sum_i x_i [\boldsymbol{A}_i] \in \mathbb{G}^{m \times n/k}$ and $[\widehat{\boldsymbol{t}}] = \boldsymbol{z}^\top [\vec{\boldsymbol{u}}] = \sum_{\ell=-k}^{k} z_\ell [\boldsymbol{u}_\ell] \in \mathbb{G}$ the new batched statement. Moreover, $\mathcal{P}$ computes $\widehat{\boldsymbol{w}} = \vec{\boldsymbol{w}}^\top \boldsymbol{y} = \sum_i \boldsymbol{w}_i y_i$. The protocol may then be (recursively resumed), setting $n \leftarrow n/k$, $\boldsymbol{w} \leftarrow \widehat{\boldsymbol{w}}$, $[\boldsymbol{t}] \leftarrow [\widehat{\boldsymbol{t}}]$, $[\boldsymbol{A}] \leftarrow [\widehat{\boldsymbol{A}}]$.

Base case. Suppose $n \leq k$.

- $\mathcal{P} \to \mathcal{V}$: Send $\boldsymbol{w}$.
- $\mathcal{V}$: Tests if $[\boldsymbol{A}]\boldsymbol{w} \overset{?}{=} [\boldsymbol{t}]$.

See Appendix G for a sketch of the protocol.

For efficiency reasons, our base case could start at $n = 2k$, not k. This (only) makes a difference for $k = 2$, where it saves one round-trip. However, this would interfere with our zero-knowledge conversions, slightly complicating complicating the proof and constructions. We could describe Protocol $\mathsf{LMPA}_{\mathrm{noZK}}$ for general $n = k_1 \dots k_\ell$, as [13]. To keep the technicalities in check, we choose not to.

Lemma 3.10 (Recursive extraction)**.** *Consider the situation above. Let $\widetilde{\chi}_{2k-1}$ be a testing distribution with $x_i y_j = z_{j-i}$ as above.*[15] *Let $[\boldsymbol{u}_\ell]$, $[\boldsymbol{A}_i]$, $[\boldsymbol{t}]$, $\boldsymbol{w}_j$ and $[\widehat{\boldsymbol{A}}]$, $[\widehat{\boldsymbol{t}}]$ be defined as above. Then:*

1. *Given a non-trivial kernel element of $[\widehat{\boldsymbol{A}}]$, we (efficiently) find a non-trivial kernel element of $[\boldsymbol{A}]$.*
2. *Given $2k-1$ linearly independent challenges (with accepting transcripts), i.e. an invertible matrix $\boldsymbol{Z}$, one can extract (unconditionally) a witness $[\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$.*
3. *Given $2k$ challenges in general position,*[16] *if the witness from above does not fit w.r.t. the $[\boldsymbol{u}_\ell]$, i.e. if an honest prover would send different $[\boldsymbol{u}_\ell]$ for $\boldsymbol{w}$, then we find (additionally) a non-trivial kernel element $\boldsymbol{v}$, i.e $[\boldsymbol{A}]\boldsymbol{v} = 0$.*

Moreover, we have short-circuit extraction*: From k independent challenges, one can compute a candidate witness $\boldsymbol{w}'$ for quick-extraction. If $\sum_{j-i=\ell} [\boldsymbol{A}_i]\boldsymbol{w}'_j \neq [\boldsymbol{u}_\ell]$ for some ℓ, then we are guaranteed to find a non-trivial kernel relation from $2k$ challenges in general position.*

Note that, maybe surprisingly, extraction of a witness $\boldsymbol{w}$ with $[\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$ is unconditional, i.e. we have a *proof* of knowledge,[17] see also Question 3.7. The proof is a minor generalisation of [13, 16].

[13] It is possible to give a suitably generalised definition of testing distributions.

[14] It can be shown that, up to scalar multiples, these are all possible such testing distributions. In the case $n = 2$, it's easy to see that $\frac{x_2}{x_1} = \frac{y_1}{y_2} =: \rho$. By fixing either two consecutive x's or y's and letting the other vary, we see that $\frac{x_i}{x_j} = \frac{y_j}{y_i} = \rho^{j-i}$.

[15] Note that the soundness error $\delta_{\mathsf{snd}}(\widetilde{\chi}_{2k-1})$ is an upper bound for the soundness errors of the (induced) testing distributions for $\boldsymbol{x}$ and $\boldsymbol{y}$.

[16] By Footnote 14, $2k$ challenges are in general position if $\rho = x_2/x_1$ is different for all challenges.

[17] Note that unconditional extraction does not apply to openings of $[\boldsymbol{u}_\ell]$.

Proof. Given a non-trivial kernel element $\widehat{\boldsymbol{w}}$ for $\boldsymbol{x}^\top[\vec{\boldsymbol{A}}^\top]$, i.e. $0 = \boldsymbol{x}^\top[\vec{\boldsymbol{A}}^\top]\widehat{\boldsymbol{w}} = \sum_i [\boldsymbol{A}_i] x_i \widehat{\boldsymbol{w}}$, we see that $\widehat{\boldsymbol{w}}_{\boldsymbol{x}}$ as defined below satisfies $[\boldsymbol{A}]\widehat{\boldsymbol{w}}_{\boldsymbol{x}} = 0$. Thus, we can recursively "extend" kernel elements to earlier rounds.

Now to the interesting case. Given $2k-1$ linearly independent $\boldsymbol{z}^{(i)}$, we find

$$\begin{aligned}
[\boldsymbol{u}_{-k+1}, \dots, \boldsymbol{u}_{k-1}]\boldsymbol{z}^{(i)} &= (\sum_{j=1}^{k} x_j^{(i)}[\boldsymbol{A}_j])\widehat{\boldsymbol{w}}^{(i)} \\
&= \sum_j [\boldsymbol{A}_j] x^{(i)} \widehat{\boldsymbol{w}}^{(i)} \qquad \text{where} \quad \widehat{\boldsymbol{w}}_{\boldsymbol{x}}^{(i)} := \begin{pmatrix} x_1^{(i)}\widehat{\boldsymbol{w}} \\ \vdots \\ x_n^{(i)}\widehat{\boldsymbol{w}} \end{pmatrix} \in (\mathbb{F}_p^{n/k})^k. \\
&= [\boldsymbol{A}_1, \dots, \boldsymbol{A}_n]\widehat{\boldsymbol{w}}_{\boldsymbol{x}}^{(i)}
\end{aligned}$$

Note that $\widehat{\boldsymbol{w}}_{\boldsymbol{x}}^{(i)}$ is a column vector of vectors, and is multiplied with a a row vector of matrices.[18] We will sometimes explicate this by writing $[\vec{\boldsymbol{A}}]^\top$ and $\vec{\boldsymbol{w}}$. Concretely, we let $[\vec{\boldsymbol{A}}^\top] = \left[\begin{smallmatrix} \boldsymbol{A}_1 \\ \cdots \\ \boldsymbol{A}_k \end{smallmatrix}\right] \in (\mathbb{F}_p^{m \times n/k})^k$ be a matrix (of matrices). Thus, we get $[\vec{\boldsymbol{A}}]\vec{\boldsymbol{w}} = [\boldsymbol{t}]$ for the a witness $\vec{\boldsymbol{w}}$. For simplicity, the reader may think of the case $m = 1$, $n = k$ where we deal with "normal" vectors and matrices, c.f. Remark 3.8. To gather all equations in a single linear system, let

$$\boldsymbol{Z} := (\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(2k-1)}) \in \mathbb{F}_p^{(2k-1)\times(2k-1)} \quad \text{and} \quad \widehat{\boldsymbol{W}} := (\widehat{\boldsymbol{w}}_{\boldsymbol{x}}^{(1)}, \dots, \widehat{\boldsymbol{w}}_{\boldsymbol{x}}^{(2k-1)}) \in (\mathbb{F}_p^{n/k})^{k\times(2k-1)}$$

and note that we obtain

$$[\boldsymbol{u}_{-k+1}, \dots, \boldsymbol{u}_{k-1}]\boldsymbol{Z} = [\boldsymbol{A}_1, \dots, \boldsymbol{A}_k]\widehat{\boldsymbol{W}}$$

as the linear system. Multiplication by $\boldsymbol{Z}^{-1}$ yields $\boldsymbol{W} := \widehat{\boldsymbol{W}}\boldsymbol{Z}^{-1}$ which satisfies

$$[\boldsymbol{u}_{-k+1}, \dots, \boldsymbol{u}_{k+1}] = [\boldsymbol{A}_1, \dots, \boldsymbol{A}_k]\boldsymbol{W}.$$

In particular, numbering columns from $-k+1$ to $k-1$, shows that the ℓ-th column of $\boldsymbol{W}$ is a preimage of $\boldsymbol{u}_\ell$. (However, this preimage is under $[\boldsymbol{A}_1, \dots, \boldsymbol{A}_k]$, and hence not necessarily one an honest prover could have produced.) We only care about the preimage of $[\boldsymbol{u}_0] = [\boldsymbol{t}]$, hence the corresponding column yields a witness $\widetilde{\boldsymbol{w}}$ satisfying $[\boldsymbol{A}]\widetilde{\boldsymbol{w}} = [\widehat{\boldsymbol{t}}]$. This finishes the first part of the claim, i.e. unconditional extraction.

Now, we consider the structure of $\boldsymbol{W}$. A $\boldsymbol{W}$ obtained from an honest prover has zeroes in certain components $\boldsymbol{W}_{i,j}$. This is, because an honest $[\boldsymbol{u}_\ell]$ is the sum of the ℓ-th off-diagonal of $[\boldsymbol{Q}] := [\boldsymbol{A}_1, \dots, \boldsymbol{A}_k]^\top(\boldsymbol{w}_1, \dots, \boldsymbol{w}_k)$, c.f. Eq. (3.1). By using the structure of $\boldsymbol{w}_{\boldsymbol{x}}$, which was used to build $\widehat{\boldsymbol{W}}$ and from the structure of each $\boldsymbol{z}$ (namely $\boldsymbol{z} = \alpha(1, \rho, \rho^2, \dots, \rho^{k-1})$, we find as in [13, 16] that $\widehat{\boldsymbol{W}}$ must have the correct structure, or yields a non-trivial kernel element.

Finally, let us remark the following: Given k (independent) challenges, we can compute a candidate $\vec{\boldsymbol{w}}$ via $(\boldsymbol{w}_1, \dots, \boldsymbol{w}_k)\boldsymbol{Y} = (\widehat{\boldsymbol{w}}_1, \dots, \widehat{\boldsymbol{w}}_k)$. If this is a suitable witness, we have quick-extraction. If this fails, we need $2k-1$ transcripts, and *must* obtain a non-trivial kernel element, hence we have short-circuit extraction.[19] □

Remark 3.11. Let us sketch a possible generalisation of this argument (for which security claims can be adapted). Consider $\boldsymbol{P} \colon \mathbb{F}_p^k \otimes \mathbb{F}_p^k = \mathbb{F}_p^{k\times k} \to \mathbb{F}_p^m$, which maps $[\vec{\boldsymbol{A}}]^\top \vec{\boldsymbol{w}}^\top \in \mathbb{G}^{k\times k}$ to the "relevant part". Instead of $\boldsymbol{P} = \mathrm{id}$ (as in the motivation) or $\boldsymbol{P}$ being the off-diagonal sums as our instantiation, one can take any $\boldsymbol{P}$ which satisfies certain compatibility properties with (appropriately generalised) testing distribution $\chi_{\boldsymbol{P}}$, and soundness properties.

For example, we find that we can make do with $m = k-1$ messages $[\boldsymbol{u}_\ell]$ instead of $2(k-1)$. We achieve this by additionally summing the $\pm\ell$ off-diagonals. However, the compatible testing distributions have (in some sense) rank 2, because they are constant and symmetric on off-diagonals.

[18] This notational horror suggests that we should be using multilinear algebra, i.e. tensor product notation. Alas, to keep things "simple", we don't.

[19] Let $\vec{\boldsymbol{v}}^\top$ be $(\widehat{\boldsymbol{w}}_1, \dots, \widehat{\boldsymbol{w}}_k)\boldsymbol{Y}^{-1}$ as described. Suppose that by using all $2k-1$ transcripts we obtain a witness $\vec{\boldsymbol{w}} \neq \vec{\boldsymbol{v}}$ (in all other cases, we're done). Then from $\vec{\boldsymbol{w}}^\top \boldsymbol{y}_i = \widehat{\boldsymbol{w}}_i = \vec{\boldsymbol{v}}^\top \boldsymbol{y}_i$, we find that $\vec{\boldsymbol{w}}^\top \boldsymbol{Y} = \widehat{\boldsymbol{w}}_i = \vec{\boldsymbol{v}}^\top \boldsymbol{Y}$, and hence $\vec{\boldsymbol{w}} = \vec{\boldsymbol{v}}$. A contradiction.

For example, $\boldsymbol{x}^\top \otimes \boldsymbol{y}^\top + \boldsymbol{y}^\top \otimes \boldsymbol{x}^\top$. This results in having to carry out *two* follow-up arguments, one for $\boldsymbol{x}^\top[\boldsymbol{A}]\boldsymbol{w}^\top\boldsymbol{y}$ and one for $\boldsymbol{y}^\top[\boldsymbol{A}]\boldsymbol{w}^\top\boldsymbol{x}$. so the size reduction is $2/k$ (instead of $1/k$). Now every round sends $k-1$ messages (instead of $2(k-1)$), but also $k \geq 3$. Unfortunately, this has worse communication than the (simpler) approach we presented above.

3.4.3 Going zero-knowledge.

There are many variations for going zero-knowledge. The most straightforward one is to run Protocol 3.1 (Σ_{std}) and replace sending $\boldsymbol{z}$ by proving $\exists \boldsymbol{z}\colon [\boldsymbol{A}]\boldsymbol{z} = \beta[\boldsymbol{t}] + [\boldsymbol{a}]$ via $\mathsf{LMPA}_{\mathrm{noZK}}$. This gives a *proof* of knowledge, denoted $\mathsf{LMPA}_{\mathrm{simpleZK}}$, and is quite communication efficient. But computing $[\boldsymbol{A}]\boldsymbol{r}$ for random $\boldsymbol{r}$ is expensive. This is similar to [13, 16], where $\mathsf{LMPA}_{\mathrm{noZK}}$ was only used to save communication.

We achieve zero-knowledge more carefully. Instead of blinding the witness, we note that it is enough to blind the prover's responses. For this, a *logarithmic amount of randomness* suffices. This should make the prover more efficient.

Warm-up: Proving knowledge of opening of a commitment. For simplicity, we first sketch a protocol which assumes that $[\boldsymbol{A}] = [\boldsymbol{g}] \in \mathbb{G}^{1\times n}$, and $[\boldsymbol{g}]$ is a commitment key and $k = 2$. Thus, $[\boldsymbol{A}]$ has hard kernel assumption by construction. Later, we deal with $m > 1$ and adversarially chosen $[\boldsymbol{A}]$, which we actually solve with a different technique. But the techniques employed in this simple example help understanding the more complex technique, and they are reused and extended in Section 4.4.

So our current problem is to prove in *zero-knowledge* that $\exists \boldsymbol{w}\colon [\boldsymbol{g}]\boldsymbol{w} = [t]$. We will employ a masked version of $\mathsf{LMPA}_{\mathrm{noZK}}$, with judiciously chosen randomness $\boldsymbol{r}$, to achieve this. In particular, we do not pick $\boldsymbol{r} \leftarrow \mathbb{F}_p^n$. We pick $\boldsymbol{r}$ so that only logarithmically many r_i are non-zero. Thus, computing $[\boldsymbol{g}]\boldsymbol{r} = [a]$ is quite cheap (unlike in Protocol Σ_{std}). By the uniform-or-unique guideline, we want that each message $[\boldsymbol{u}_{\pm 1}]$ looks uniformly random. By analysing the recursive structure of $\mathsf{LMPA}_{\mathrm{noZK}}$, we can see we can achieve this by picking $r_i \leftarrow \mathbb{F}_p$ for $i \in \mathbb{M}_n \subseteq \{0, \ldots, n-1\}$ with $\mathbb{M}_n$ as defined below, and $r_i = 0$ else.[20]

Definition 3.12 (Masking sets). For some implicitly fixed k, we define the **masking (randomness) sets/spaces** $\mathbb{M}_n \subseteq \{0, \ldots, n-1\}$ (for $n = k^d$) by the formulas below. The set $\mathbb{M}_n$ describes the unit vectors of $\mathbb{F}_p^n$ (with zero-based indexing) which are used for random masking. We typically treat $\mathbb{M}_n$ as a subvector space of $\mathbb{F}_p^n$ (instead of explicitly referring to its span $\langle \boldsymbol{e}_i \mid i \in \mathbb{M}_n \rangle$).

- $\mathbb{M}_1 \coloneqq \{0\}$ and $\mathbb{M}_k \coloneqq \{0, \ldots, k-1\}$
- $\mathbb{M}_{k^d} \coloneqq \{\mathbb{M}_{k^{d-1}}\} \,\dot\cup\, \{ik^{d-1}, ik^{d-1} + 1 \mid i = 1, \ldots, k-1\}$ for $d \geq 2$.

See Fig. 2 for a pictorial description for $k = 2$.

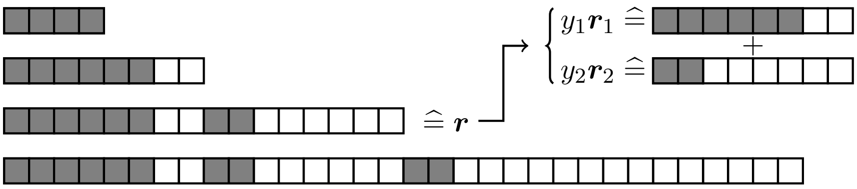
$y_1\boldsymbol{r}_1 \mathrel{\widehat{=}}$
$+$
$y_2\boldsymbol{r}_2 \mathrel{\widehat{=}}$
$\mathrel{\widehat{=}} \boldsymbol{r}$

Figure 2: *Left:* The (construction of the) masking randomness sets $\mathbb{M}_4$, $\mathbb{M}_8$, $\mathbb{M}_{16}$ and $\mathbb{M}_{32}$ (for $k = 2$). The squares denote the numbers $0, \ldots, n-1$ (or the respective basis vectors (with zero-based indexing)). *Right:* A demonstration of the "overlap" when a recursive step is applied to $\mathbb{M}_{16}$, i.e. $\widehat{\boldsymbol{r}} = y_1\boldsymbol{r}_1 + y_2\boldsymbol{r}_2$ is computed. Note that by removing two dark squares in the overlap (i.e. the randomness being "used up" in $[\boldsymbol{u}_{\pm 1}]$), the sum is still is randomised as $\mathbb{M}_8$. This "recursive property" is essential. The indices in $\mathbb{M}_n$ can also be constructed recursively via string concatenation: $m_{2n} = m_n|110^{n-2}$ and $m_1 = 1$, $m_2 = 11$.

[20] The masking sets $\mathbb{M}$ use zero-based indexing for convenience.

By the structure of the masking sets, we have that (for $k = 2$), if $\boldsymbol{r}$ is split into $\boldsymbol{r} = \left(\begin{smallmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \end{smallmatrix}\right)$ as in $\mathsf{LMPA}_{\mathrm{noZK}}$, then $[\boldsymbol{u}_{j-i}] = [\boldsymbol{g}_i]\boldsymbol{r}_j$ is uniformly distributed for $\boldsymbol{r} \leftarrow \mathbb{M}_n$. Moreover, $\widehat{\boldsymbol{r}} = y_1\boldsymbol{r}_1 + y_2\boldsymbol{r}_2$ is distributed like a fresh $\boldsymbol{r}' \leftarrow \mathbb{M}_{n/k}$. This holds even when considering the joint distribution $(\boldsymbol{u}_{-1}, \boldsymbol{u}_1, \widehat{\boldsymbol{r}})$. Thus, masking sets exhibit a useful recursive structure. There are some minor prerequisites to use the recursive structure, which we ignore for now.

Protocol 3.13. Let $\mathrm{crs} = [\boldsymbol{g}] \in \mathbb{G}^{1\times n}$ be a uniformly random commitment key (in particular, $[\boldsymbol{g}]$ has hard kernel relation under the DLOG assumption on $\mathbb{G}$.). The following is a protocol to prove $\exists \boldsymbol{w}\colon [\boldsymbol{g}]\boldsymbol{w} = [t]$. Let $\widetilde{\chi}_{2k-1}$ be a testing distribution as in Protocol 3.9. Common input is $(\mathrm{crs}, [t]) \in \mathbb{G}^{1\times n} \times \mathbb{G}$. We assume $n = k^d$. The prover's witness is some $\boldsymbol{w} \in \mathbb{F}_p^n$.

- $\mathcal{P} \to \mathcal{V}$: Choose $\boldsymbol{r} \leftarrow \mathbb{M}_n$. Compute $[a] = [\boldsymbol{g}]\boldsymbol{r}$. Send $[a]$.
- $\mathcal{V} \to \mathcal{P}$: Choose $\beta \leftarrow \chi^{(\beta)}$. Send β.
- $\mathcal{P} \leftrightarrow \mathcal{V}$: Let $\boldsymbol{z} \coloneqq \beta\boldsymbol{w} + \boldsymbol{r}$ and $[t'] \coloneqq \beta[t] + [a]$. Engage in $\mathsf{LMPA}_{\mathrm{noZK}}$ for $\exists \boldsymbol{z}\colon [\boldsymbol{g}]\boldsymbol{z} = [t']$.

It is clear that this protocol is correct. Short-circuit extraction follows easily as this is almost a sequential composition of Protocol Σ_{std} and $\mathsf{LMPA}_{\mathrm{noZK}}$. Thus, only zero-knowledge remains. For this, one should note that $\boldsymbol{z} = \beta\boldsymbol{w} + \boldsymbol{r}$ behaves like a linear combination throughout the protocol, because the reduced witness $\widehat{\boldsymbol{z}}$ is of the form $\beta\widehat{\boldsymbol{w}} + \widehat{\boldsymbol{r}}$. Indeed, we can view the protocol as a linear combination of protocols. Thus, to see that $[\boldsymbol{u}_{\pm\ell}]$ is uniformly distributed, we can focus our attention on $\boldsymbol{r}$ and its effect alone. As explained before, due to the form of $\mathbb{M}_n$ $(\widehat{\boldsymbol{r}}, [\boldsymbol{u}_{-1}], [\boldsymbol{u}_1])$ is uniformly distributed in $\mathbb{M}_{n/k} \times \mathbb{G} \times \mathbb{G}$. Thus, each iteration outputs uniformly distributed $[\boldsymbol{u}_{\pm 1}]$, and $\widehat{\boldsymbol{r}}$ distributed as $\widehat{\boldsymbol{r}} \leftarrow \mathbb{M}_{n/2}$. For the base case, we note that by construction, $\mathbb{M}_k = \{0, \dots, k-1\}$. Thus, $\boldsymbol{r} \leftarrow \mathbb{M}_k$ is uniformly random in $\mathbb{F}_p^k$, and hence $\beta\boldsymbol{w} + \widehat{\boldsymbol{r}}$ is uniformly random for $n \leq k$, perfectly hiding $\boldsymbol{w}$. In particular, the messages in the base case are uniformly random too. Since the uniform-or-unique property is satisfied, the zero-knowledge simulator can construct the transcript in reverse, as usual.

Difficulties arising from general $[\boldsymbol{A}]$. There are two main difficulties arising from general $[\boldsymbol{A}] \in \mathbb{G}^{m\times n}$. First, the higher dimension due to $m > 1$ makes masking sets as described not directly applicable anymore. Since $[\boldsymbol{u}_\ell] \in \mathbb{G}^m$, the prover now communicates mk elements, and hence we expect that $mk\log(n)$ random entries are necessary to randomise all of $[\boldsymbol{u}_\ell]$. Interestingly, the naive approach of using Protocol Σ_{std} shows that n random entries are sufficient. Note that $n < mk\log(n)$ is possible for large m. (In practice $mk\log(n) \ll n$.)

Second, we want to deal with *adversarial* $[\boldsymbol{A}]$. In the above sketch for zero-knowledge, we ignored a detail concerning the recursion. If it ever happens that in $[\boldsymbol{g}]$, for some $i \in \mathbb{M}_n$, the element $[g_i]$ is zero, the distribution of $(\widehat{\boldsymbol{r}}, [\boldsymbol{u}_{-1}], [\boldsymbol{u}_1])$ is skewed and zero-knowledge fails. Note that $[\boldsymbol{g}]$ is reduced in each statement, so this can happen randomly. Thus, even the naive reduction is only statistically zero-knowledge. If $[\boldsymbol{A}]$ is chosen adversarially, it may be so that this failure case always (or often) happens. Making the definition of $\mathbb{M}_n$ dynamic and depend on $[\boldsymbol{A}]$ is inconvenient and hard. Our choice is therefore to act "dually" to commitment-extension. Remember that a commitment-extension adds a row to $[\boldsymbol{A}]$ so that $[\boldsymbol{A}]$ is "computationally injective". In contrast, we will, very roughly, add columns to $[\boldsymbol{A}]$, to ensure that $[\boldsymbol{A}]$ is surjective. Our concrete approach is detailed below.

We remark that the naive approach to zero-knowledge for general $[\boldsymbol{A}]$ is a simple and viable option if the computational overhead is acceptable. Considering the computational costs of $\mathsf{LMPA}_{\mathrm{noZK}}$, this is often the case. Nevertheless, we demonstrate that, by applying our design guidelines, a more efficient, but more technical, conversion to zero-knowledge (with slightly larger proofs) is possible.[21]

Dealing with general $[\boldsymbol{A}]$. Our proof system separates the masking randomness from the actual witness and is a linear combination of multiple protocol instances of $\mathsf{LMPA}_{\mathrm{noZK}}$: The actual protocol for $[\boldsymbol{A}] \eqqcolon [\boldsymbol{H}^{(0)}]$, and protocols for $[\boldsymbol{H}^{(i)}]$, $i = 1, \dots, m$, where $[\boldsymbol{H}^{(i)}]$ essentially contains a Pedersen commitment key in the i-th row and is zero otherwise.

To keep things simple, we let $m = 1, k = 2$ in the following discussion. Intuitively, we want to run a "randomness-extended" protocol for $[\boldsymbol{B}] = [\boldsymbol{A}|\boldsymbol{H}]\left(\begin{smallmatrix} \boldsymbol{w} \\ \boldsymbol{r} \end{smallmatrix}\right)$. The intuition is that $\boldsymbol{r}$ will randomise

[21] It may be possibile to achieve smaller proof sizes. However, the current construction and security proofs are technical enough, and better proof size would likely complicate at least security proofs.

all $[u_{\pm 1}]$'s (because $[\boldsymbol{H}]$ is not adversarial). Unfortunately, this intuition is wrong: $[u_1] = [\boldsymbol{H}]\boldsymbol{w}$ is certainly not zero-knowledge. The problem is how $\mathsf{LMPA}_{\mathrm{noZK}}$ divides the statement. Appropriate shuffling of $[\boldsymbol{B}]$ and $\left(\begin{smallmatrix}\boldsymbol{w}\\ \boldsymbol{r}\end{smallmatrix}\right)$ would solve this. Instead, we work with a linear combination of $\mathsf{LMPA}_{\mathrm{noZK}}$ instances.

More precisely, we run *two* arguments, one for $[\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}']$ and one for $[\boldsymbol{H}]\boldsymbol{r} = [\boldsymbol{t}'']$. The messages $[u_{-1}]$ and $[u_1]$ are the sums of the messages which individual protocols would send, e.g. $[u_{-1}] = [\boldsymbol{A}_2]\boldsymbol{w}_1 + [\boldsymbol{H}_2]\boldsymbol{r}_1$. Concretely

$$\begin{bmatrix} u'_{-1} \\ u'_1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_1\boldsymbol{w}_2 \\ \boldsymbol{A}_2\boldsymbol{w}_1 \end{bmatrix}, \qquad \begin{bmatrix} u''_{-1} \\ u''_1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{H}_1\boldsymbol{r}_2 \\ \boldsymbol{H}_2\boldsymbol{r}_1 \end{bmatrix}, \qquad \begin{bmatrix} u_{-1} \\ u_1 \end{bmatrix} = \begin{bmatrix} u'_{-1} \\ u'_1 \end{bmatrix} + \begin{bmatrix} u''_{-1} \\ u''_1 \end{bmatrix}$$

This ensures that the $[u''_{\pm 1}]$ are uniformly random in every round, because $[u''_{\pm 1}]$ is. In the base case of the recursion, i.e. small n, the prover proves $[\boldsymbol{A}]\boldsymbol{w} + [\boldsymbol{H}]\boldsymbol{r} = [\boldsymbol{t}]$ in zero-knowledge, using (for concreteness) Protocol Σ_{std}.

To keep our protocol modular and comprehensible, we split it into two steps.

Protocol 3.14 ($\mathsf{LMPA}_{\mathrm{almSnd}}$). The following is a protocol to prove $\exists \boldsymbol{w}\colon [\boldsymbol{t}^{(0)}] = [\boldsymbol{A}]\boldsymbol{w}$, using testing distributions χ_{m+1} resp. $\widetilde{\chi}_{2k-1}$ (resp. $\chi^{(\beta)}$). with $\widetilde{\chi}_{2k-1}$, $\chi^{(\beta)}$ as in Protocol $\mathsf{LMPA}_{\mathrm{noZK}}$. Furthermore, we require that $\boldsymbol{x} \leftarrow \chi_{m+1}$ satisfies $x_i \neq 0$ for all i.

Common input is $([\boldsymbol{A}], [\boldsymbol{t}^{(0)}]) \in \mathbb{G}^{m \times n} \times \mathbb{G}^n$ and some $\boldsymbol{h} \in \mathbb{G}^n$ (typically derived from the CRS when this protocol is used as a subprotocol). We assume $n = k^\ell > 2k$. Moreover, we let $[\boldsymbol{H}^{(i)}] \in \mathbb{G}^{m \times n}$ for $i = 1, \dots, m$, be defined as the matrix with $[\boldsymbol{h}]$ in the i-th row and zeroes elsewhere, i.e. $[\boldsymbol{H}^{(i)}] = \boldsymbol{e}_i[\boldsymbol{h}]$. We use a superscript 0, e.g. $[\boldsymbol{H}^{(0)}] \coloneqq [\boldsymbol{A}]$, for terms related to $[\boldsymbol{A}]$. The prover's witness is some $\boldsymbol{w} \in \mathbb{F}_p^n$ (also written $\boldsymbol{r}^{(0)}$).

- $\mathcal{P} \to \mathcal{V}$: (Step 1: Prepare masking.) Pick $\boldsymbol{r}^{(i)} \leftarrow \mathbb{M}_n \leq \mathbb{F}_p^n$ and compute $[\boldsymbol{t}^{(i)}] = [\boldsymbol{H}^{(i)}]\boldsymbol{r}^{(i)}$. Send $[\boldsymbol{t}^{(i)}]$ for $i = 1, \dots, m$.
- $\mathcal{V} \to \mathcal{P}$: (Step 2: Random linear combination.) Pick and send $\boldsymbol{x} \leftarrow \chi_{m+1}$. The statement we prove is now effectively

  $$[\boldsymbol{A}|\boldsymbol{H}^{(1)}|\dots|\boldsymbol{H}^{(m)}]\begin{pmatrix} x_0\boldsymbol{w} \\ x_1\boldsymbol{r}^{(1)} \\ \vdots \end{pmatrix} = [\boldsymbol{t}] \coloneqq \sum_i x_i[\boldsymbol{t}^{(i)}].$$

  For simplicity, the prover redefines $\boldsymbol{r}^{(i)} \coloneqq x_i\boldsymbol{r}^{(i)}$ for $i = 0, \dots, m$.
- $\mathcal{P} \to \mathcal{V}$: (Step 3: Begin the shrinking AoK.) Let $[\boldsymbol{H}^{(i)}] = [\boldsymbol{H}_1^{(i)}, \dots, \boldsymbol{H}_k^{(i)}]$ with $\boldsymbol{H}_j^{(i)} \in \mathbb{G}^{m \times n/k}$. Compute $[\boldsymbol{u}_\ell] = \sum_{i=0}^m [\boldsymbol{u}_\ell^{(i)}]$, where $[\boldsymbol{u}_\ell^{(i)}]$ is computed as usual, i.e. $[\boldsymbol{u}_\ell^{(i)}] = \sum_{j-i=\ell}[\boldsymbol{H}_\ell^{(i)}]\boldsymbol{r}_\ell^{(i)}$. Send $[\boldsymbol{u}_\ell]$ for $\ell = \pm 1, \dots, \pm(k-1)$.
- $\mathcal{V} \to \mathcal{P}$: Pick $\boldsymbol{z} \leftarrow \widetilde{\chi}_{2k-1}$ (with associated $\boldsymbol{x}, \boldsymbol{y}$). Send $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$.
- $\mathcal{P} \to \mathcal{V}$: As in $\mathsf{LMPA}_{\mathrm{noZK}}$, compute $\boldsymbol{w} = \boldsymbol{x}^\top \vec{\boldsymbol{w}} = \sum_j x_j \boldsymbol{w}_i$ and $\widehat{\boldsymbol{r}}^{(i)} = \boldsymbol{x}^\top \vec{\boldsymbol{r}}^{(i)} = \sum_j x_j \boldsymbol{r}_j^{(i)}$ and $[\widehat{\boldsymbol{A}}] = \boldsymbol{x}^\top[\vec{\boldsymbol{A}}] = \sum_j x_j[\boldsymbol{A}_j]$, $[\widehat{\boldsymbol{H}}^{(i)}] = \sum_j x_j[\boldsymbol{H}_j^{(i)}]$, and $[\boldsymbol{t}] = \boldsymbol{z}^\top\boldsymbol{u} = \sum_\ell z_j\boldsymbol{u}_\ell$, for the reduced statement (which $\mathcal{V}$ also computes).

  If $n > 2k$, engage recursively in the AoK for this statement, i.e. goto Step 3. If $n \leq 2k$, engage in (for concreteness) Protocol Σ_{std} to prove the statement.

It is easy to check that Protocol 3.14 is complete.

Lemma 3.15. *Protocol* $\mathsf{LMPA}_{\mathrm{almSnd}}$ *has μ-special soundness (with $\mu = (m+1, 2k, \dots, 2k, 2)$) for finding a preimage $\vec{\boldsymbol{v}} \in (\mathbb{F}_p^n)^m$ (unconditionally) with $[\boldsymbol{A}|\boldsymbol{H}^{(1)}|\dots|\boldsymbol{H}^{(m)}]\left(\begin{smallmatrix}\boldsymbol{v}_0\\ \cdots\\ \boldsymbol{v}_m\end{smallmatrix}\right) = [\boldsymbol{t}^{(0)}]$, or a non-trivial kernel element of $[\boldsymbol{A}|\boldsymbol{H}'^{(1)}|\dots|\boldsymbol{H}'^{(m)}]$. Here, $[\boldsymbol{H}'^{(i)}]$ consists only of the non-zero components of $[\boldsymbol{H}^{(i)}]$. (It is easy to find non-trivial kernel elements if $[\boldsymbol{h}]$ has zeroes, so we exclude them, c.f. Remark 2.2.)*

The protocol inherits short-circuit extraction with $\mu' = (m+1, k, \dots, k, 2)$.

Note Lemma 3.15 *does not assert* a witness $\boldsymbol{w} \in \mathbb{F}_p^n$ for $[\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}^{(0)}]$. That will be assured in follow-up step.

Proof. We only sketch the proof. Let $tree_\mu$ be a good μ-tree of transcripts. First of all, we can extract the base subprotocol of Step 3. Using these witnesses, we can extract the linearly combined argument essentially as in Lemma 3.10.[22]

Now we extract Step 2. From Step 3, we have $m+1$ preimages $\vec{\boldsymbol{v}}_i \in (\mathbb{F}_p^m)^n$ with $[\boldsymbol{A}|\boldsymbol{H}^{(1)}|\dots|\boldsymbol{H}^{(m)}]\vec{\boldsymbol{v}}_i = [\boldsymbol{T}]\boldsymbol{x}_i$ where $[\boldsymbol{T}] = [\boldsymbol{t}^{(0)},\dots,\boldsymbol{t}^{(m)}]$. Arrange matrices $\boldsymbol{V} = (\vec{\boldsymbol{v}}_0,\dots,\vec{\boldsymbol{v}}_m)$ and $\boldsymbol{X}$ as usual. ($\boldsymbol{V}$ corresponds to $\widehat{\boldsymbol{W}}$ from Lemma 3.10.) We find $[\boldsymbol{A}|\boldsymbol{H}^{(1)}|\dots|\boldsymbol{H}^{(m)}]\boldsymbol{V}_i = [\boldsymbol{t}]\boldsymbol{X}$. Multiplying with $\boldsymbol{X}^{-1}$, we find preimages for each $[\boldsymbol{t}^{(i)}]$, in particular a preimage for $[\boldsymbol{t}^{(0)}]$. □

To prove zero-knowledge of Protocol $\mathsf{LMPA}_{\mathrm{almSnd}}$, we first show that the prover's messages $[\boldsymbol{u}_\ell]$ in the recursive steps are almost always uniformly distributed. This yields statistical HVZK via straightforward simulation.

As a preparation, note following (easy) linear algebra facts:

Lemma 3.16. *Let $\mathbb{V} \cong \mathbb{F}_p^n$ and $\mathbb{W} \cong \mathbb{F}_p^m$ be some vector spaces. Let $\boldsymbol{M}\colon \mathbb{V} \to \mathbb{W}$ be a linear map (i.e. a matrix $\boldsymbol{M} \in \mathbb{F}_p^{m\times n}$). Then $\boldsymbol{r} \mapsto \boldsymbol{M}\boldsymbol{r}$ for $\boldsymbol{r} \leftarrow \mathbb{V}$ uniformly random induces the uniform distribution on $\mathbb{W}$ if and only if $\boldsymbol{M}$ is surjective. (Equivalently, if the rows of $\boldsymbol{M}$ (as a matrix) are linearly independent.)*

Lemma 3.17. *Consider Protocol 3.14 ($\mathsf{LMPA}_{\mathrm{almSnd}}$). Suppose that (at least) all components of $[\boldsymbol{h}]$ in $\mathbb{M}_n$ are distributed uniformly random (and the rest may be 0). Suppose that for any $\boldsymbol{x} \leftarrow \chi_{m+1}$ we have $x_i \neq 0$ for all i.*

Then, in this argument system, with probability about $O(\log_k(n)k)/p$ the vector $\boldsymbol{U}$ consisting of messages $[\boldsymbol{u}_\ell]$ of all *recursive rounds is uniformly random. The randomness is over $[\boldsymbol{h}]$, the challenges and the prover's randomness.*

We give a short proof intuition for the case $k = 2$, $m = 1$. So we have $[\boldsymbol{A}], [\boldsymbol{H}] \in \mathbb{G}^{1\times n}$. Intuitively, we need 2 $\mathbb{F}_p$-elements of randomness in each round to mask $[u_{\pm 1}]$. Moreover, these two terms of randomness must be split so that one is in the first half $\boldsymbol{r}_1$ of $\boldsymbol{r}$, and one in the second half $\boldsymbol{r}_2$, since $[u_{j-i}] = [\boldsymbol{H}_i]\boldsymbol{r}_j$. The masking sets $\mathbb{M}_n$ are built exactly as such, see Fig. 2. Moreover, to allow inductive reasoning, the masking sets are built in such a way that even when "removing" two terms of randomness (say $\boldsymbol{r}_{1,0}$ and $\boldsymbol{r}_{2,1}$), the sum $\boldsymbol{r}_1' + \boldsymbol{r}_2'$ is distributed according to $\mathbb{M}_{n/k}$. Evidently, we need $x_i \neq 0$ to prevent loss of randomness by multiplication with 0. More precisely, we want surjectivity of the "transition map", $\begin{pmatrix} x_1 \operatorname{id}_n & x_2 \operatorname{id}_n \\ \boldsymbol{H}_2 & 0 \\ 0 & \boldsymbol{H}_2 \end{pmatrix} \begin{pmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \end{pmatrix} = \begin{pmatrix} \widehat{\boldsymbol{r}} \\ u''_{-1} \\ u''_1 \end{pmatrix}$ when restricted to $\mathbb{M}_{2n} \leq \mathbb{F}_p^{2n}$ in each step. A full proof follows.

Proof. It suffices to consider $m = 1$, because the matrices $\boldsymbol{H}^{(i)}$ are constructed such that they mask the i-th row only (they are zero in all other rows). Evidently, there is also no "interference" between rows in the protocol because $\widehat{\boldsymbol{H}}^{(i)}$ is again only non-zero in the i-th row ($i \neq 0$). Consequently, we consider $[\boldsymbol{A}], [\boldsymbol{H}] \in \mathbb{G}^{1\times n}$ and drop the superscripts.

As a first step, we consider the case where the masking randomness is simply taken from $\mathbb{F}_p^n$ uniformly, i.e. $\mathbb{M}_n = \{0,\dots,n-1\}$. (Note that we use zero-based indexing for the masking randomness.)

[22] Indeed, after suitably permuting the columns of $[\boldsymbol{A}|\boldsymbol{H}^{(1)}|\dots|\boldsymbol{H}^{(m)}]$, witness, and randomness, the exact same reasoning as in Lemma 3.10 works for the recursive step.

By construction, we have

$$\underbrace{\begin{pmatrix} x_1\,\mathrm{id} & x_2\,\mathrm{id} & \dots & x_{k-1}\,\mathrm{id} & x_k\,\mathrm{id} \\ \boldsymbol{H}_k & 0 & \dots & 0 & 0 \\ \boldsymbol{H}_{k-1} & \boldsymbol{H}_k & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{H}_1 & \boldsymbol{H}_2 & \dots & \boldsymbol{H}_{k-1} & \boldsymbol{H}_k \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \boldsymbol{H}_1 & \boldsymbol{H}_2 \\ 0 & 0 & \dots & 0 & \boldsymbol{H}_1 \end{pmatrix}}_{=:\boldsymbol{M}'} \begin{pmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \\ \vdots \\ \boldsymbol{r}_{k-1} \\ \boldsymbol{r}_k \end{pmatrix} = \begin{pmatrix} \widehat{\boldsymbol{r}} \\ u_{1-k} \\ u_{2-k} \\ \vdots \\ u_0 \\ \vdots \\ u_{k-2} \\ u_{k-1} \end{pmatrix} \tag{3.2}$$

Let $\boldsymbol{M}$ be the matrix where the row $(\boldsymbol{H}_1, \dots, \boldsymbol{H}_k)$ corresponding to $\boldsymbol{u}_0$ is removed. Then, by Lemma 3.16, it suffices to show that

- $\boldsymbol{M}$ is surjective, i.e. has independent rows. Then we know that u_ℓ (for $\ell \neq 0$) and $\widehat{\boldsymbol{r}}$ are uniformly distributed.
- This surjectivity is "preserved by reduction", i.e. even after "application" of $\boldsymbol{x}$, for $\widehat{\boldsymbol{H}}$ the respective $\boldsymbol{M}$ is again surjective (with overwhelming probability).

In fact, we want a bit more: We want to bound the (worst-case) probability that, *conditioned* on some fixed $\boldsymbol{H}_1 \neq \boldsymbol{0}$, this matrix is surjective. Note that surjectivity certainly implies $\boldsymbol{H}_1 \neq \boldsymbol{0}$. Thus, restricting to $\boldsymbol{H}_1 \neq \boldsymbol{0}$ is already covered by restricting to surjective matrices only. The probability that, for fixed $\boldsymbol{H}_1 \neq \boldsymbol{0}$, the matrix fails to be surjective with probability $\leq O(n)/p$, can be checked by many means.[23] However, a formal derivation for the general case is straightforward, but tedious. We therefore specialise to the case $k = 2$. (But we still write k instead of 2 where it makes not much difference.) We consider

$$\begin{pmatrix} x_1\,\mathrm{id} & x_2\,\mathrm{id} \\ \boldsymbol{H}_2 & 0 \\ 0 & \boldsymbol{H}_1 \end{pmatrix} \xRightarrow{\text{Gauß}} \begin{pmatrix} x_1\,\mathrm{id} & x_2\,\mathrm{id} \\ 0 & -\frac{x_2}{x_1}\boldsymbol{H}_2 \\ 0 & \boldsymbol{H}_1 \end{pmatrix} \quad \text{and note that} \quad \begin{pmatrix} -\frac{x_2}{x_1}\boldsymbol{H}_2 \\ \boldsymbol{H}_1 \end{pmatrix}$$

evidently has full rank if $\boldsymbol{H}_i$ are linearly independent. Since $\boldsymbol{H}_i$ are random vectors of dimension ≥ 2, this happens with overwheming probability. More precisely, it fails with probability $1 - (1 - p^{-1})(1 - p^{-2})$. Conditioned on $\boldsymbol{H}_1 \neq \boldsymbol{0}$, it fails with probability p^{-1}.

Now we argue why the recursions pose no problem. This follows because: $\mathbb{M}_{n/2} \cap \mathbb{M}_n = \mathbb{M}_{n/2}$ by assumption.[24] Thus, $\widehat{\boldsymbol{r}}$ lies in the "correct" subspace of $\mathbb{F}_p^{n/2}$, namely the masking space for $\mathbb{M}_{n/2}$. Furthermore, the distribution of $\widehat{\boldsymbol{r}}$ is uniformly random (in that subspace), even conditioned on $u_{1-k}, \dots, u_{k-1}$. Thus, we can recursively apply our reasoning to find that all $u_{1-k}, \dots, u_{k-1}$ are uniformly random in all rounds. More precisely, we use that *conditioned* on a fixed $\boldsymbol{H}_1$, we obtained these properties. Thus, $\widehat{\boldsymbol{H}} = \sum x_i \boldsymbol{H}_i$ is (almost) uniformly random, even conditioned on $\boldsymbol{H}_i$ for $i > 1$. This holds because $\boldsymbol{H}_1$ is still uniformly random and "untouched" (conditioned on not being zero). This allows us to "restart" our reasoning. (We only guarantee that $\widehat{\boldsymbol{H}}$ is statistically close to uniform. But this is evidently good enough.)

Since the matrix $\boldsymbol{M}$ has dimensions $n \times n/k + 2(k-1)$ we need $n \geq n/k + 2(k-1)$, i.e. $n \geq 2k$ for it to possibly be surjective. Thus, stopping our recursion there works fine.

Now, let us consider the case where $\mathbb{M}_n$ as in Definition 3.12. Note that $\mathbb{M}_n$ has following structural properties (for general k):

- $\mathbb{M}_{k^\ell,i} := \mathbb{M}_{k^\ell} \cap \{ik^{\ell-1}, \dots, (i+1)k^{\ell-1} - 1\}$ satisfies $\mathbb{M}_{n,0} = \mathbb{M}_{n/k}$ and $\mathbb{M}_{k^\ell,i} = Q_{\ell-1}$ with $Q_\ell := \{ik^\ell + \delta \mid i = 0, \dots, k-1; \delta = 0, 1\}$.

[23] We note that it is vital that $\boldsymbol{H}_i$ is a vector, not a scalar. Obviously, doing the recursion in the base case $n = k$ (with scalar $\boldsymbol{H}_i$) yields a non-surjective $\boldsymbol{M}$.

[24] Remember that we are working with $\mathbb{M}_n = \{0, \dots, n-1\}$ at the moment. But the same applies to general $\mathbb{M}_n$.

- Consequently, modulo $n/k = k^{\ell-1}$, $\mathbb{M}_n$ maps (surjectively) onto $\mathbb{M}_{n/k}$.

Now we analyse $\boldsymbol{M}$ in this setting. First of all, we remove the columns not in $\mathbb{M}_n$ from $\boldsymbol{M}$. (The respective columns are "useless" for randomisation, since $\boldsymbol{r}$ is only non-zero for components in $\mathbb{M}_n$.) Second, we remove the rows not in $\mathbb{M}_{n/k}$ from the upper part of $\boldsymbol{M}$ (corresponding to $\widehat{\boldsymbol{r}}$). (Again, since $\widehat{\boldsymbol{r}}$ need only be non-zero ony for components in $\mathbb{M}_{n/k}$, we only need surjectivity in those components.) Note that now all remaining components of $\boldsymbol{r}$ and consequently of $\boldsymbol{H}$ were chosen uniformly at random.

Now, only the relevant portions of $\boldsymbol{M}$ remain. Note that $\boldsymbol{M}$ has now dimensions $\dim(\mathbb{M}_n) \times (\dim(\mathbb{M}_{n/k}) + 2(k-1))$. Since $\dim(\mathbb{M}_n) = \dim(\mathbb{M}_{n/k}) + 2(k-1)$, we see that $\boldsymbol{M}$ is in fact a square matrix.

We now split $\boldsymbol{r}_1$ into $(\boldsymbol{r}_1', \boldsymbol{r}_1'')$, the $\mathbb{M}_{n/k}$ components and those in $(\mathbb{M}_n \cap \{0, \dots, n/k-1\}) \setminus \mathbb{M}_{n/k}$. Using the same notation for matrices $\boldsymbol{H}_i$, we see that $\boldsymbol{H}_i'$ for $i \neq 1$ is zero when restricted to $\mathbb{M}_{n,0} = \mathbb{M}_n \cap \{0, \dots, n/k-1\} = \mathbb{M}_{n/k}$. On the other hand, $\boldsymbol{H}_i''$ is uniformly random. For $\boldsymbol{H}_1$, we see that $\boldsymbol{H}_1'$ and $\boldsymbol{H}_1''$ are uniformly random. Note that we use the structure of $\mathbb{M}_n$ here. If we reorder (the columns of) $\mathrm{id}_{\mathbb{M}_{n/k}}$ to $(\boldsymbol{P}'|\boldsymbol{P}'')$, where $\boldsymbol{P}' \in \mathbb{F}_p^{\dim(\mathbb{M}_{n/k})\times 2}$ corresponds to the components in Q_n and $\boldsymbol{P}''$ to $\mathbb{M}_{n/k} \setminus Q_n$, we find:

$$\underbrace{\begin{pmatrix} x_1\boldsymbol{P}' & x_1\boldsymbol{P}'' & x_2\boldsymbol{P}'' & \dots & x_{k-1}\boldsymbol{P}'' & x_k\boldsymbol{P}'' \\ 0 & \boldsymbol{H}_k & 0 & \dots & 0 & 0 \\ 0 & \boldsymbol{H}_{k-1} & \boldsymbol{H}_k & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \boldsymbol{H}_2 & \boldsymbol{H}_3 & \dots & \boldsymbol{H}_k & 0 \\ 0 & 0 & \boldsymbol{H}_1 & \dots & \boldsymbol{H}_{k-2} & \boldsymbol{H}_{k-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \dots & \boldsymbol{H}_1 & \boldsymbol{H}_2 \\ 0 & 0 & \dots & \dots & 0 & \boldsymbol{H}_1 \end{pmatrix}}_{=:\boldsymbol{M}} \begin{pmatrix} \boldsymbol{r}_1' \\ \boldsymbol{r}_1'' \\ \boldsymbol{r}_2'' \\ \vdots \\ \boldsymbol{r}_{k-1}'' \\ \boldsymbol{r}_k'' \end{pmatrix} = \begin{pmatrix} \widehat{\boldsymbol{r}} \\ u_{1-k} \\ u_{2-k} \\ \vdots \\ u_{-1} \\ u_1 \\ \vdots \\ u_{k-2} \\ u_{k-1} \end{pmatrix}$$

Here, to reduce visual noise, we omitted all primes, i.e. we write $\boldsymbol{H}_i$ instead of the formally correct $\boldsymbol{H}_i''$ everywhere. By suitable reordering of rows (and columns) related to $\boldsymbol{P}'$ and $\boldsymbol{P}''$, we obtain

$$\begin{pmatrix} x_1\,\mathrm{id}_\ell & 0 & 0 & \dots & 0 & 0 \\ 0 & x_1\,\mathrm{id}_2 & x_2\,\mathrm{id}_2 & \dots & x_{k-1}\,\mathrm{id}_2 & x_k\,\mathrm{id}_2 \\ 0 & \boldsymbol{H}_k & 0 & \dots & 0 & 0 \\ 0 & \boldsymbol{H}_{k-1} & \boldsymbol{H}_k & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \boldsymbol{H}_2 & \boldsymbol{H}_3 & \dots & \boldsymbol{H}_k & 0 \\ 0 & 0 & \boldsymbol{H}_1 & \dots & \boldsymbol{H}_{k-2} & \boldsymbol{H}_{k-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \dots & \boldsymbol{H}_1 & \boldsymbol{H}_2 \\ 0 & 0 & \dots & \dots & 0 & \boldsymbol{H}_1 \end{pmatrix} \tag{3.3}$$

where $\ell = \dim(\mathbb{M}_{n/k}) - 2$. The removal of the rows corresponding to $\mathbb{M}_{n/k} \setminus Q_n$ (i.e. id_ℓ in the above) means that all zero-rows in $\boldsymbol{P}$ are removed. What remains are blocks with $\boldsymbol{P}'' = \mathrm{id}_{2\times 2}$. Thus, we are in fact in exactly the same setting as before, namely Eq. (3.2)! Since $\boldsymbol{H}_i''$ are uniformly random, the same argument works.

With this, all of our reasoning before applies. In fact, things get simpler because $\boldsymbol{H}_i'' \in \mathbb{G}^{1\times 2}$ by construction. Thus, the probability that $\boldsymbol{H}_1'' \neq \boldsymbol{0}$ is $1 - p^{-2}$ (for uniform $\boldsymbol{H}_1''$). Conveniently, $\boldsymbol{H}_1'$ does not even appear in our reasoning! Finally note that $\widehat{\boldsymbol{H}} = \sum_i x_i \boldsymbol{H}_i$ is distributed statistically close to uniform (over $\mathbb{M}_{n/k}$), because $\boldsymbol{H}_1'$ is. Hence we can apply our reasoning recursively, as before.

We see that the probability that the matrix in Eq. (3.3) is not invertible is about $2(k-1)/p$ (or less). This follows because by assumption all x_i are non-zero. Thus only the $\boldsymbol{H}_i''$ part has to be "checked". Again, after some tedious computations, one obtains $O(2(k-1))/p$ as a bound of the failure probability.

Finally, after $\log(n)$ recursions, we find a combined failure probability of about $2k\log_k(n)(1/p + 1/p^2) \leq 2\log_k(n)/p$ (by the union bound).[25] (The $1/p^2$ term is from the condition that $\boldsymbol{H}_1'' \neq 0$.) □

For $k = 2$, one can easily test if $\boldsymbol{M}$ is surjective, and could therefore sacrifice perfect correctness to gain perfect zero-knowledge by aborting bad executions.

Lemma 3.18. $\mathsf{LMPA}_{\mathrm{ZK}}$*is ε-statistical zero-knowledge for $\varepsilon \in O(\log_k(n)k)/p$.*

We sketch HVZK simulation: For a recursive step, the HVZK simulator picks $[\boldsymbol{u}_\ell] \leftarrow \mathbb{G}^m$ for $\ell \neq 0$ and computes the uniquely defined $[\boldsymbol{u}_0]$ which makes the verifier accept that round. For Step 1 note that $[\boldsymbol{t}^{(i)}] = [\boldsymbol{e}_i t^{(i)}]$ ($i \neq 0$) and hence $[\boldsymbol{t}^{(0)}]$ and $[\boldsymbol{t}]$ (which is $[\boldsymbol{u}_0]$ of the last recursion) uniquely define all $[\boldsymbol{t}^{(i)}]$. Since the messages $[\boldsymbol{u}_\ell]$ are uniformly distributed in an honest execution with probability $O(\log_k(n)k)/p$, our claim follows. A more detailed proof follows.

Proof. We can assume that all messages $[\boldsymbol{u}_\ell]$ in the protocol are uniformly random, in the sense of Lemma 3.17 and simulate in this case. By Lemma 3.17, this fails with probability at most $O(\log_k(n)k)/p$, so our simulation will be statistically close to an honest execution. As usual, the simulator picks all challenges beforehand. By using the simulator of the base subprotocol, i.e. Protocol Σ_{std}, we only need to show how to simulate one round of reduction, and how to simulate the initial masking step.

Let us consider how to simulate one recursion step. We are given $[\widehat{\boldsymbol{H}^{(i)}}]$ and $[\widehat{\boldsymbol{t}}]$ (and challenge $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$). To simulate, we pick uniformly random $[\boldsymbol{u}_\ell]$ for $\ell = \pm 1, \dots, \pm(k-1)$ for this round. We set $[\boldsymbol{t}] = [\boldsymbol{u}_0] = [\widehat{\boldsymbol{t}}] - \sum_{\ell \neq 0} z_\ell [\boldsymbol{u}_\ell]$, which is the unique $[\boldsymbol{t}]$ which would make this an accepting transcript. The simulator's messages $[\boldsymbol{u}_\ell]$ are all uniformly random, hence just as in an honest transcript.

For the initial masking step we need to find $[\boldsymbol{t}^{(i)}] \in \mathbb{G}^m$ for $i = 1, \dots, m$ such that $[\boldsymbol{t}] = \sum_{i=0}^m x_i [\boldsymbol{t}^{(i)}]$. Since $[\boldsymbol{t}^{(i)}]$ is zero, except in the i-th component (for $i \neq 0$), and $[\boldsymbol{t}^{(0)}]$ is fixed in the statement, this already uniquely defines $[\boldsymbol{t}^{(i)}]$ for all i. (And they are efficiently computable.) Thus, simulation of this step is perfect. □

Now, we finish the protocol and ensure that extraction yields a witness $\boldsymbol{w}$ for $[\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$ as we desired. For this, we use a dual testing distribution to ensure $\boldsymbol{v}_i \overset{!}{=} \boldsymbol{0}$ for $i \geq 1$ (with notation as in Lemma 3.15).

Protocol 3.19 ($\mathsf{LMPA}_{\mathrm{ZK}}$). The following is a protocol to prove $\exists \boldsymbol{w}\colon [\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$. We use Protocol 3.14 ($\mathsf{LMPA}_{\mathrm{almSnd}}$) as a subprotocol with the same testing distributions χ_{m+1} resp. $\widetilde{\chi}_{2k-1}$ (resp. $\chi^{(\beta)}$). By $\chi^{\vee}_{\dim(\mathbb{M}_n)+1}$, we refer to the dual testing distribution of $\chi_{\dim(\mathbb{M}_n)+1}$ as in Definition 2.16. In particular, we require that the first component x_0 of $\boldsymbol{x} \leftarrow \chi_{\dim(\mathbb{M}_n)+1}$ is always 1.

Common input is $([\boldsymbol{A}], [\boldsymbol{t}]) \in \mathbb{G}^{m \times n} \times \mathbb{G}^n$ We assume $n = k^\ell > 2k$. The prover's witness is some $\boldsymbol{w} \in \mathbb{F}_p^n$ (also written $\boldsymbol{r}^{(0)}$). The CRS crs contains randomly (independently) chosen $[\boldsymbol{q}] \leftarrow \mathbb{G}^{1 \times \dim(\mathbb{M}_n)+1}$.

- $\mathcal{V} \to \mathcal{P}$: (Step 0: Setup of a "new" crs.) $\mathcal{V}$ picks and sends $\boldsymbol{M} \coloneqq \boldsymbol{M}_{\boldsymbol{x}} \leftarrow \chi^{\vee}_{\dim(\mathbb{M}_n)+1}$ (as described in Definition 2.16).
- Both parties compute $[\widetilde{\boldsymbol{h}}] \coloneqq [\boldsymbol{q}]\boldsymbol{M} \in \mathbb{G}^{1 \times \dim(\mathbb{M}_n)}$. They define $[\boldsymbol{h}] \in \mathbb{G}^n$ so that the components $\mathbb{M}_n \subseteq \{0, \dots, n-1\}$ of $[\boldsymbol{h}]$ correspond to $[\widetilde{\boldsymbol{h}}]$ (in order). All components of $[\boldsymbol{h}]$ not in $\mathbb{M}_n$ are set to 0. See Fig. 2 for a pictorial description of (non-)zero components of $[\boldsymbol{h}]$ in case of $k = 2$.
- $\mathcal{P} \leftrightarrow \mathcal{V}$: Engage in Protocol $\mathsf{LMPA}_{\mathrm{almSnd}}$ for $\exists \boldsymbol{w}\colon [\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$ with parameters (in particular $[\boldsymbol{h}]$) as above.

Lemma 3.20. *Protocol* $\mathsf{LMPA}_{\mathrm{ZK}}$ *has μ-special soundness (with $\mu = (\dim(\mathbb{M}_n)+1, m+1, 2k, \dots, 2k, 2)$) for finding a witness $\boldsymbol{w} \in \mathbb{F}_p^n$ with $[\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$, or a non-trivial kernel element of $[\boldsymbol{A}|\boldsymbol{e}_1^\top \boldsymbol{q}|\dots|\boldsymbol{e}_m^\top \boldsymbol{q}]$ (equivalently $[\boldsymbol{A}|\operatorname{diag}(\boldsymbol{q}, \dots, \boldsymbol{q})]$).*

The protocol has short-circuit extraction with $\mu' = (1, m+1, k, \dots, k, 2)$.

There are reasons to suspect that $\mathsf{LMPA}_{\mathrm{ZK}}$ may have unconditional extraction, i.e. it is *proof* of knowledge. But we could not (dis)prove it yet. Compare to Question 3.7.

[25] Including that recursive steps are only statistically close to the desired uniform distribution.

Proof. By extracting $\mathsf{LMPA}_{\mathrm{almSnd}}$ i.e. applying Lemma 3.15, we can find preimages $\vec{\boldsymbol{u}} \in (\mathbb{F}_p^n)^{m+1}$. (Also, we inherit short-circuit and unconditional extraction.) Let $[\boldsymbol{h}]$ and $[\boldsymbol{H}^{(i)}] = \boldsymbol{e}_i[\boldsymbol{h}]$ be as constructed in the protocols.

For simplicity, we first consider the case $m = 1$ and remove all 0-columns of $[\boldsymbol{H}]$. In other words, we consider $[\boldsymbol{A}|\boldsymbol{qM}] \in \mathbb{G}^{1\times n+\dim(\mathbb{M})}$.

We know (i.e. extracted) some $\boldsymbol{w} \in \mathbb{F}_p^n, \boldsymbol{v} \in \mathbb{F}_p^{\dim(\mathbb{M})}$ such that $[\boldsymbol{A}]\boldsymbol{w} + [\boldsymbol{H}]\boldsymbol{v} = [t]$. We have to show that $[\boldsymbol{A}]\boldsymbol{w} = [t]$, or we find a non-trivial element in the kernel of $[\boldsymbol{A}|\boldsymbol{q}]$. In the case that $[\boldsymbol{H}]\boldsymbol{v} = 0$, $\boldsymbol{w}$ is the witness we want. So suppose that $[\boldsymbol{H}]\boldsymbol{v} \neq 0$. In that case, we guarantee short-circuit extraction. So, suppose we have $\dim(\mathbb{M}) + 1$ transcripts with "independent" challenge matrices $\boldsymbol{M}_i$. (Remember that this means $\bigcap_{i=0}^{\dim(\mathbb{M})} \operatorname{im}(\boldsymbol{M}_i) = \{0\}$, which is equivalent to $\boldsymbol{x}_i$ being linearly independent since $\boldsymbol{M}_i = \boldsymbol{M}_{\boldsymbol{x}_i}$.) By subtracting the 0-th witness from the i-th witness, we find

$$[\boldsymbol{A}](\boldsymbol{w}_i - \boldsymbol{w}_0) + [\boldsymbol{q}](\boldsymbol{M}_i\boldsymbol{v}_i - \boldsymbol{M}_0\boldsymbol{v}_0) = 0.$$

Thus, if $\boldsymbol{M}_i\boldsymbol{v}_i - \boldsymbol{M}_0\boldsymbol{v}_0 \neq \boldsymbol{0}$, we obtain a non-trivial kernel element. The only case where we do not obtain a non-trivial kernel element of $[\boldsymbol{A}|\boldsymbol{qM}]$ is, if for all i we have $\boldsymbol{M}_i\boldsymbol{v}_i = \boldsymbol{M}_0\boldsymbol{v}_0 =: \boldsymbol{u}$. However, this implies that $\boldsymbol{u} \in \bigcap_{i=0}^{\dim(\mathbb{M})} \operatorname{im}(\boldsymbol{M}_i)$. But, by assumption we have $\bigcap_{i=0}^{\dim(\mathbb{M})} \operatorname{im}(\boldsymbol{M}_i) = \{0\}$. Thus, the bad case is impossible.

For general m, we have $\operatorname{diag}(\boldsymbol{Q},\ldots)$ and $\operatorname{diag}(\boldsymbol{M},\ldots)$ instead of $\boldsymbol{Q}$ and $\boldsymbol{M}$. We obtain $\left(\begin{smallmatrix}\boldsymbol{w}\\ \boldsymbol{v}\end{smallmatrix}\right)$ from $\mathsf{LMPA}_{\mathrm{almSnd}}$ with $[\boldsymbol{A}|\operatorname{diag}(\boldsymbol{H},\ldots)]\left(\begin{smallmatrix}\boldsymbol{w}\\ \boldsymbol{v}\end{smallmatrix}\right) = [\boldsymbol{t}]$. Evidently,

$$\bigcap_{i=0}^{\dim(\mathbb{M})} \operatorname{im}(\operatorname{diag}(\boldsymbol{M}_i,\ldots,\boldsymbol{M}_i)) = \bigcap_{i=0}^{\dim(\mathbb{M})} \operatorname{im}(\boldsymbol{M}_i)^m = \{\boldsymbol{0}\}.$$

Thus, our claim follows analogously. □

Corollary 3.21. *Protocol* $\mathsf{LMPA}_{\mathrm{ZK}}$ *has* ε*-statistical* HVZK *with* $\varepsilon \in O(\log_k(n)k)/p$.

Proof. This is immediate from Lemma 3.18. □

Remark 3.22. Instead of jumping through hoops, randomising the witness as usual via $\beta\boldsymbol{w} + \boldsymbol{r}$, for $\boldsymbol{r} \leftarrow \mathbb{M}_n$, is tempting. It even is more efficient. However, the surjectivity requirements of Lemma 3.17 now refer to $[\boldsymbol{A}]$ (instead of $[\boldsymbol{H}]$), which is adversarially chosen. Moreover, the masking sets we constructed only work for $m = 1$, so we need new (larger) masking sets. This setting is very technical and entangles soundness and zero-knowledge. In particular, the probability for surjectivity seems related to $\delta_{\mathsf{snd}}(\widetilde{\chi}_{2k-1})$ now. By testing for surjectivity (which is possible at least for $k = 2$), we may still obtain (perfect) zero-knowledge via aborts. All in all, this mixes correctness, soundness and zero-knowledge, which we avoid.

One might try to use $[\boldsymbol{A}] + x[\boldsymbol{H}]$, i.e. use batching on $\left[\begin{smallmatrix}\boldsymbol{A}\\ \boldsymbol{H}\end{smallmatrix}\right]$ to attain a suitable $[\widehat{\boldsymbol{A}}]$ for the above. However, now we may additionally have trouble with soundness similar to Question 3.7.

Remark 3.23 (Proof size and computation.). It is plausible that (perhaps after enlarging the base case of $\mathbb{M}_n$) similar reasoning as used in Lemma 3.18 could be applied to remove the base case of $\mathsf{LMPA}_{\mathrm{ZK}}$ by instead permuting $[\boldsymbol{A}|\boldsymbol{H}]$ so that running $\mathsf{LMPA}_{\mathrm{noZK}}$ (for the permutation) is zero-knowledge. This may be slightly more efficient, but for $m = 2$, which is the standard use case (because $\mathsf{LMPA}_{\mathrm{batch}}$ almost always improves both computation and communication), we do not expect much.

With regards to communication alone, the naive approach sketched in Section 3.4.3 seems to be the best option with size $O(m\log(n))$ instead of $O(m\log(mn))$. Moreover, it yields a proof of knowledge[26] with efficient extraction, but see Question 3.7. The price is $O(mn)$ instead of $O(m\log(n))$ worst-case additional computation over $\mathsf{LMPA}_{\mathrm{noZK}}$.

Due to its more complicated nature, the computational efficiency improvement of $\mathsf{LMPA}_{\mathrm{ZK}}$ over the naive $\mathsf{LMPA}_{\mathrm{simpleZK}}$ is perhaps mostly of theoretical interest.

[26] Formally, the other arguments are also proofs of knowledge, but they are proofs with respect to a changed (OR-type) statement. $\mathsf{LMPA}_{\mathrm{simpleZK}}$ is a proof of knowledge for the original $\exists\boldsymbol{w}\colon [\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$.

3.5 Step 3: Adding (arithmetic circuit) relations to the witness

If the witness $\boldsymbol{w}$ for $[\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$ is committed to, e.g. if the first row of $[\boldsymbol{A}]$ is a Pedersen commitment CRS $[\boldsymbol{g}]$, it is easily possible to make other (zero-knowledge) statements about $\boldsymbol{w}$ by composition of zero-knowledge protocols. Using Protocol $\mathsf{QESA}_{\mathsf{Copy}}$ from Section 4 (or [16]), it is possible to add constraints on the witness. In particular, one can use range-proofs to control $\boldsymbol{w}$.

Remark 3.24. Often, $\boldsymbol{w}$ is much larger than the part which has to satisfy some constraints. It is efficiently possible to "split" and "merge" Pedersen commitments i.e. $[c] = [c_1] + [c_2]$ where $[\boldsymbol{G}] = [\boldsymbol{G}_1|\boldsymbol{G}_2]$ and $[c_i] = [\boldsymbol{G}_i]\boldsymbol{w}_i$. (Indeed, we use this quite often. With small changes, this is possible in zero-knowledge.) With this, one can split off the relevant portion $\boldsymbol{w}_1$ of $\boldsymbol{w}$ into the commitment $[c_1]$ and prove additional relations about this portion only. Splitting is generally very cheap. See Appendix C.1 for a concrete application.

4 Arithmetic circuit satisfiability from quadratic equations

In this section, we describe quadratic gates, and relate them to rank 1 constraint systems (R1CS) and arithmetic circuits (AC). Then, we construct a proof of satisfiability of a set of quadratic equations via a (zero-knowledge) inner-product argument.

4.1 Quadratic gates

The equations our scheme is able to prove are *quadratic equations*, i.e. given a witness $\boldsymbol{w} \in \mathbb{F}_p^n$ and a matrix $\boldsymbol{\Gamma} \in \mathbb{F}_p^{n \times n}$ we wish to prove

$$\boldsymbol{w}^\top \boldsymbol{\Gamma} \boldsymbol{w} = 0.$$

We choose this description of quadratic equations for *simplicity* and *uniformity* of notation. In particular, we assume without loss of generality, that the witness $\boldsymbol{w}$ has the constant 1 as first component, i.e. $w_1 = 1$. Our notation is similar to [22], which uses such notation for Groth–Sahai proofs [32]. Indeed, our arguments are essentially *commit-and-prove* systems [22].

Consider a general quadratic equation $\boldsymbol{x}^\top \boldsymbol{\Gamma} \boldsymbol{x} + \boldsymbol{a}^\top \boldsymbol{x} = t$, with $\boldsymbol{a}, \boldsymbol{x} \in \mathbb{F}_p^n$, $\boldsymbol{\Gamma} \in \mathbb{F}_p^{n \times n}$, $t \in \mathbb{F}_p$ with statement given by the constants $(\boldsymbol{a}, \boldsymbol{\Gamma}, t)$. This can be encoded via $\boldsymbol{w} = \left(\begin{smallmatrix} 1 \\ \boldsymbol{x} \end{smallmatrix}\right)$ and suitably (re)defined $\boldsymbol{\Gamma}$, namely $\boldsymbol{w}^\top \left(\begin{smallmatrix} -t & 0 \\ \boldsymbol{a} & \boldsymbol{\Gamma} \end{smallmatrix}\right) \boldsymbol{w} = 0$.

It is straightforward to encode arithmetic circuits (ACs) as systems of quadratic equations. Doing this allows for ACs built from *quadratic gates*, i.e. gates whose input-output behaviour is described by a quadratic equation.

4.2 Arithmetic circuits and rank 1 constraint systems

Rank 1 constraint systems (R1CS) are systems of equations of the form $(\boldsymbol{w}^\top \boldsymbol{a})(\boldsymbol{b}^\top \boldsymbol{w}) - \boldsymbol{c}^\top \boldsymbol{w} = 0$, where $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \mathbb{F}_p^n$. Evidently, these are special cases of quadratic equations with $\boldsymbol{\Gamma} = \boldsymbol{a}\boldsymbol{b}^\top + e_1 \boldsymbol{c}^\top$.[27] Arithmetic circuit satisfiability can also be encoded in R1CS. See [9] for details.

The gates testable by one R1CS equation allow a single "multiplication". As we saw in the introduction, quadratic equations are more flexible. For example, the inner product $\boldsymbol{x}^\top \boldsymbol{y}$ is a single quadratic gate. To the best of our knowledge, n gates are necessary to encode this in R1CS (essentially one per $x_i y_i$ multiplication). Thus, quadratic gates enable new optimisations. Indeed, all "AC to R1CS" optimisations (and more), are applicable for "AC to QE". Implementing optimisations for "R1CS to QE" is harder, since the "directed graph" structure is implicit.

4.3 The verification strategy

Verification that a system of quadratic gates is satisfied is easy given the witness $\boldsymbol{w}$, in our case the wire assignments of the AC, and equations $\boldsymbol{\Gamma}_{\mathfrak{g}}$ (the gate $\mathfrak{g}$ encoded as a matrix). One simply checks

[27] The name may be R1CS misleading, since $\boldsymbol{\Gamma}$ may have (tensor) rank 2, i.e. the (tensor) rank of $\boldsymbol{\Gamma}$ is ≤ 2 for R1CS (and arbitrary for general quadratic equations). Nevertheless, we follow this standard naming convention.

$\boldsymbol{w}^\top \boldsymbol{\Gamma}_{\mathfrak{g}} \boldsymbol{w} = 0$ for all $\mathfrak{g} \in \mathfrak{G}$. By batching this can be sped up: Pick $(r_{\mathfrak{g}})_{\mathfrak{g}} \leftarrow \chi_{\#\mathfrak{G}}$ from a testing distribution. Then compute $\boldsymbol{\Gamma} := \sum_{\mathfrak{g}\in\mathfrak{G}} r_{\mathfrak{g}} \boldsymbol{\Gamma}_{\mathfrak{g}}$ as the "batched statement". Finally, check if $\boldsymbol{w}^\top \boldsymbol{\Gamma} \boldsymbol{w} = 0$.

We run this strategy in a commit-then-prove manner. First, we commit to the witness $\boldsymbol{w}$. Then we let the verifier pick testing randomness $(r_{\mathfrak{g}})_{\mathfrak{g}}$ and we prove that $\boldsymbol{w}^\top \boldsymbol{\Gamma} \boldsymbol{w} = 0$ where $\boldsymbol{\Gamma} := \sum_{\mathfrak{g}\in\mathfrak{G}} r_{\mathfrak{g}} \boldsymbol{\Gamma}_{\mathfrak{g}}$ is the "batched statement". Note that $\boldsymbol{w}^\top \boldsymbol{\Gamma} \boldsymbol{w} = \langle \boldsymbol{w}, \boldsymbol{\Gamma}\boldsymbol{w} \rangle$ is an inner product. Hence, we require a *zero-knowledge* inner-product argument.

For technical reasons, we cannot generate a commitment to $\boldsymbol{\Gamma}\boldsymbol{w}$ efficiently (prior to knowing $\boldsymbol{\Gamma}$).[28] Therefore, the prover first commits to $\boldsymbol{w}$ as $[c_{\boldsymbol{x}}] = \mathsf{Com}_{\mathrm{ck}_1}(\boldsymbol{w})$. Then he obtains $\boldsymbol{\Gamma}$ and commits to $\boldsymbol{\Gamma}\boldsymbol{w}$ as $[c_{\boldsymbol{y}}] = \mathsf{Com}_{\mathrm{ck}_2}(\boldsymbol{\Gamma}\boldsymbol{w})$. Then the prover carries out the inner product argument. He must also *prove* that the commitments $[c_{\boldsymbol{x}}]$ and $[c_{\boldsymbol{y}}]$ open to values $\boldsymbol{x} = \boldsymbol{w}$ and $\boldsymbol{y} = \boldsymbol{\Gamma}\boldsymbol{w}$ as promised. Again, we use (linear) batching to shorten the proof for $\boldsymbol{y} = \boldsymbol{\Gamma}\boldsymbol{x}$. Namely, to check $\boldsymbol{y} = \boldsymbol{\Gamma}\boldsymbol{x}$, the verifier picks random $\boldsymbol{s} \leftarrow \chi_n$ (after $[c_{\boldsymbol{x}}], [c_{\boldsymbol{y}}]$ and hence $\boldsymbol{x}$, $\boldsymbol{y}$ are fixed) and the prover proves $0 = \langle \boldsymbol{\Gamma}\boldsymbol{x} - \boldsymbol{y}, \boldsymbol{s} \rangle$.

Instead of running two inner product arguments (for $\langle \boldsymbol{\Gamma}\boldsymbol{x} - \boldsymbol{y}, \boldsymbol{s} \rangle = 0$ and $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = 0$) we immediately batch verify again: The verifier picks randomness α and the prover proves knowledge of openings $\boldsymbol{x}, \boldsymbol{y}$ such that,

$$\begin{aligned} \langle \boldsymbol{x} - \alpha \boldsymbol{s}, \boldsymbol{y} + \alpha \boldsymbol{\Gamma}^\top \boldsymbol{s} \rangle &= \langle \boldsymbol{x}, \boldsymbol{y} \rangle + \alpha \Big(\langle \boldsymbol{x}, \boldsymbol{\Gamma}^\top \boldsymbol{s} \rangle - \langle \boldsymbol{s}, \boldsymbol{y} \rangle \Big) - \alpha^2 \langle \boldsymbol{s}, \boldsymbol{\Gamma}^\top \boldsymbol{s} \rangle \\ &= \langle \boldsymbol{x}, \boldsymbol{y} \rangle + \alpha \langle \boldsymbol{\Gamma}\boldsymbol{x} - \boldsymbol{y}, \boldsymbol{s} \rangle - \alpha^2 \langle \boldsymbol{s}, \boldsymbol{\Gamma}^\top \boldsymbol{s} \rangle \\ &\overset{!}{=} - \underbrace{\alpha^2 \langle \boldsymbol{s}, \boldsymbol{\Gamma}^\top \boldsymbol{s} \rangle}_{=:t} \end{aligned} \tag{4.1}$$

where t is fixed by the random choices of the verifier. If $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\Gamma}, \boldsymbol{s}$ are fixed, the lemma of Schwartz–Zippel can be applied to the polynomial in α. If $\alpha \leftarrow \mathcal{S}$, the probability that Eq. (4.1) holds but $\langle \boldsymbol{x}, \boldsymbol{y} \rangle \neq 0$ or $\langle \boldsymbol{\Gamma}\boldsymbol{x} - \boldsymbol{y}, \boldsymbol{s} \rangle \neq 0$ is $2/\#\mathcal{S}$. If $\boldsymbol{s}$ is chosen from a testing distribution χ_n with error $\delta_{\mathsf{snd}}(\chi_n)$, the probability that $\boldsymbol{\Gamma}\boldsymbol{x} - \boldsymbol{y} \neq 0$ is at most $\delta_{\mathsf{snd}}(\chi_n)$. Thus, this strategy is sound. To instantiate it, we need a zero-knowledge inner product argument.

4.4 Zero-knowledge inner product argument

Now, we show how to construct a zero-knowledge inner product argument (IPA). We first recall [13, 16], from a high level. We identify [16] as a linear combination of protocols. Unlike for $\mathsf{LMPA}_{\mathsf{ZK}}$, we will not use linear combination (with "extended randomness") to attain zero-knowledge. We achieve HVZK similar to Protocol 3.13 by masking the witness, but we also exploit redundancy (or kernel) guideline. Addition of zero-knowledge adds a single round, where one group element and one challenge are sent. (Note that $m = 1$ now.) For technical reasons we have a base case at $n = 8$ (for $k = 2$). In practice, this is hardly worth mentioning.

4.4.1 Inner product argument (IPA).

First, we describe the inner product argument following [13, 16]. It will also be evident how to extend [16] to $k > 2$. Since $k = 2$ minimises communication, we only mention this in passing. For simplicity, we ignore zero-knowledge.

Our setting is as follows: We have a CRS crs $= ([\boldsymbol{g}'], [\boldsymbol{g}''], [Q])$ for which finding a non-trivial kernel element of $[\boldsymbol{g}', \boldsymbol{g}'', Q] \in \mathbb{G}^{2n+1}$ is hard. In other words, these are three independent (or one large three-split) Pedersen commitment keys.

Naively, one proves knowledge of openings of $c'_{\boldsymbol{w}}$ and $c''_{\boldsymbol{w}}$ with $\langle \boldsymbol{w}', \boldsymbol{w}'' \rangle = t$. The idea and argument(s) in Section 3.4, in particular Protocol 3.13, allow to recursively shrink our statement. After one recursion step, we obtain $\langle \widehat{\boldsymbol{w}}', \widehat{\boldsymbol{w}}'' \rangle = \widehat{t}$. The prover sends terms $v_{\pm 1} = \langle \boldsymbol{w}'_i, \boldsymbol{w}''_j \rangle$ (for $j - i = \pm 1$), so that the verifier can compute $\widehat{t}$, in analogy to $[u_{\pm 1}]$ in Section 3.4,

A linear combination of Protocol $\mathsf{LMPA}_{\mathrm{noZK}}$ for $\boldsymbol{w}'$ and $\boldsymbol{w}''$ with the same challenge $(\boldsymbol{x}, \boldsymbol{y})$ does not work so well with $\widehat{t}$ as indicated by following formula: $\widehat{t} = \langle \widehat{\boldsymbol{w}}', \widehat{\boldsymbol{w}}'' \rangle = \langle \boldsymbol{x}^\top \vec{\boldsymbol{w}}', \boldsymbol{x}^\top \vec{\boldsymbol{w}}'' \rangle$. There are

[28] If $[\boldsymbol{g}]$ is a Pedersen commitment key and $[c] = [\boldsymbol{g}]\boldsymbol{w}$, then $[\boldsymbol{h}] = [\boldsymbol{g}]\boldsymbol{\Gamma}^{-1}$ is a Pedersen commitment key where $[c] = [\boldsymbol{h}](\boldsymbol{\Gamma}\boldsymbol{w})$. We do not use this for various reasons.

no compatibility guarantees for this expression, and indeed for $\boldsymbol{x} = (1, \xi)$, $\boldsymbol{y} = (\xi, 1)$, we find

$$\langle \widehat{\boldsymbol{w}}', \widehat{\boldsymbol{w}}'' \rangle = \langle \boldsymbol{w}_1', \boldsymbol{w}_1'' \rangle + \xi(\langle \boldsymbol{w}_1', \boldsymbol{w}_2'' \rangle + \langle \boldsymbol{w}_2', \boldsymbol{w}_1'' \rangle) + \xi^2 \langle \boldsymbol{w}_2', \boldsymbol{w}_2'' \rangle.$$

In analogy to $[u_0]$ in $\mathsf{LMPA}_{\text{noZK}}$, we want that the term $t = \langle \boldsymbol{w}_1', \boldsymbol{w}_1'' \rangle + \langle \boldsymbol{w}_1', \boldsymbol{w}_1'' \rangle$ appears (perhaps scaled by ξ) and is preserved. Instead the "mixed terms" are preserved this way! Fortunately, we solved this problem in Section 3.4 already. The solution is to use $\langle \boldsymbol{x}^\top \boldsymbol{w}', \boldsymbol{y}^\top \boldsymbol{w}'' \rangle$, since $\boldsymbol{x}$ and $\boldsymbol{y}$ are constructed like that.[29] Thus, we find

$$\langle \boldsymbol{x}^\top \boldsymbol{w}', \boldsymbol{y}^\top \boldsymbol{w}'' \rangle = \xi \langle \boldsymbol{w}', \boldsymbol{w}'' \rangle + \langle \boldsymbol{w}_1', \boldsymbol{w}_2'' \rangle + \xi^2 \langle \boldsymbol{w}_2', \boldsymbol{w}_1'' \rangle$$

Therefore we run the protocol for $\boldsymbol{w}'$ with challenge $(\boldsymbol{x}, \boldsymbol{y})$, and we run the protocol for $\boldsymbol{w}''$ with flipped challenge $(\boldsymbol{y}, \boldsymbol{x})$. Now, as in Protocol $\mathsf{LMPA}_{\text{noZK}}$, it suffices to send $v_{j-i} \coloneqq \langle \boldsymbol{w}_i', \boldsymbol{w}_{2-i}'' \rangle$ (for $i = 1, 2$).

The argument described above is a hybrid of [13] and [16]. For security, we need that "commitment merging" (see Remark 3.24), which the linear combination of protocols induces, still is *binding*. To obtain [16], we simply *commit* to v_ℓ as well (using $[Q]$), and send the combined commitment, i.e. apply again a linear combination. This "merged" commitment key is now $[\boldsymbol{g}', \boldsymbol{g}'', Q]$. Thus instead of sending two messages thrice (namely $[u_{\pm 1}']$, $[u_{\mp 1}'']$, $[v_{\mp 1} Q]$), we only send the two "merged commitments" $[u_{\pm 1}] = [u_{\pm 1}'] + [u_{\mp 1}''] + [v_{\mp 1} Q]$. Unlike [16], which uses $\boldsymbol{x} = (\xi^{-1}, \xi)$ we prefer $\boldsymbol{x} = (1, \xi)$ since exponentiation with 1 is free.

Protocol 4.1 ($\mathsf{IPA}_{\text{noZK}}$). The following is an inner product argument to prove

$$\exists \boldsymbol{w}', \boldsymbol{w}'' \in \mathbb{F}_p^n \colon [c] = [\boldsymbol{g}']\boldsymbol{w}' + [\boldsymbol{g}'']\boldsymbol{w}'' + t[Q] \;\wedge\; \langle \boldsymbol{w}', \boldsymbol{w}'' \rangle = t.$$

Let $\widetilde{\chi}_{2k-1}$ (and $\chi^{(\beta \neq 0)}$) be a testing distribution with the properties as in Protocol 3.9, i.e. we have $\boldsymbol{z} \leftarrow \widetilde{\chi}_{2k-1}$ (with $\boldsymbol{z}$ indexed from $-k$ to k) together with $\boldsymbol{x}, \boldsymbol{y}$ such that $z_{j-i} = x_i y_j$. Common input is $\text{crs} = ([\boldsymbol{g}', \boldsymbol{g}'', Q])) \in \mathbb{G}^{1 \times n} \times \mathbb{G}^{1 \times n} \times \mathbb{G}$ and the statement $([c], t)$ We assume $n = k^d$. The prover's witness is $(\boldsymbol{w}', \boldsymbol{w}'')$.

- $\mathcal{V} \to \mathcal{P}$: (Step 0: "Fixing" t.) $\mathcal{V}$ picks $\alpha \leftarrow \chi^{(\beta \neq 0)}$. Send α. Both sides set $[Q] \coloneqq \alpha^{-1}[Q]$. Then they set $[c] \coloneqq ([c] - \alpha t[Q]) + t[Q]$.[30]
  Recursive step. Suppose $n = k^d > 1$.
- $\mathcal{P} \to \mathcal{V}$: Compute $[\boldsymbol{u}_\ell'] = \sum_{i-j=\ell} [\boldsymbol{g}_i'] \boldsymbol{w}_j'$, where $[\boldsymbol{g}_j']$ and $[\boldsymbol{w}_j']$ are as usual (i.e. split $[\boldsymbol{g}']$, $[\boldsymbol{w}']$ into k equal-size pieces). Compute the respective $[\boldsymbol{u}_\ell'']$. Let $v_\ell \coloneqq \sum_{\ell=j-i} \langle \boldsymbol{w}_i', \boldsymbol{w}_j'' \rangle$. Let $[\boldsymbol{u}_\ell] \coloneqq [\boldsymbol{u}_\ell'] + [\boldsymbol{u}_{-\ell}''] + v_{-\ell}[Q]$. Send $[\boldsymbol{u}_\ell]$ for $\ell = \pm 1, \dots, \pm(k-1)$.[31]
- $\mathcal{V} \to \mathcal{P}$: pick $\boldsymbol{z} \leftarrow \widetilde{\chi}_{2k-1}$ with corresponding $\boldsymbol{x}, \boldsymbol{y}$. Send $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$.
- Both parties compute $[\widehat{\boldsymbol{g}}'] = \boldsymbol{x}^\top [\vec{\boldsymbol{g}}'] = \sum x_i [\boldsymbol{g}_i'] \in \mathbb{G}^{1 \times n/k}$ and $[\widehat{\boldsymbol{g}}''] = \boldsymbol{y}^\top [\vec{\boldsymbol{g}}''] \in \mathbb{G}^{1 \times n/k}$. and $[\widehat{c}] = \boldsymbol{z}^\top [\vec{\boldsymbol{u}}] = \sum_\ell \boldsymbol{z}_\ell [\boldsymbol{u}_\ell] \in \mathbb{G}$ as new batched statement. Moreover, $\mathcal{P}$ computes $\widehat{\boldsymbol{w}}' = \boldsymbol{y}^\top \vec{\boldsymbol{w}}'$ and $\widehat{\boldsymbol{w}}'' = \boldsymbol{x}^\top \vec{\boldsymbol{w}}''$. (Note the invariant: $\widehat{t} = \boldsymbol{z}^\top \boldsymbol{v} = \sum_{\ell=-k+1}^{k-1} z_\ell v_\ell = \langle \widehat{\boldsymbol{w}}', \widehat{\boldsymbol{w}}'' \rangle$.)
  Skipping Step 0, recursively continue with $n \leftarrow n/k$, $\boldsymbol{w}' \leftarrow \widehat{\boldsymbol{w}}'$, $\boldsymbol{w}'' \leftarrow \widehat{\boldsymbol{w}}''$, $[c] \leftarrow [\widehat{c}]$, $[\boldsymbol{g}'] \leftarrow [\widehat{\boldsymbol{g}}']$, $[\boldsymbol{g}''] \leftarrow [\widehat{\boldsymbol{g}}'']$.
  Base case. Suppose $n = 1$.
- $\mathcal{P} \to \mathcal{V}$: Send $\boldsymbol{w}'$, $\boldsymbol{w}''$.
- $\mathcal{V}$: Let $t \coloneqq \langle \boldsymbol{w}', \boldsymbol{w}'' \rangle$ and test if $[c] \overset{?}{=} [\boldsymbol{g}']\boldsymbol{w}' + [\boldsymbol{g}'']\boldsymbol{w}'' + t[Q]$.

See Appendix G for a sketch of this protocol.

The above argument is correct by inspection. Due to space constraints, we do not consider Step 0 a transformation on its own, compare [16, Protocol 1]. In Step 0, we "force" a commitment to t. Hence t is not explicit in any further compuation, but $[c]$ satisfies the invariant that it is a commitment to $\boldsymbol{w}', \boldsymbol{w}'', t$ where $t = \langle \boldsymbol{w}', \boldsymbol{w}'' \rangle$.

Lemma 4.2. *Protocol* $\mathsf{IPA}_{\text{noZK}}$ *is special μ-sound (with $\mu = (2, 2k, \dots, 2k)$) for finding a valid witness or a non-trivial element in the kernel of $[\boldsymbol{g}', \boldsymbol{g}'', Q]$. It has short-circuit extraction with $\mu' = (1, k, \dots, k)$.*

[29] Remember that in Protocols 3.9, 3.14 and 3.19 $\boldsymbol{x}$, $\boldsymbol{y}$ associated to $\boldsymbol{z}$ satisfy $x_i y_j = z_{j-i}$.

[30] This changes the committed value s in $[c]$ to $\alpha(s - t) + t$ under the new $[Q]$.

[31] Note that $[\boldsymbol{u}_0]$ is implicitly known to $\mathcal{V}$.

The proof is straightforward, and essentially the same as [13, 16]

Proof. First, we ignore Step 0, i.e. we treat t as part of the prover's witness and do not change $[Q]$ or $[c]$. (This is the case in the recursive steps.)

Then, the proof of extraction is essentially the same as Lemma 3.10. More concretely: The base case is extractable by definition. In each recursive step, we find $2k$ transcripts $\widehat{\boldsymbol{w}}'_i$, $\widehat{\boldsymbol{w}}''_i$, $\widehat{t}_i$, $\boldsymbol{z}_i$ (with fixed $[\boldsymbol{u}_\ell]$) which we arrange into a matrix equation as in Lemma 3.10. Using invertibility of the obtained matrix $\boldsymbol{Z}$, we can compute a *candidate* witness $\boldsymbol{w}'$, $\boldsymbol{w}''$, s which is a valid opening of $[c]$ and all $[\boldsymbol{u}_\ell]$. Again, by computing $\boldsymbol{w}'$ and $\boldsymbol{w}''$ from k transcripts by using $\boldsymbol{Y}^{-1}$ (resp. $\boldsymbol{X}^{-1}$) we get quick-extraction, if $t = \langle \boldsymbol{w}', \boldsymbol{w}'' \rangle$.

If quick-extraction fails, then either we get a non-trivial element in $\ker([\boldsymbol{g}', \boldsymbol{g}'', \alpha Q_{\mathrm{old}}])$, as in Lemma 3.10, and (since $\alpha \neq 0$) find one in $\ker([\boldsymbol{g}', \boldsymbol{g}'', Q_{\mathrm{old}}])$.[32] Except, if we have the case $t \neq \langle \boldsymbol{w}', \boldsymbol{w}'' \rangle$. Suppose the latter happens. Since we have openings of the commitment, and by induction hypothesis, we find

$$\sum_{i=-k+1}^{k-1} \langle \widehat{\boldsymbol{w}}', \widehat{\boldsymbol{w}}'', z \rangle_\ell = \langle \boldsymbol{x}^\top \boldsymbol{w}', \boldsymbol{y}^\top \boldsymbol{w}', = \rangle \sum_\ell v_\ell z_\ell$$

for all $2k$ challenges. This implies that $t = v_0 = \langle \boldsymbol{w}', \boldsymbol{w}'' \rangle$, so the case $t \neq \langle \boldsymbol{w}', \boldsymbol{w}'' \rangle$ cannot happen (without breaking the commitment first).

Finally, let us consider Step 0. Given 2 transcripts $\alpha_1 \neq \alpha_2$ with extracted witnesses (from subtrees) $\boldsymbol{w}'$, $\boldsymbol{w}''$, and $s := \langle \boldsymbol{w}', \boldsymbol{w}'' \rangle$. These witnesses are identical for both subtrees, or we find a non-trival kernel element. But then the recomputation of $[c]$ and $[Q]$ in Step 0 implies that $\alpha_1(s-t) = \alpha_2(s-t)$, which implies $s - t = 0$ as claimed. □

4.4.2 Going zero-knowledge.

Making the inner-product argument zero-knowledge can be done in many ways. To be competetive with Bulletproofs [16], which uses the IPA without zero-knowledge, we directly mask the witness (as in Protocol 3.13, unlike $\mathsf{LMPA}_{\mathrm{ZK}}$). This is problematic, since the scalar product is non-linear. Consequently, our (initial) approach only works under some (mild) constraints.

As mentioned above, the problem with using masking randomness and proving $\langle \boldsymbol{w}' + \boldsymbol{r}', \boldsymbol{w}'' + \boldsymbol{r}'' \rangle$ is the non-linearity: Sending only $t_{\boldsymbol{r}} = \langle \boldsymbol{r}', \boldsymbol{r}'' \rangle$ to the verifier is not enough. So we need to send also $\langle \boldsymbol{w}', \boldsymbol{r}'' \rangle$ or $\langle \boldsymbol{r}', \boldsymbol{w}'' \rangle$ or some other "error term" to correct the non-linearity. Then we have to show that these terms don't expose "information" about the witness. In particular, sending $\beta \boldsymbol{w}' + \boldsymbol{r}'$, which was possible in Section 3.3, seems impossible.

Fortunately, we already saw that the recursive argument only needs a small amount of randomness to conceal the witness. We exploit this now to show that the sketched masking *almost* yields zero-knowledge. Instead of sending the error terms, we pick randomness with the "kernel guideline" in mind:

- $\boldsymbol{r}' \in \ker(\boldsymbol{w}''^\top)$, i.e. $\langle \boldsymbol{r}', \boldsymbol{w}'' \rangle = 0$.
- $\boldsymbol{r}'' \in \ker(\boldsymbol{w}'^\top) \cap \ker(\boldsymbol{r}'^\top)$, i.e. $\langle \boldsymbol{w}', \boldsymbol{r}'' \rangle = 0 = \langle \boldsymbol{r}', \boldsymbol{r}'' \rangle$.

In other words, we pick randomness which does not induce any errors. Thus, we do not need to send anything besides $[t_{\boldsymbol{r}}] = [\boldsymbol{g}']\boldsymbol{r}' + [\boldsymbol{g}'']\boldsymbol{r}''$ to the verifier. Let us first outline our almost zero-knowledge argument, using an augmented masking set $\mathbb{M}_n^+$ which is defined later.

Protocol 4.3 ($\mathsf{IPA}_{\mathrm{almZK}}$). The following is an inner product argument with the same statement, witness and notation as Protocol 4.1 ($\mathsf{IPA}_{\mathrm{noZK}}$).

- $\mathcal{P} \to \mathcal{V}$: Pick $\boldsymbol{r}' \leftarrow \ker(\boldsymbol{w}''^\top) \cap \mathbb{M}_n^+$ and $\boldsymbol{r}'' \leftarrow \ker(\left(\begin{smallmatrix} \boldsymbol{w}'^\top \\ \boldsymbol{r}'^\top \end{smallmatrix}\right)) \cap \mathbb{M}_n^+$. Compute $[c_{\boldsymbol{r}}] := [\boldsymbol{g}']\boldsymbol{r}' + [\boldsymbol{g}'']\boldsymbol{r}''$. Send $[c_{\boldsymbol{r}}]$.
- $\mathcal{V} \to \mathcal{P}$: Pick $\beta \leftarrow \chi^{(\beta)}$. Send β.

[32] Here $[Q_{\mathrm{old}}]$ is $[Q]$ before overwriting in Step 0.

- $\mathcal{P} \leftrightarrow \mathcal{V}$: Engage in Protocol $\mathsf{IPA}_{\mathrm{noZK}}$ for $\langle \beta \boldsymbol{w}' + \boldsymbol{r}', \beta \boldsymbol{w}'' + \boldsymbol{r}'' \rangle = \beta^2 t$ (with commitment $[c] = \beta[c_{\boldsymbol{w}}] + [c_{\boldsymbol{r}}] + \beta^2 t[Q]$). Verifier (and prover) use t (and $[c_{\boldsymbol{w}}]$) from the statement to compute $[c]$.[33]

See Appendix G for a sketch of this protocol.

Correctness follows by inspection. Special soundness follows essentially from Lemma 4.2 and Lemma 3.2.

Corollary 4.4. *Protocol 4.3 is special μ-sound (with $\mu = (2, 2, 2k, \dots, 2k)$) for finding a valid witness or a non-trivial element in the kernel of $[\boldsymbol{g}', \boldsymbol{g}'', Q]$. It has short-circuit extraction with $\mu = (2, 1, k, \dots, k)$.*

Showing zero-knowledge is more contrived. As for $\mathsf{LMPA}_{\mathrm{ZK}}$ in Lemma 3.17, we want to show that the prover's messages are uniformly random. Unfortunately, the constraints which must be satisfied now depend on the witness. Thus, an adversarially chosen witness may be a problem. Fortunately, we use $\mathsf{IPA}_{\mathrm{almZK}}$ with "randomised" witnesses, so this problem does not manifest.

Definition 4.5. Let k be fixed and $n \geq 4k$. Define $\mathbb{M}_n^+ := \mathbb{M}_n \,\dot\cup\, \{n-2, n-1\}$. (Recall that $\mathbb{M}_n$ indices are zero-based and $n-2, n-1 \notin \mathbb{M}_n$ for $n \geq 4k$.)

We introduce $\mathbb{M}_n^+$ because satisfying the kernel constraints "consumes" one (resp. two) pieces of randomness in $\boldsymbol{r}'$ (resp. $\boldsymbol{r}''$). We compensate this in $\mathbb{M}_n^+$.

For the sake of simplicity, we stick to $k = 2$. It should be evident how to appropriately generalise, c.f. Lemma 3.10.

Lemma 4.6. *Let* $\mathrm{crs} = ([\boldsymbol{g}', \boldsymbol{g}'', Q]))$ *be as in Protocol 4.3 ($\mathsf{IPA}_{\mathrm{almZK}}$) and $k = 2$. Define*

$$\widetilde{\boldsymbol{M}}' := \begin{pmatrix} \boldsymbol{w}_1''^\top & \boldsymbol{w}_2''^\top \\ \boldsymbol{g}_2' & 0 \\ 0 & \boldsymbol{g}_1' \\ y_1 \,\mathrm{id} & y_2 \,\mathrm{id}_2 \end{pmatrix} \quad \text{and} \quad \widetilde{\boldsymbol{M}}'' := \begin{pmatrix} \boldsymbol{w}_1'^\top & \boldsymbol{w}_2'^\top \\ \boldsymbol{r}_1'^\top & \boldsymbol{r}_2'^\top \\ \boldsymbol{g}_2'' & 0 \\ 0 & \boldsymbol{g}_1'' \\ x_1 \,\mathrm{id} & x_2 \,\mathrm{id}_2 \end{pmatrix}.$$

Suppose that $n \geq 4k$ and let $\mathbb{M}_n^+$ be as in Definition 4.5. Suppose $\boldsymbol{r}' \leftarrow \ker(\boldsymbol{w}''^\top) \cap \mathbb{M}_n^+$ and $\boldsymbol{r}'' \leftarrow \ker(\boldsymbol{w}'^\top) \cap \mathbb{M}_n^+$. If $\widetilde{\boldsymbol{M}}'$ and $\widetilde{\boldsymbol{M}}''$ restricted to columns in $\mathbb{M}_n^+$ are surjective, then all messages $[\boldsymbol{u}_\ell]$ of the IPA are uniformly randomly distributed with probability $O(\log_k(n))/p$. Moreover, the final (plain) messages $\widehat{\boldsymbol{w}}'$, $\widehat{\boldsymbol{w}}''$ are uniformly random.

Let $\mathcal{A}$ be an HVZK *adversary which picks witnesses $(\boldsymbol{w}', \boldsymbol{w}'')$ which satisfy the conditions (except with probability ε). Then protocol is δ-statistical* HVZK *against $\mathcal{A}$, with $\delta \in O(\log_k(n)k)/p + \varepsilon$.*

We introduce $\mathbb{M}_n^+$ because we have to satisfy the kernel constraints of $\boldsymbol{r}'$ and $\boldsymbol{r}''$. Concretely, since $\widetilde{\boldsymbol{M}}'' \in \mathbb{F}_p^{\dim(\mathbb{M}_n^+) \times (\dim(\mathbb{M}_{n/k})+4)}$ must be surjective, we require $\dim(\mathbb{M}_n^+) \geq \dim(\mathbb{M}_n) + 2 \geq \dim(\mathbb{M}_{n/k}) + 4$. This increases our base case (e.g. $n \geq 8$ for $k = 2$). Otherwise, the idea and proof of Lemma 4.6 is very similar to Lemma 3.17. Generalisations of this result to $k \geq 2$ are straightforward. Also, note that to ensure all $[\boldsymbol{u}_\ell]$ are randomised, using either $\boldsymbol{r}'$ or $\boldsymbol{r}''$ is enough, because either randomises $[\boldsymbol{u}_\ell']$ resp. $[\boldsymbol{u}_\ell'']$. Since $\boldsymbol{r}'' = \boldsymbol{0}$ is *not* sufficient for randomisation, we treat $\boldsymbol{r}'$ and $\boldsymbol{r}''$ symmetrically for simplicity.

Lemma 4.6. For the claim of random $[\boldsymbol{u}_\ell]$, we are essentially only concerned with the subprotocol and its specific input. We now analyse the rounds of this subprotocol.

The first round. Recall Lemma 3.16 which states that, if $\widetilde{\boldsymbol{M}} \in \mathbb{F}_p^n \to \mathbb{F}_p^m$ is surjective, the image a uniformly drawn element is uniformly distributed. By definition, the matrix $\widetilde{\boldsymbol{M}}'$ (resp. $\widetilde{\boldsymbol{M}}''$) is constructed essentially as in Lemma 3.17:

- The first (resp. first two) rows are the *constraints* imposed on $\boldsymbol{r}'$ (resp. $\boldsymbol{r}''$).
- The rows with $\boldsymbol{g}_1'$, $\boldsymbol{g}_2'$ (resp. $\boldsymbol{g}_1''$, $\boldsymbol{g}_2''$) yield the respective $\boldsymbol{u}_{\pm 1}'$ (resp. $\boldsymbol{u}_{\pm 1}''$.

[33] As usual, the choice β and $\langle \beta \boldsymbol{w}' + \boldsymbol{r}', \beta \boldsymbol{w}'' + \boldsymbol{r}'' \rangle = \beta^2 t$ is purely for simplicity and suitable general testing distributions work. In particular, if $\beta \leftarrow \chi^{(\beta \neq 0)}$, then $\langle \boldsymbol{w}' + \beta \boldsymbol{r}', \boldsymbol{w}'' + \beta \boldsymbol{r}'' \rangle = t$ works as well. (We want $\beta \neq 0$ for zero-knowledge.)

- The blocks of multiples of identity matrices yield the batched randomness/witness $\widehat{\boldsymbol{r}}'$ (resp. $\widehat{\boldsymbol{r}}''$).

In formulas:

$$\widetilde{\boldsymbol{M}}'\begin{pmatrix}\boldsymbol{r}_1'\\ \boldsymbol{r}_2'\end{pmatrix} = \begin{pmatrix}0\\ u_{-1}'\\ u_{+1}'\\ \widehat{\boldsymbol{r}}'\end{pmatrix} \quad\text{and}\quad \widetilde{\boldsymbol{M}}''\begin{pmatrix}\boldsymbol{r}_1''\\ \boldsymbol{r}_2''\end{pmatrix} = \begin{pmatrix}0\\ 0\\ u_{-1}''\\ u_{+1}''\\ \widehat{\boldsymbol{r}}''\end{pmatrix}.$$

Thus, if $\widetilde{\boldsymbol{M}}'$ (resp. $\widetilde{\boldsymbol{M}}''$) is surjective, we get that the image of $\boldsymbol{r}'$ (resp. $\boldsymbol{r}''$) is uniformly distributed. This holds, even if we choose $\boldsymbol{r}', \boldsymbol{r}''$ uniformly random subject to the constraint imposed by the first (and second) rows.[34] Consequently, $\boldsymbol{w}' + \boldsymbol{r}'$ is reduced to a some $\widehat{\boldsymbol{w}}' + \widehat{\boldsymbol{r}}'$ where $\boldsymbol{r}'$ is uniformly distributed in $\mathbb{M}_{n/2}$. (We argue identically for $\boldsymbol{w}'' + \boldsymbol{r}''$). Note that we use linearity of the protocol's recursive step (i.e. $\vec{\boldsymbol{v}} \mapsto \sum_i x_i \boldsymbol{v}_i$) to treat $\widehat{\boldsymbol{w}}'$ and $\widehat{\boldsymbol{r}}'$ as separately batched entities.

Recursive rounds. From this point on, we are working with the same masking set $\mathbb{M}_{n/2}$ as in Lemma 3.17. Let $\boldsymbol{M}'$ (resp. $\boldsymbol{M}''$) be $\widetilde{\boldsymbol{M}}'$ (resp. $\widetilde{\boldsymbol{M}}''$) without the first (and second) row, i.e. without the constraints. These are the "transition matrices" for a recursion, c.f. Lemma 3.17. By the same arguments as in Lemma 3.17, we find that each $[\boldsymbol{u}_\ell]$ (in fact, each $[\boldsymbol{u}_\ell']$, $[\boldsymbol{u}_\ell'']$) in each round is uniformly distributed with high probability. The biggest difference is that we separate $\boldsymbol{w}$ and $\boldsymbol{r}$ only "conceptually" (as we did above). In a sense, we work with $[\boldsymbol{g}', \boldsymbol{g}']\left(\begin{smallmatrix}\boldsymbol{w}'\\ \boldsymbol{r}'\end{smallmatrix}\right)$.

Since we run protocol $\mathsf{LMPA}_{\text{noZK}}$ essentially twice, we obtain at most twice the failure probability.[35] That is, the failure probability stays at $O(\log_k(n))/p$ as claimed. (Remember that the failure probability is the probability that $[\boldsymbol{u}_\ell]$ is not uniformly distributed. This happens if (both) $\boldsymbol{M}'$ and $\boldsymbol{M}''$ are not surjective.)

The last round. Finally, consider the last recursive round, i.e. the reduction from $n = k$ to $n = 1$. Note that by definition of $\mathbb{M}_n$, we have $\mathbb{M}_k = \{0, \dots, k-1\}$, i.e. $\mathbb{M}_k$ masks everything. Remember that we need $2(k-1)$ terms of randomness for randomising the $\boldsymbol{u}_{\pm\ell}$, i.e. the matrices $\boldsymbol{M}', \boldsymbol{M}'' \in \mathbb{F}_p^{k\times 2k-1}$ *cannot* be surjective! However, we can conceptually separate the randomisation of $\boldsymbol{u}_{\pm\ell}$ into $\boldsymbol{u}_\ell$ and $\boldsymbol{u}_{-\ell}$ (for $\ell = 1, \dots, k-1$) as follows: For $\boldsymbol{u}_\ell$ we pick $\boldsymbol{r}'$, i.e. $\boldsymbol{u}_\ell'$, to randomise. For $\boldsymbol{u}_{-\ell}$ we pick $\boldsymbol{r}''$, i.e. $\boldsymbol{u}_\ell''$, to randomise.[36] Consequently, we can work with different $\boldsymbol{M}'$, $\boldsymbol{M}''$ where one row is removed. Now the matrices are square matrices and evidently bijective (since $x_i, y_i \neq 0$). Thus $\widehat{\boldsymbol{w}}' = \sum x_i(w_i' + r_i')$ and $\widehat{\boldsymbol{w}}''$ are uniformly random. This finishes our argument. (More formally, write down the combined transition matrix, instead of keeping $\boldsymbol{M}'$ and $\boldsymbol{M}''$ separate. It is evident that this $2k \times (2k-1)$ matrix is surjective.)

HVZK Simulation. Simulation for adversaries as described is straightforward. We sketch an explicit simulator for completeness. With probability $\eta = \varepsilon + O(\log_k(n)k)/p$, all transition matrices $\boldsymbol{M}'$, $\boldsymbol{M}''$ as well as $\widetilde{\boldsymbol{M}}'$, $\widetilde{\boldsymbol{M}}''$ are surjective. Hence with probability η, the response is uniformly random. We simulate a transcripts in reverse. For the base case, we pick $w_1', w_2'' \leftarrow \mathbb{F}_p$ and computing the respective $[c] \mathrel{\widehat{=}} [\boldsymbol{u}_0]$.

Now, we simulate recursive steps by picking $[\boldsymbol{u}_\ell]$, $\ell = \pm 1, \dots, \pm k - 1$ uniformly at random and computing $[\boldsymbol{u}_0]$ from this. The final round yields $[\boldsymbol{u}_0] = [c]$, which was supposedly used in the (non-zeroknowledge) subprotocol. From this, we compute $[d] = [c] - \beta[c_{\boldsymbol{w}}] - \beta^2 t[Q]$. This constitutes an accepting transcript. Note that in each step, non-uniformly distributed elements (which lead to an accepted response) are unique. Thus, we see that, conditioned on the event that all (combined) transition matrices are surjective, the simulation is perfect. The probability for the event is at least $1 - \eta$. □

[34] This can be seen by using that the constraint rows are linearly independent of the remaining rows. More generally, if the intersection of the vector space spanned by the constraint rows and the vector space spanned by the remaining rows is $\{0\}$, then choosing $\boldsymbol{r}'$ (resp. $\boldsymbol{r}''$) uniformly but subject to the constraints still results in uniformly random images under $\widetilde{\boldsymbol{M}}'$ (resp. $\widetilde{\boldsymbol{M}}'$).

[35] This is very naive. Actually, if either $\boldsymbol{M}'$ or $\boldsymbol{M}''$ is good, the response is perfectly randomised as well. (Because we compute $[\boldsymbol{u}_\ell']$ and $[\boldsymbol{u}_\ell'']$ individually.) Thus, the failure probability actually decreases.

[36] Yes, it is index ℓ again, because the $[\boldsymbol{u}_\ell'']$ is "swapped" in this linear sum of protocols.

4.5 Quadratic equation satisfiability

We can finally instantiate our sketch of an argument system for satisfiability of a system of quadratic equations from Section 4.3. It is a commit-and-prove system as follows. The prover commits to the solution $\boldsymbol{w}$. Then $\boldsymbol{\Gamma}$ is fixed and $\langle \boldsymbol{w}, \boldsymbol{\Gamma w} \rangle = 0$ shown to hold. The commitment scheme pads $\boldsymbol{w} \in \mathbb{F}_p^{n-2}$ with randomness and extends $\boldsymbol{\Gamma}$ in a suitable way. Intuition for soundness is given in Section 4.3.

Protocol 4.7 ($\mathsf{QESA}_{\mathrm{ZK}}$). Let $\boldsymbol{\Gamma}_i \in \mathbb{F}_p^{(n-2)\times(n-2)}$ $(i = 1, \dots, N)$ be a system of quadratic eqations. Suppose $N \geq 2$.[37] Let $\boldsymbol{w} \in \mathbb{F}_p^{n-2}$ be a solution, i.e. $\boldsymbol{w}^\top \boldsymbol{\Gamma}_i \boldsymbol{w} = 0$ for all i. We assume that the first component w_1 of $\boldsymbol{w}$ is 1.

Let $\mathrm{crs} = [\boldsymbol{g}', \boldsymbol{g}'', Q]$, $\widetilde{\chi}_{2k-1}$, $\chi^{(\beta\neq 0)}$ and $n \geq 4k$ as in Protocol 4.3, and $\mathbb{M}_n^+$ as in Lemma 4.6. Let $\boldsymbol{x} \leftarrow \chi_N$ be a testing distribution with $x_1 = 1$ and $x_2 \neq 0$ for all $\boldsymbol{x}$.[38] Let $\boldsymbol{y} \leftarrow \chi_{n+1}$ be a testing distribution with $y_1 = 1$ always. The following is a protocol for proving

$$\exists \boldsymbol{w} \in \mathbb{F}_p^{n-2}\colon \quad \forall i\colon \boldsymbol{w}^\top \boldsymbol{\Gamma}_i \boldsymbol{w} = 0$$

where crs and $\boldsymbol{\Gamma}_i$ are common inputs and the prover's witness is $\boldsymbol{w}$.

- $\mathscr{P} \to \mathscr{V}$: (Step 0: Commitment.) Pick $\boldsymbol{r}' \leftarrow \mathbb{F}_p^2$. Let the "extended" witness be $\boldsymbol{w}' := \left(\begin{smallmatrix} \boldsymbol{w} \\ \boldsymbol{r}' \end{smallmatrix}\right)$ and compute the commitment $[c'_{\boldsymbol{w}}] = [\boldsymbol{g}']\boldsymbol{w}'$. Send $[c'_{\boldsymbol{w}}]$.
- $\mathscr{V} \to \mathscr{P}$: (Step 1: Batch verification.) Pick and send $\boldsymbol{x} \leftarrow \chi_N$.
- (Batch equations): Both parties compute $\boldsymbol{\Gamma} := \sum x_i \boldsymbol{\Gamma}_i \in \mathbb{F}_p^{(n-2)\times(n-2)}$.
- (Fix w_1 to 1): Both parties let $\beta := x_2$ and do: Redefine $[g'_1]$ as $\beta^{-1}[g'_1]$. Redefine $[c'_{\boldsymbol{w}}] \leftarrow [c'_{\boldsymbol{w}}] - (\beta - 1)[g'_1]$ (with the new $[g'_1]$).
- $\mathscr{P} \to \mathscr{V}$: Let $\boldsymbol{r}'' = \boldsymbol{R}\boldsymbol{r}'$ where $\boldsymbol{R} = \left(\begin{smallmatrix} 0 & -1 \\ 1 & 0 \end{smallmatrix}\right)$ is a rotation by 90 degrees. Let $\boldsymbol{w}'' = \left(\begin{smallmatrix} \boldsymbol{\Gamma w} \\ \boldsymbol{r}'' \end{smallmatrix}\right)$. Compute and send $[c''_{\boldsymbol{w}}] = [\boldsymbol{g}'']\boldsymbol{w}''$.
- $\mathscr{V} \to \mathscr{P}$: Pick $(1, \boldsymbol{s}, \boldsymbol{b}) \leftarrow \chi_{n+1}$, where $\boldsymbol{s} \in \mathbb{F}_p^{n-2}$, $\boldsymbol{b} \in \mathbb{F}_p^2$. Send $\boldsymbol{s}' := \left(\begin{smallmatrix} \boldsymbol{s} \\ \boldsymbol{b} \end{smallmatrix}\right)$.
- $\mathscr{P} \leftrightarrow \mathscr{V}$: Engage in Protocol $\mathsf{IPA}_{\mathrm{almZK}}$ for $\langle \boldsymbol{w}' - \boldsymbol{s}', \boldsymbol{w}'' + \boldsymbol{\Gamma}'^\top \boldsymbol{s}' \rangle = t$ with $t = -\langle \boldsymbol{s}, \boldsymbol{\Gamma}^\top \boldsymbol{s} \rangle$, and commitment $([c'_{\boldsymbol{w}}] - [\boldsymbol{g}']\boldsymbol{s}') + ([c''_{\boldsymbol{w}}] + [\boldsymbol{g}'']\boldsymbol{\Gamma}'^\top \boldsymbol{s}')$ and the modified $[\boldsymbol{g}']$ (and unmodified $[\boldsymbol{g}'']$, $[Q]$) as commitment keys. Here $\boldsymbol{\Gamma}' = \left(\begin{smallmatrix} \boldsymbol{\Gamma} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R} \end{smallmatrix}\right) \in \mathbb{F}_p^{n\times n}$ where $\boldsymbol{R}$ is as in Step 1.

See Appendix G for a sketch of this protocol.

Remark 4.8. It is not hard to see that the prover never needs to compute $[c] = ([c'_{\boldsymbol{w}}] - [\boldsymbol{g}']\boldsymbol{s}') + ([c''_{\boldsymbol{w}}] + [\boldsymbol{g}'']\boldsymbol{\Gamma}'^\top \boldsymbol{s}')$. (In general, $\mathscr{P}$ does not need $[\boldsymbol{u}_0]$.) While the verifier has to check $[c]$, using lazy evaluation and optimisations from [16], this hardly affects its runtime. All in all, dealing with $\boldsymbol{s}'$ is almost free.

Lemma 4.9. *Protocol* $\mathsf{QESA}_{\mathrm{ZK}}$ *has perfect correctness.*

Using $\langle \left(\begin{smallmatrix} \boldsymbol{u}' \\ \boldsymbol{r}' \end{smallmatrix}\right), \left(\begin{smallmatrix} \boldsymbol{u}'' \\ \boldsymbol{r}'' \end{smallmatrix}\right) \rangle = \langle \boldsymbol{u}', \boldsymbol{u}'' \rangle + \langle \boldsymbol{r}', \boldsymbol{r}'' \rangle$ and $\langle \boldsymbol{r}, \boldsymbol{Rr} \rangle = 0$ for all $\boldsymbol{r} \in \mathbb{F}_p^2$, this is a straightforward check.

Proof. $\mathsf{IPA}_{\mathrm{almZK}}$ is perfectly correct and the verifier only rejects an honest prover if the $\mathsf{IPA}_{\mathrm{almZK}}$ rejects. So consider the statment $\langle \boldsymbol{w}' - \boldsymbol{s}', \boldsymbol{w}'' + \boldsymbol{\Gamma}'^\top \boldsymbol{s}' \rangle = t$, for $t = -\langle \boldsymbol{s}, \boldsymbol{\Gamma}^\top \boldsymbol{s} \rangle$. which the IPA proves. Let $\boldsymbol{w}' = \left(\begin{smallmatrix} \boldsymbol{w} \\ \boldsymbol{r}' \end{smallmatrix}\right)$, $\boldsymbol{w}'' = \left(\begin{smallmatrix} \boldsymbol{\Gamma w} \\ \boldsymbol{Rr}' \end{smallmatrix}\right)$ as in the protocol and let $\boldsymbol{s}' = \left(\begin{smallmatrix} \boldsymbol{s} \\ \boldsymbol{b} \end{smallmatrix}\right)$ with $\boldsymbol{b}$, $\boldsymbol{s}$ as in $\mathsf{QESA}_{\mathrm{ZK}}$. By construction, $\langle \boldsymbol{x}, \boldsymbol{Rx} \rangle = 0$ for any $\boldsymbol{x} \in \mathbb{F}_p^2$. With this, we find

$$\begin{aligned} \langle \boldsymbol{w}' - \boldsymbol{s}', \boldsymbol{w}'' + \boldsymbol{\Gamma}'^\top \boldsymbol{s}' \rangle &= \langle \boldsymbol{w} - \boldsymbol{s}, \boldsymbol{\Gamma w} + \boldsymbol{\Gamma}^\top \boldsymbol{s} \rangle + \langle \boldsymbol{r}' - \boldsymbol{b}, \boldsymbol{Rr}' + \boldsymbol{R}^\top \boldsymbol{b} \rangle \\ &= -\langle \boldsymbol{s}, \boldsymbol{\Gamma}^\top \boldsymbol{s} \rangle \end{aligned}$$

because from $\boldsymbol{R}^\top = -\boldsymbol{R}$ and $\boldsymbol{\Gamma w} = \boldsymbol{w}$ we have

$$\begin{aligned} \langle \boldsymbol{r}' - \boldsymbol{b}, \boldsymbol{Rr}' + \boldsymbol{R}^\top \boldsymbol{b} \rangle &= \langle \boldsymbol{r}' - \boldsymbol{b}, \boldsymbol{R}(\boldsymbol{r}' - \boldsymbol{b}) \rangle = 0 \\ \text{and} \quad \langle \boldsymbol{w} - \boldsymbol{s}, \boldsymbol{\Gamma w} + \boldsymbol{\Gamma}^\top \boldsymbol{s} \rangle &= \underbrace{\langle \boldsymbol{w}, \boldsymbol{\Gamma w} \rangle}_{=0} + \underbrace{(\langle \boldsymbol{w}, \boldsymbol{\Gamma}^\top \boldsymbol{s} \rangle - \langle \boldsymbol{s}, \boldsymbol{\Gamma w} \rangle)}_{=\langle \boldsymbol{\Gamma w} - \boldsymbol{w}, \boldsymbol{s} \rangle = 0} - \langle \boldsymbol{s}, \boldsymbol{\Gamma}^\top \boldsymbol{s} \rangle \end{aligned}$$

and thus

$$\langle \boldsymbol{w}' - \boldsymbol{s}', \boldsymbol{w}'' + \boldsymbol{\Gamma}'^\top \boldsymbol{s}' \rangle = -\langle \boldsymbol{s}, \boldsymbol{\Gamma}^\top \boldsymbol{s} \rangle. \qquad \square$$

[37] Otherwise, add trivial equations $\boldsymbol{\Gamma} = \boldsymbol{0}$.

[38] Restrictions on χ_N are merely to simplify protocol description and proofs.

Lemma 4.10. *Protocol* $\mathsf{QESA}_{\mathrm{ZK}}$ *has* μ*-special soundess (with* $\mu = (N, n+1, 2, 2, 2k, \ldots, 2k)$*) for extracting a witness or a non-trivial kernel element of* $[\boldsymbol{g}', \boldsymbol{g}'', Q]$*. It inherits short-circuit extraction with* $\mu = (1, 1, 2, 2, k, \ldots, k)$*.*

We did away with "α" compared to Section 4.3 to improve soundness. Extracting a challenge $(\alpha, \boldsymbol{s})$ naively requires a $(3, n-2)$ subtree. Our construction only needs an $(n+1)$ sub-"tree".

Proof. First of all, note that the randomisation of $[g'_1]$ is as in Lemma 4.2. In particular, non-trivial kernel elements in $[\boldsymbol{g}'_{\mathrm{new}}]$ yield such in $[\boldsymbol{g}'_{\mathrm{old}}]$. Therefore, we need only consider $[\boldsymbol{g}'] = [\boldsymbol{g}_{\mathrm{old}}]$ in the following.

The proof is straightforward. First extract the subprotocol, using the guarantees of Corollary 4.4. We find $\boldsymbol{w}'_i, \boldsymbol{w}''_i, t_i$ with $[\boldsymbol{g}', \boldsymbol{g}'', Q]\begin{pmatrix} \boldsymbol{w}'_i \\ \boldsymbol{w}''_i \\ t_i \end{pmatrix}$. Note that these openings are *prior* to randomisation of $[g'_1]$. We may assume that $\boldsymbol{w}' = \boldsymbol{w}'_i$, $\boldsymbol{w}'' = \boldsymbol{w}''_i$, $0 = t = t_i = \langle \boldsymbol{w}', \boldsymbol{w}'' \rangle$ for all i, otherwise we get a non-trivial kernel element of $[\boldsymbol{g}', \boldsymbol{g}'', Q]$. In particular, if $\beta_1 \neq \beta_2$ for some transcript, we find $v'_1 = 1$ as in Lemma 4.2.

We're left with a $(N, n+1)$-tree of transcripts where the extracted witness $\boldsymbol{w}', \boldsymbol{w}''$ is fixed and satisfies

$$\langle \boldsymbol{w}' - \boldsymbol{s}', \boldsymbol{w}'' + \boldsymbol{\Gamma}'^{\top} \boldsymbol{s}' \rangle = -\langle \boldsymbol{s}, \boldsymbol{\Gamma}^{\top} \boldsymbol{s} \rangle$$

for each transcript with challenge $\boldsymbol{s}'$. Since $\langle \boldsymbol{s}', \boldsymbol{\Gamma}^{\top} \boldsymbol{s}' \rangle = \langle \boldsymbol{s}, \boldsymbol{\Gamma}^{\top} \boldsymbol{s} \rangle$ we find

$$\begin{aligned} 0 &= \langle \boldsymbol{w}' - \boldsymbol{s}', \boldsymbol{w}'' + \boldsymbol{\Gamma}'^{\top} \boldsymbol{s}' \rangle + \langle \boldsymbol{s}', \boldsymbol{\Gamma}^{\top} \boldsymbol{s}' \rangle \\ &= \langle \boldsymbol{w}', \boldsymbol{w}'' \rangle + \langle \boldsymbol{\Gamma}' \boldsymbol{w}' - \boldsymbol{w}'', \boldsymbol{s}' \rangle \\ &= (1, \boldsymbol{s}'^{\top}) \begin{pmatrix} \langle \boldsymbol{w}', \boldsymbol{w}'' \rangle \\ \boldsymbol{\Gamma}' \boldsymbol{w}' - \boldsymbol{w}'' \end{pmatrix} \end{aligned}$$

Since χ_{n+1} is a testing distribution (and $\left(\begin{smallmatrix} 1 \\ \boldsymbol{s}' \end{smallmatrix}\right) \leftarrow \chi_{n+1}$), we find that $\langle \boldsymbol{w}', \boldsymbol{w}'' \rangle = 0$ and $\boldsymbol{\Gamma}' \boldsymbol{w}' - \boldsymbol{w}'' = 0$. Now, let $\boldsymbol{w}' = \left(\begin{smallmatrix} \boldsymbol{u}' \\ \boldsymbol{r}' \end{smallmatrix}\right)$ with $\boldsymbol{r}' \in \mathbb{F}_p^2$, $\boldsymbol{u}'$ in $\mathbb{F}_p^{n-2}$ and likewise $\boldsymbol{w}'' = \left(\begin{smallmatrix} \boldsymbol{u}'' \\ \boldsymbol{r}'' \end{smallmatrix}\right)$. We find (using the block structure of $\boldsymbol{\Gamma}'$) that

$$\boldsymbol{r}'' = \boldsymbol{R} \boldsymbol{r}' \quad \text{and} \quad \boldsymbol{u}'' = \boldsymbol{\Gamma} \boldsymbol{u}'.$$

Thus, we get

$$0 = \langle \boldsymbol{w}', \boldsymbol{w}'' \rangle = \underbrace{\langle \boldsymbol{r}', \boldsymbol{r}'' \rangle}_{=0} + \langle \boldsymbol{u}', \boldsymbol{u}'' \rangle.$$

Consequently, $\boldsymbol{w} := \boldsymbol{u}'$ is a witness with $\boldsymbol{w}^{\top} \boldsymbol{\Gamma} \boldsymbol{w} = 0$. We get $w_1 = 1$ because we are guaranteed to have two different β's (since all $n+1$ challenges are linearly independent and $x_1 = 1$ always).

What's left to show is that given N transcripts (with linearly independent challenges), we must have a solution (or a non-trivial kernel element). Since we assume that all extractions yield the same $\boldsymbol{w}', \boldsymbol{w}''$, hence the same $\boldsymbol{w}$, we show that $\boldsymbol{w}$ must be a witness. Consider the vector $\boldsymbol{e} \in \mathbb{F}_p^N$ defined by $e_j := \boldsymbol{w}^{\top} \boldsymbol{\Gamma}_j \boldsymbol{w}$. Write $\boldsymbol{\Gamma}^{(i)} := \sum_j x_j^{(i)} \boldsymbol{\Gamma}_j$, where the superscript i indicates the i-th transcript. Since all transcripts are valid, we know that $\boldsymbol{w}^{\top} \boldsymbol{\Gamma}^{(i)} \boldsymbol{w} = 0$. We get

$$0 = \boldsymbol{w}^{\top} \boldsymbol{\Gamma}^{(i)} \boldsymbol{w} = \sum_j x_j^{(i)} \boldsymbol{w}^{\top} \boldsymbol{\Gamma}_j \boldsymbol{w} = \boldsymbol{x}^{(i)} \boldsymbol{e}.$$

Since all $\boldsymbol{x}^{(i)}$ are linearly independent this implies $\boldsymbol{e} = 0$, i.e. $\boldsymbol{w}$ solves each equation. □

Lemma 4.11. *Protocol* $\mathsf{QESA}_{\mathrm{ZK}}$ *is* ε*-statistical zero-knowledge for some* $\varepsilon \in O(\log_k(n)k)/p$*.*

For the proof, we establish that the conditions of Lemma 4.6 are met except with probability $O(\log_k(n)k)/p$. This follows essentially because $\mathsf{QESA}_{\mathrm{ZK}}$ uses $\boldsymbol{w}' = \left(\begin{smallmatrix} \boldsymbol{w} \\ \boldsymbol{r} \end{smallmatrix}\right)$, where $\boldsymbol{r}$ is *random* (and similar for $\boldsymbol{w}''$). Thus, $\mathsf{IPA}_{\mathrm{almZK}}$ is statistical zero-knowledge, and consequently $\mathsf{QESA}_{\mathrm{ZK}}$ is statistical zero-knowledge as well.

Proof. We use Lemma 4.6 to show that (with probability $O(\log_k(n)k)/p$), the prover's messages in the IPA subprotocol are all uniformly random. This will allow us to simulate the IPA subprotocol. The rest is standard and straightforward.

Namely, assume that the subprotocol is zero-knowledge. Our simulator then first picks $[c'], [c''] \leftarrow \mathbb{G}$ uniformly at random. Note that the randomisation terms $\boldsymbol{r}'$ resp. $\boldsymbol{r}'' = \boldsymbol{R}\boldsymbol{r}'$ in Step 0 (resp. Step 1) ensure that $[c']$ and $[c'']$ are uniformly distributed in the honest protocol as well. Then we compute $[c'_{\mathrm{new}}]$ as in the protocol. Finally, we simulate the IPA subprotocol (for otherwise "honestly" computed inputs).

Now, let us show that in an honest protocol run, the prover satisfies the requirement of Lemma 4.6 with probability $O(\log_k(n)k)/p$.

Let $\boldsymbol{u}' := \boldsymbol{w}' - \boldsymbol{s}'$, $\boldsymbol{u}'' := \boldsymbol{w}'' + \boldsymbol{\Gamma}'^{\top}\boldsymbol{s}'$ as in Protocol 4.7 ($\mathsf{QESA}_{\mathrm{ZK}}$). Note that u'_n and u''_n are (independent and) uniformly random: $u'_n = r'_1 - s'_1$ and $u''_n = r'_2 + s'_2$ (and $\boldsymbol{r}' = (r'_1, r'_2)^\top \in \mathbb{F}_p^2$ is uniformly random.

Let $\boldsymbol{M}''$ and $\boldsymbol{C}''$ be as in Lemma 4.6. (The proof for $\boldsymbol{M}'$ is analogous, and simpler.) We specialise to $k = 2$. For general k one argues analogously.

Consider the columns $\{1, \dots, 4\}$ and $\{n-3, \dots, n\}$ of $\boldsymbol{M}''$. In other words, take the first 4 and last 4 columns of $\boldsymbol{M}''$. Note that $\mathbb{M}_n^+$ ($n \geq 8$) contains all of these by construction. After removing 0-rows (from the x_i id blocks) we end up with:

$$\boldsymbol{M}'' := \left(\begin{array}{cccc|cccc}
u_1 & u_2 & u_3 & u_4 & u_{n-3} & u_{n-2} & u_{n-1} & u_n \\
r_1 & r_2 & r_3 & r_4 & \star & \star & r_{n-1} & r_n \\
g_{2,1} & g_{2,2} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & g_{1,1} & g_{1,2} & g_{1,3} & g_{1,4} \\
\hline
x_1 & & & & x_2 & & & \\
 & x_1 & & & & x_2 & & \\
 & & x_1 & & & & x_2 & \\
 & & & x_1 & & & & x_2
\end{array}\right)$$

Empty entries are 0, $\star$ entries don't matter. We dropped all "primes" to simplify notation. Furthermore, the r_i belong to $\boldsymbol{r}''$ from Lemma 4.6.

Note that $\boldsymbol{r}'' \leftarrow K := \ker(\binom{\boldsymbol{u}'}{\boldsymbol{r}'}) \cap \mathbb{M}_n^+$ (for the $\boldsymbol{r}'$ of Lemma 4.6, which is of no further interest). In particular, $\dim(K) \geq \dim(\mathbb{M}_n^+) - 2$. Thus, we find some index $j \in \{1, 2, 3, 4\}$ such that $r_j \in \mathbb{F}_p$ is distributed uniformly random (in fact, we find at least two such indices). Suppose r_2 is uniformly random. The other cases can be handled analogously.

We now develop the determinant of above matrix (row-wise). We pick the terms u_n, r_2, $g_{2,1}$ and $g_{1,3}$ for this. We find

$$\pm \det(\boldsymbol{M}'') = v_n \cdot r_2 \cdot g_{2,1} \cdot g_{1,3} \cdot \underbrace{\det \begin{pmatrix} & & x_2 & \\ & & & x_2 \\ x_1 & & & \\ & x_1 & & \end{pmatrix}}_{\neq 0} + \mathsf{poly}$$

where poly is some polynomial (in the entries of $\boldsymbol{M}''$) whose monomials are different from $v_n r_2 g_{2,1} g_{1,3}$. Since u_1, r_3, $g_{2,1}$ and $g_{1,4}$ are distributed uniformly at random, the Lemma of Schwatz–Zippel implies that $\mathbb{P}(\det(\boldsymbol{M}'') = 0) \leq 4/p$ for any (fixed) $x_1, x_2 \neq 0$. □

4.6 Combining $\mathsf{QESA}_{\mathrm{ZK}}$ with other proof systems

As is, $\mathsf{QESA}_{\mathrm{ZK}}$ can be used to commit-and-prove quadratic equations. However, oftentimes, one wishes to prove statements about commitments which come from some other source (and do not include auxiliary information necessary for a proof). For example, Bulletproofs [16] were designed for confidential transaction, where the commitments are *input* to the proof system. This is not immediately feasible with $\mathsf{QESA}_{\mathrm{ZK}}$ as is, because $\mathsf{QESA}_{\mathrm{ZK}}$ is commit-and-prove only w.r.t. the solution of the set of quadratic equations. So either the commitment *includes* the auxiliary information (e.g. a bit decomposition for range proofs), or $\mathsf{QESA}_{\mathrm{ZK}}$ is not directly applicable.

Fortunately, applying $\mathsf{QESA}_{\mathrm{ZK}}$ in such circumstances is not hard. Although, since $\mathsf{QESA}_{\mathrm{ZK}}$ only assures a solution to a set of quadratic equations, some additional steps (beyond $\mathsf{QESA}_{\mathrm{ZK}}$'s guarantees) may be necessary to achieve the desired properties.

We consider following setting. There are commitment keys $\widetilde{\mathrm{ck}}^{(i)}$ for $i = 1, \dots, M$. Each commitment key corresponds to a subset $\mathcal{I}_i \subseteq \{1, \dots, n\}$ of the components of $[\boldsymbol{g}']$, where $\mathrm{crs} = ([\boldsymbol{g}', \boldsymbol{g}'', Q])$ is the commitment key of $\mathsf{QESA}_{\mathrm{ZK}}$. That is $\widetilde{\mathrm{ck}}^{(i)} \mathrel{\widehat{=}} \{[g'_j]\}_{j \in \mathcal{I}_i}$. Let $\mathcal{I} := \cup_{i=1}^{M} \mathcal{I}_i$ be the set of all indices which are part of some $\widetilde{\mathrm{ck}}^{(i)}$. Let $M^{(i)} := \#\mathcal{I}_i$ be the size of $\widetilde{\mathrm{ck}}^{(i)}$. We assume the following: Every commitment key $\widetilde{\mathrm{ck}}^{(i)}$ uses $[g'_n]$ (or $[g'_{n-1}]$) as its randomness components. Moreover, $1 \notin \mathcal{I}_i$, because the index $1 \mathrel{\widehat{=}} [g'_1]$ is reserved for the commitment to value 1 in $\mathsf{QESA}_{\mathrm{ZK}}$. A useful point of view is that $\widetilde{\mathrm{ck}}^{(i)}$ is a commitment under $[\boldsymbol{g}'] \in \mathbb{G}^n$ to a vector $\boldsymbol{v}^{(i)} \in \mathbb{F}_p^n$ with

$$\forall i \notin \mathcal{I}_i \colon v_i = 0. \tag{4.2}$$

We assume for simplicity that there is only one commitment per commitment key $\widetilde{\mathrm{ck}}^{(i)}$. To model the case of multiple commitments $[c_1], \dots, [c_M]$ for one key, e.g. all commitments are under $\widetilde{\mathrm{ck}} = \widetilde{\mathrm{ck}}^{(1)}$, we simply duplicate $\widetilde{\mathrm{ck}}$, i.e. we rewrite this as $[\widetilde{c}^{(i)}] = [c_i]$, $\widetilde{\mathrm{ck}}^{(i)} = \widetilde{\mathrm{ck}}$.

Example 4.12. In a typical range proof, with Pedersen committed value, we would have $\widetilde{\mathrm{ck}}^{(1)} \mathrel{\widehat{=}} [g'_2, g'_n]$, where $M = 1$. We write $\widetilde{\mathrm{ck}} := \widetilde{\mathrm{ck}}^{(1)}$ for simplicity. This means $\mathcal{I} = \{2, n\}$.

Remark 4.13. Using the in n varying $[g'_n]$ in the commitment keys $\widetilde{\mathrm{ck}}^{(i)}$ is problematic and inconvenient. We want the randomness terms in $\mathsf{QESA}_{\mathrm{ZK}}$ and our commitment to "overlap". But now, running $\mathsf{QESA}_{\mathrm{ZK}}$ for a smaller or larger instance, e.g. an instance of size $n/2$ or $2n$ is incompatible. A simple solution is to fix some (random) $[g'^{,\star}_{\mathtt{rnd1}}, g'^{,\star}_{\mathtt{rnd2}}]$ (as part of crs) and construct $[\boldsymbol{g}']$ when starting Protocol $\mathsf{QESA}_{\mathrm{ZK}}$ so that $[g'_{n-1}, g'_n] = [g'^{,\star}_{\mathtt{rnd1}}, g'^{,\star}_{\mathtt{rnd2}}]$. Another solution is to permute the position of the randomness and reserve the fixed indices $2, 3$ for randomness (instead of $n-1, n$). Either approach fixes the group elements corresponding to the randomising term, solving the problem.

With this setup, we can extend $\mathsf{QESA}_{\mathrm{ZK}}$ as follows: Given commitments $[\widetilde{c}^{(i)}]$ under keys $\widetilde{\mathrm{ck}}^{(i)}$, prove that the values committed in $[\widetilde{c}^{(i)}]$ satisfy some set of quadratic equations. In other words, prove that the $[\widetilde{c}^{(i)}]$ satisfy some arithmetic circuit.

Example 4.14 (Aggregate range proof). Consider $[\widetilde{c}^{(j)}]$, $j = 1, ..., 10$. We wish to prove that the values $\boldsymbol{v}^{(j)}$ committed in $[\widetilde{c}^{(j)}]$ all lie in the range $\{0, \dots, 2^8 - 1\}$. We can also prove additional properties, like $\boldsymbol{v}^{(j)} \leq \boldsymbol{v}^{(j+1)}$ for all j.

Unsurprisingly, our solution to the problem is probabilistic verification. On a high level, we proceed as follows: The verifier knows the commitments $[\widetilde{c}^{(i)}]$ as part of the statement. We start $\mathsf{QESA}_{\mathrm{ZK}}$ as usual, the prover sends the commitment $[c'_{\boldsymbol{w}}]$ to the witness, where the components $\mathcal{I}$ are zeroed (except for the randomness in $n-1, n$). Then the verifier sends a challenge $\boldsymbol{\alpha}\mathbb{F}_p^{M(+)1}$ with $\alpha_0 = 1$. Both sides compute the random linear combination $[c'_{\boldsymbol{w}}] \leftarrow [c'_{\boldsymbol{w}}] + \sum_i \alpha_i [\widetilde{c}^{(i)}]$ as the new commitment. The prover adjusts his (extended) witness $\boldsymbol{w}' = \left(\begin{smallmatrix}\boldsymbol{w}\\ \boldsymbol{r}'\end{smallmatrix}\right)$ to $\boldsymbol{w}' \leftarrow \alpha_0 \boldsymbol{w}' + \sum_i \alpha_i \boldsymbol{v}^{(i)}$. The statements, i.e. the matrices $\boldsymbol{\Gamma}_i$ are also adjusted, and perhaps additional equations are included.

For a single commitment, the strategy can be made to work as described. For multiple commitments, it depends on the statement. However, there is one problem which this kind of probabilistic verification does not address: It is never proven that the commitments $[\widetilde{c}^{(i)}]$ actually satisfy Eq. (4.2). In general, this is *not* implied. We can either assume that this is enforced by means outside of the protocol, or add proof for this, e.g. another proof of knowledge.

Example 4.15. Suppose there is no statement to verify, i.e. $\boldsymbol{\Gamma} = 0$. Then $\mathsf{QESA}_{\mathrm{ZK}}$ and its modification above degrades to a proof of knowledge of an opening. But it does *not* prove that $\boldsymbol{v}^{(i)}$ satisfies Eq. (4.2). If there are components of the witness, which are unused (because less than $2^n - 3$ values are needed in the proof), then similar problems apply. So this is a practical problem. It is possible to (artificially) zero all unused components. But even then, there are (trivial) statements where Eq. (4.2) is violated.

Our idea for general interoperability is as follows: The initial $\mathsf{QESA}_{\mathrm{ZK}}$ witness $\boldsymbol{w}$ (commitment $c'_{\boldsymbol{w}}$) has all components in $\mathcal{I}$ zeroed (except for randomness $n-1$, n) and also contains *copies* of the committed $\boldsymbol{v}^{(i)}$. The actual equations, i.e. the $\boldsymbol{\Gamma}_i$, only refer to the copies and the components $\mathscr{I}$. As before, for verifier randomness $\boldsymbol{\alpha}$, we set $[c'_{\boldsymbol{w}}] \leftarrow [c'_{\boldsymbol{w}}] + \sum_i \alpha_i [\widetilde{c}^{(i)}]$, and obtain $\boldsymbol{w}' \leftarrow \boldsymbol{w}' + \sum_i \alpha_i \boldsymbol{v}^{(i)}$ as new extended witness. Note that all (extended) equations $\boldsymbol{w}'\boldsymbol{\Gamma}_i'^{\top}\boldsymbol{w}'$ still hold (for an honest prover). Now we add (linear) equations $\boldsymbol{\Gamma}^{(i)}_{\text{copy}}$ to the statement, which we call *copy-equations* and which depend on the randomness α_i. These equations simply assert that, if we compute $\sum_i \alpha_i \boldsymbol{v}^{(i)}$ using the committed copies in $\boldsymbol{w}$, then this equals the values committed in components $\mathscr{I}$ (again excluding the randomness components $n-1$, n). In other words, we assert that the purported copies of $\boldsymbol{v}^{(i)}$ in witness $[c'_{\boldsymbol{w},\text{old}}]$ were valid copies. This "copy-based" approach is simple and modular.

The formulaic description of $\mathsf{QESA}_{\text{Copy}}$ is arguably technical. However, the examples in Fig. 3 and Fig. 4 demonstrate that it is actually a simple concept.

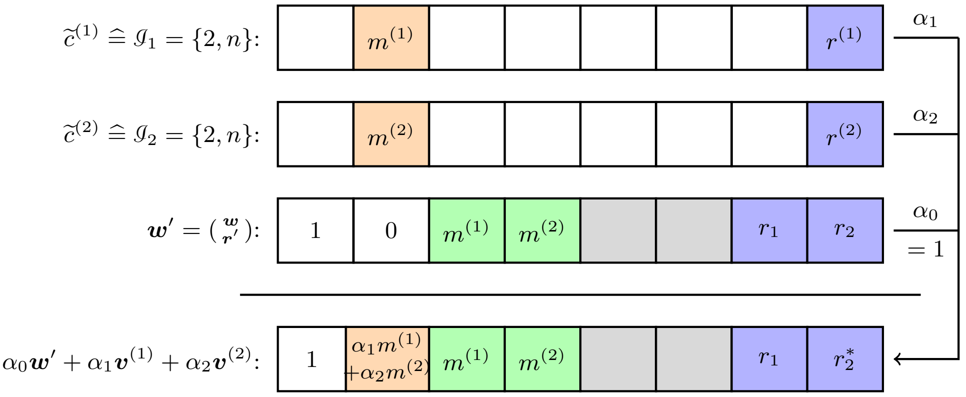
$\widetilde{c}^{(1)} \mathrel{\widehat{=}} \mathscr{I}_1 = \{2, n\}$:
$m^{(1)}$
$r^{(1)}$
α_1
$\widetilde{c}^{(2)} \mathrel{\widehat{=}} \mathscr{I}_2 = \{2, n\}$:
$m^{(2)}$
$r^{(2)}$
α_2
$\boldsymbol{w}' = (\begin{smallmatrix}\boldsymbol{w}\\ \boldsymbol{r}'\end{smallmatrix})$:
1
0
$m^{(1)}$
$m^{(2)}$
r_1
r_2
α_0
$= 1$
$\alpha_0\boldsymbol{w}' + \alpha_1\boldsymbol{v}^{(1)} + \alpha_2\boldsymbol{v}^{(2)}$:
1
$\alpha_1 m^{(1)}$
$+\alpha_2 m^{(2)}$
$m^{(1)}$
$m^{(2)}$
r_1
r_2^*

Figure 3: An example of a copying two values from two commitments. The blocks are colour-coded as follows: White blocks contain either 0 or the value indicated. Orange blocks belong to the (value-part) of commitment indices, i.e. to $\mathscr{I}$. Green blocks denote "copied" values. Gray blocks contain an arbitrary value. Blue blocks refer to randomness parts (i.e. components $n-1, n$). Note that randomness is *not* copied, as it is not a relevant part of the committed value. It is simply accumulated in $r_2^* = \alpha_0 r_2 + \alpha_1 r^{(1)} + \alpha_2 r^{(2)}$. The actual statements (i.e. the matrices $\boldsymbol{\Gamma}_i$) are statements over all variables except the orange (and blue) blocks, as these are merely "test-values" which ensure that $\boldsymbol{w}$ contains copies of (the message part of) $\boldsymbol{v}^{(i)}$, here $m^{(i)}$, as claimed.

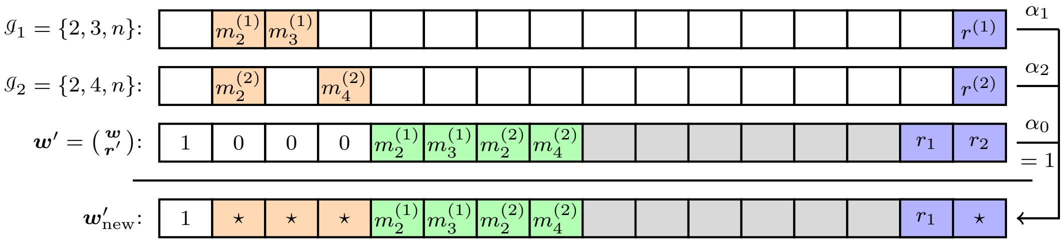

Figure 4: This is a more complex example of the copying technique. Colour-coding is as before. Note that $\mathscr{I}_1 \neq \mathscr{I}_2$. Again, all orange values $\boldsymbol{m}^{(i)} \mathrel{\widehat{=}} \boldsymbol{v}^{(i)}$, are copied and appear as green values in $\boldsymbol{w}$. Note that we can go much further than this: Green values could be implicitly given by quadratic equations, as noted in Remark 4.18. The copy for a one commitment, e.g. $[\widetilde{c}^{(1)}]$ could be elided, c.f. Remark 4.18.

Protocol 4.16 ($\mathsf{QESA}_{\text{Copy}}$). Let $n \geq 4k$, $\boldsymbol{\Gamma}_i$, $\mathrm{crs} = [\boldsymbol{g}', \boldsymbol{g}'', Q]$, $\widetilde{\chi}_{2k-1}$ be as in Protocol $\mathsf{QESA}_{\mathsf{ZK}}$. Let $\widetilde{\chi_{M+1}}$ be a testing distribution where the first component is always 1, i.e. $\boldsymbol{\alpha} \leftarrow \chi_{M+1}$ has $\alpha_0 = 1$.[39] Let $\widetilde{\mathrm{ck}}^{(i)} \mathrel{\widehat{=}} \mathscr{I}_i$ be commitment keys for commitments (for $i = 1, \ldots, M$), as described above. Let $[\widetilde{c}^{(i)}]$ be commitments to values $\boldsymbol{v}^{(i)}$. We identify $\boldsymbol{v}^{(i)}$ with a vector in $\mathbb{F}_p^n$ when necessary (satisfying Eq. (4.2)). Let $\boldsymbol{w} \in \mathbb{F}_p^{n-2}$ be a solution, i.e. $\boldsymbol{w}^{\top}\boldsymbol{\Gamma}_i\boldsymbol{w} = 0$ for all i. We assume that the first component w_1 of $\boldsymbol{w}$ is 1 and

$$\forall i \in \mathscr{I}_i \cap \{1, \ldots, n-2\} \colon w_i = 0.$$

[39] The restriction $\alpha_0 = 1$ is just for convenience.

For simplicity, assume that the last component of $\boldsymbol{v}^{(i)}$ is used for commitment randomness and $\widetilde{\mathrm{ck}}^{(i)}_{M^{(i)}} = [g'_n]$, hence message size is $M^{(i)} - 1$. We assume there is an injective map τ with

$$\tau(i, \_)\colon \{1, \dots, M^{(i)} - 1\} \to \{1, \dots, n-2\} \setminus \mathcal{I} \quad \text{such that} \quad w_{\tau(i,j)} = \boldsymbol{v}^{(i)}_j,$$

which tells us components $\boldsymbol{v}^{(i)}_j$ of the committed message are copied to. This excludes the commitment randomness $\boldsymbol{v}^{(i)}_{M^{(i)}}$, which by assumption corresponds to component n and maps to component n.[40] In other words, the mapping τ tells us where copied components $\boldsymbol{v}^{(i)}_j$ are stored in $\boldsymbol{w}$. (We ignore commitment randomness at indices $n-1, n$, as this is of no interest,[41] and part of the extended witness $\boldsymbol{w}' = \binom{\boldsymbol{w}}{\boldsymbol{r}'} \in \mathbb{F}_p^n$.) Let $\mathcal{S}$ be a protocol proving knowledge of $\boldsymbol{v}^{(i)}$ for all i.[42] Then the following is a protocol for proving

$$\begin{aligned}\exists \boldsymbol{w} \in \mathbb{F}_p^{n-2}, \boldsymbol{v}^{(i)} \in \mathbb{F}_p^{M^{(i)}} \leq \mathbb{F}_p^n \text{ such that } \quad & \forall j\colon \boldsymbol{w}^\top \boldsymbol{\Gamma}_j \boldsymbol{w} = 0 \\ \text{and} \quad & \forall i \, \forall j \in \mathcal{I} \cap \{1, \dots, n-2\}\colon w_{\tau(i,j)} = \boldsymbol{v}^{(i)}_j \\ \text{and} \quad & \forall i\colon [\widetilde{c}^{(i)}] = \widetilde{\mathrm{ck}}^{(i)} \boldsymbol{v}^{(i)} \mathrel{\widehat{=}} [\boldsymbol{g}'] \boldsymbol{v}^{(i)}.\end{aligned}$$

The prover's witness consists of $\boldsymbol{w}$ and $\boldsymbol{v}^{(i)}$. The statement consists of $\{\boldsymbol{\Gamma}_j\}_j$ and $[\widetilde{c}^{(i)}]$.

- $\mathcal{P} \to \mathcal{V}$: (Step -1: Prove well-formedness of $[\widetilde{c}^{(i)}]$)
  Engage in $\mathcal{S}$ to prove that $\exists \boldsymbol{v}^{(i)}\colon [\boldsymbol{g}]\boldsymbol{v}^{(i)} = [\widetilde{c}^{(i)}]$ and $\boldsymbol{v}^{(i)}$ satisfies Eq. (4.2). (This may be run in parallel.)
- $\mathcal{P} \to \mathcal{V}$: (Step 0: Commit to $\boldsymbol{w}$.) Send $[c'_{\boldsymbol{w}}] \coloneqq [\boldsymbol{g}']\boldsymbol{w}'$ with $\boldsymbol{w}' = \binom{\boldsymbol{w}}{\boldsymbol{r}'}$, where $\boldsymbol{w}$ is as outlined above and $\boldsymbol{r}' \leftarrow \mathbb{F}_p^2$.
- $\mathcal{V} \to \mathcal{P}$: (Step 1: Batch verification and statement adaption for $\mathsf{QESA}_{\mathrm{ZK}}$) Pick and send $\boldsymbol{\alpha} \leftarrow \chi_{M+1}$, where $(\alpha_0, \dots, \alpha_M) = \boldsymbol{\alpha} \in \mathbb{F}_p^{M+1}$ and where $\alpha_0 = 1$ always (by assumption). Both sides set $[c'_{\boldsymbol{w}}] \coloneqq \alpha_0 [c'_{\boldsymbol{w}}] + \sum_i \alpha_i [\widetilde{c}^{(i)}]$. The prover sets $\boldsymbol{w}' \coloneqq \alpha_0 \boldsymbol{w}' + \sum_i \alpha_i \boldsymbol{v}^{(i)} \in \mathbb{F}_p^n$. The set of equations is augmented by additional equations, given by "copy-matrices" $\boldsymbol{\Gamma}^{(k)}_{\mathrm{copy}}$ for each $k \in \mathcal{I} \cap \{1, \dots, n-2\}$ as follows:

  $$\boldsymbol{w}^\top \boldsymbol{\Gamma}^{(k)}_{\mathrm{copy}} \boldsymbol{w} = 0 \quad \widehat{=} \quad \sum_{\tau(i,k) = j \text{ if } k \in \mathcal{I}_i} \alpha_i w_{\tau(i,k)} - w_k = 0.$$

  These equations merely formalise that computing the (random) linear combination of the (purported) *copies* of $\boldsymbol{v}^{(i)}$ (as part of $\boldsymbol{w}'$) yield the same value as the (random) linear combination of the commitments, c.f. Figs. 3 and 4. (Note the linearity of the commitments).
  With these additional equations and the adapted witness, continue as in $\mathsf{QESA}_{\mathrm{ZK}}$ (Step 1) without further changes.

See Appendix G for a sketch of this protocol.

It not hard to see that $\mathsf{QESA}_{\mathrm{Copy}}$ is correct. For zero-knowledge, we merely note that $\mathsf{QESA}_{\mathrm{ZK}}$ is statistical HVZK, and by a completely analogous proof, $\mathsf{QESA}_{\mathrm{Copy}}$ is statistical HVZK as well.

Lemma 4.17. *Suppose $\mathcal{S}$ is ν-special sound. Then Protocol $\mathsf{QESA}_{\mathrm{Copy}}$ is (ν, μ)-special sound for extraction of a witness or a non-trivial kernel element of $[\boldsymbol{g}', \boldsymbol{g}'', Q]$, with $\mu = (M+1, N', n+1, 2, 2, 2k, \dots, 2k)$, where N' is the number of equations plus the number of copy equations. The μ-part is short-circuit extractable with $\mu' = (1, 1, 1, 2, 2, k, \dots, k)$.*

[40] This can trivially be relaxed to allowing randomness in components $n-1$ and n. By construction, the commitment randomness is not copied and cannot be reconstructed or used in the statements. If different components than $n-1$, n are used as commitment randomness, they are treated like a committed message, and (may be) copied as well.

[41] If commitment randomness is used in other indices, we treat it like the committed message, and do copy it.

[42] It suffices to prove that Eq. (4.2) holds for all i. If all $\mathcal{I}$ are equal, *dual testing* distributions can be used instead, to freshly generate all components g_i for $i \notin \mathcal{I}$. Remember that dual testing (Definition 2.16) ensures that previous commitments must be zero for all components $i \notin \mathcal{I}$ (or cannot be opened without breaking the hard kernel assumption). If not all $\mathcal{I}$ are equal, treating them as equal, i.e. "extending" the commitment keys $\widetilde{\mathrm{ck}}^{(i)}$ and proving that the copied values are zero outside $\mathcal{I}_i$, still allows to resort to dual testing.

Proof sketch. The proof is straightforward, but the indexing is tedious. We only sketch it.

First of all, note that just like with $\mathsf{QESA}_{\mathrm{ZK}}$, we can for each run with randomness $\boldsymbol{\alpha}$ extract a witness $\boldsymbol{w}_\alpha$ satisfying all $\boldsymbol{\Gamma}_i$, including the additional copy-equations. We only need to prove that (all) $\boldsymbol{w}_\alpha$ have correctly copied the values $\boldsymbol{v}^{(i)}$ of commitments $[\widetilde{c}^{(i)}]$. Since we assumed special soundness of the subprotocol $\mathcal{S}$, we could use it for extraction. However, we refrain from doing so, to sketch how the lemma and proof generalise to the setting, where $\mathcal{S}$ only ensures that any opening $\boldsymbol{v}^{(i)}$ satisfies Eq. (4.2). So, by soundness of $\mathcal{S}$, we can assume that for all components not in $\mathcal{I}$, we know that $\boldsymbol{w}_\alpha$ does not depend on $\boldsymbol{\alpha}$, i.e. is fixed. (Remember that $\alpha_0 = 1$, and $[c_{\boldsymbol{w}}]$ is the only commitment which is possibly non-zero in components outside $\mathcal{I}$.) In particular, the copies of $\boldsymbol{v}^{(i)}$ in $\boldsymbol{w}_\alpha$ are always identical, and do not depend on α. Otherwise we find a non-trivial kernel element. We show this in the following.

First, we note that from $M+1$ linearly independent challenges $\boldsymbol{\alpha}$, we obtain (since $[\widetilde{c}^{(i)}]$ and $[c'_{\boldsymbol{w}}]$ are fixed) openings to each commitment in the usual way. A priori, the openings $\boldsymbol{w}$ and $\boldsymbol{v}^{(i)}$ are only openings w.r.t. $[\boldsymbol{g}']$, and need not respect Eq. (4.2). However, due to subprotocol $\mathcal{S}$, we see that Eq. (4.2) holds for each i, or the soundness of $\mathcal{S}$ is broken.

Now, we reinterpret the setting to avoid carrying around too many indices. The "copy proofs" essentially state the following: There is a subvector $(\boldsymbol{a}, \boldsymbol{b}_1, \dots, \boldsymbol{b}_M)$ in $\boldsymbol{w}$ such that $\boldsymbol{b}_i$ consists of the copied values of all $\boldsymbol{v}^{(i)}$, and $\boldsymbol{a}$ should be zero. In $\boldsymbol{w}_\alpha$, we get $\boldsymbol{a}_\alpha = \boldsymbol{a} + \sum_{i=1}^M \alpha_i \boldsymbol{v}^{(i)}$. (Note that $\boldsymbol{a}$ *should* be zero, but we need to prove this.) By the "copy equations" from Step 1, we also have[43]

$$\sum_{i=1}^M \alpha_i \boldsymbol{b}_i = \boldsymbol{a}_\alpha = \alpha_0 \boldsymbol{a} + \sum_{i=1}^M \alpha_i \boldsymbol{v}^{(i)}.$$

Given $M+1$ linearly independent $\boldsymbol{\alpha}$, we find that $\boldsymbol{a} = 0$. Thus, we find that the $\boldsymbol{v}^{(i)}$ which satisfy these equations are openings of $[\widetilde{c}^{(i)}]$. If any of this fails, we find non-trivial kernel relations.

A priori, the opening $\boldsymbol{v}^{(i)}$ only respects $\mathcal{I}$, not $\mathcal{I}_i$, in the sense of Eq. (4.2). But another invocation of the soundness of $\mathcal{S}$ shows that they do respect $\mathcal{I}_i$. Thus, they are openings w.r.t. $\widetilde{\mathrm{ck}}^{(i)}$. (Actually, to get openings we need to look at the *randomness* too, i.e. include components $n-1, n$. It is not hard to do this.)

Now, we have openings for the commitments, we know that $\boldsymbol{w}$ actually contains copies of these commitments, we know that $\boldsymbol{w}$ has zeroed all w_i with $i \in \mathcal{I}$, and $\boldsymbol{w}$ satisfies the all equations $\boldsymbol{\Gamma}_i$. This is what we wanted to show. □

Remark 4.18. One can "optimise away" unnecessary copies. For example, if the value of a copy can be computed from other values by some quadratic function, then one can use this instead of making an explicit copy. This is the case for range proofs, where $\boldsymbol{v} = \sum_i 2^i b_i$ is is copied. The bit-decomposition of $\boldsymbol{v}$ is enough to recover it, so no extra copy of $\boldsymbol{v}$ is necessary. Evidently, the "copy-equations" must be adapted accordingly.

There is one further optimisation: In the case of range proofs, one does not need b_0 if one proves instead that $\boldsymbol{v} - \sum_{i\geq 1} 2^i b_i$ is a bit (which is a quadratic, even R1CS, equation). A priori, this is not possible with the copy approach. However, close inspection shows that when copying a *single* commitment, one can, instead of copying it, adapt all $\boldsymbol{\Gamma}_i$ of the statement by multiplying all rows and columns in $\mathcal{I}$ by α_1^{-1} (except randomness indicices, which are not even part of the equations).

In the "copy-based" case, we can combine both optimisations: Given (implicit) copies of $\boldsymbol{v}^{(2)}$, ..., $\boldsymbol{v}^{(M)}$, one can compute $\boldsymbol{v}^{(1)}$ from these and the sum $\sum_{i=1}^M \alpha_i \boldsymbol{v}^{(i)}$. By adapting all $\boldsymbol{\Gamma}_i$ with α_1^{-1} as before, we can make use of $\boldsymbol{v}^{(1)}$ implicitly, as $\alpha_1^{-1} \sum_{i=2}^M \alpha_i \boldsymbol{v}_{\mathrm{copy}}^{(i)}$. Thus, we need at most $M-1$ copies.

These optimisations are especially useful if a lot of components need to be copied, i.e. if $\#\mathcal{I}$ is large w.r.t. to the rest of the witness.

Example 4.19. Consider the situation of (aggregate) range proofs in [16], that is, we have a commitment key $\widetilde{\mathrm{ck}} := [g'_2, g'_n]$ and want to prove that commitments $[c_i]$, for $i = 1, \dots, M$, all under this key, contain values within a some ℓ-bit range. We would instantiate $\mathsf{QESA}_{\mathrm{Copy}}$ as follows: As the subprotocol $\mathcal{S}$, use a batch proof of knowledge of openings of $[c_i]$ (Appendix B). This requires transmitting 1 group

[43] If we don't have this, $\mathsf{QESA}_{\mathrm{ZK}}$ extraction would short-circuit.

Parameters	Bulletproofs		$\mathsf{QESA}_{\mathrm{RP}}$		$\mathsf{QESA}_{\mathrm{RP}}$ (short)	
	$\mathcal{P}$	$\mathcal{V}$	$\mathcal{P}$	$\mathcal{V}$	$\mathcal{P}$	$\mathcal{V}$
60 bit	0.26	0.17	0.16	0.07	0.15	0.06
60 bit × 2	0.47	0.29	0.32	0.15	0.30	0.10
60 bit × 32	7.4	4.5	5.1	2.4	4.6	1.7
60 bit × 128	28.9	17.9	20.6	9.4	18.4	6.7
60 bit × 512	116	78.7	82.3	37.5	73.8	27.1
124 bit	0.46	0.29	0.32	0.15	0.29	0.11
124 bit × 32	14.9	9.2	10.4	4.7	9.3	3.4
124 bit × 128	59.7	36.8	41.4	18.9	37.2	13.5
124 bit × 512	238	147	165	75.4	149	54.6
252 bit	0.95	0.59	0.65	0.30	0.57	0.22
252 bit × 32	30.2	18.6	20.8	9.5	18.9	6.8
252 bit × 128	121	74.3	83.5	37.8	76.1	27.4
252 bit × 512	484	297	358	165	302	109

Table 3: Comparison of non-optimised prover runtime in seconds of aggregate range proofs from [16] with this work. Verification times are only included for completeness. See Section 5 for details.

element and 2 scalars (and 1 challenge). Thus, the corresponding (almost) naive $\mathsf{QESA}_{\mathrm{Copy}}$ requires $\ell M + 1$ variables, and hence $n = \ell M + 4$.[44]

In total, the prover transmits $2\lceil \log(\ell M + 4)\rceil + 5$ group elements and 4 scalars. This is almost identical to [16], where $2\lceil \log(\ell M)\rceil + 4$ group elements and 5 scalars are transmitted. However, our approach is generic and not tailored to range proofs. Thus, the performance seems adequate.[45]

5 Implementation

We implemented all protocols in C++17 using the RELIC toolkit [4] for underlying group operations. Our instantiation uses $\mathbb{G} = \text{Curve25519}$ and thus $\mathbb{F}_p = \mathbb{F}_{2^{255}-19}$. For a fair comparison, we implemented Bulletproofs on the same architecture with equal care. The code is available on GitHub.[46]

Representing $\mathbf{\Gamma}$. All QESA protocols make use of sparse matrices $\mathbf{\Gamma}$. For efficient computation, a suitable representation is necessary. Decomposing $\mathbf{\Gamma}$ into a sum $\sum_i \boldsymbol{a}_i \boldsymbol{b}_i^\top$, similar to R1CS, allows for both runtime and memory optimisations. Note that vectors $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$ are sparse themselves, allowing for even further optimisation via an appropriate data structure. For multiplications $\mathbf{\Gamma}\boldsymbol{s}$, at most $m \sum_i k_i \ell_i$ scalar multiplications are necessary, where m, k_i, ℓ_i are the number of non-zero entries in $\boldsymbol{s}$, $\boldsymbol{a}_i$, $\boldsymbol{b}_i$. Thus, all operations remain polynomial in the input size.

Results. We benchmarked our protocols on an Intel Core i7-6600U CPU at 2.6GHz running Debian Stretch 4.9.168 using a single core. A point multiplication with a random 254-bit scalar takes on average 0.28ms on this platform. Table 3 shows how our aggregate range proofs $\mathsf{QESA}_{\mathrm{RP}}$ compare to Bulletproofs. For $\mathsf{QESA}_{\mathrm{RP}}$, the internal witness $\boldsymbol{w}$ contains 4 static elements: the constant 1, the aggregate element for $\mathsf{QESA}_{\mathrm{Copy}}$, and the 2 random elements added by $\mathsf{QESA}_{\mathrm{Inner}}$, c.f. Appendix G. Hence, we select the range as a power of 2 minus 4, in order to keep the CRS size from expanding to the next power of two. Our results show that $\mathsf{QESA}_{\mathrm{RP}}$ outperforms Bulletproofs for all tested parameters. Allowing batching randomnesses to be small further improves the performance (cf. $\mathsf{QESA}_{\mathrm{RP}}$ (short)

[44] We either eliminate bit b_0 or instead of copying we use the "implicit" copy $\sum 2^i b_i$. Otherwise, we would need $(\ell + 1)M + 1$ variables. With all optimisations of Remark 4.18, we could get down to ℓM variables, hence $n = \ell M + 3$.

[45] [16] instantiates arithmetic circuit proofs differently. While [16] can deal with commitments, these are only single-valued Pedersen commitments. It should be possible to extend [16] to our more general setting, but it is not obvious how hard it is

[46] `https://github.com/emsec/QESA_ZK`

Shuffle size	1000		10000		100000	
	$\mathcal{P}$	$\mathcal{V}$	$\mathcal{P}$	$\mathcal{V}$	$\mathcal{P}$	$\mathcal{V}$
Time [s]	8.8	4.4	117	56.1	1009	491

Table 4: Evaluation of shuffle proofs via $\mathsf{QESA}_{\text{Copy}}$ and $\mathsf{LMPA}_{\text{simpleZK}}$.

for 140-bit random values).[47] Note that the execution times given in [16] are lower, since a highly optimised library dedicated to a single elliptic curve was used instead of a general purpose library as in this work. However, since both protocols were benchmarked on the same platform with the same underlying library, the values in Table 3 give a fair comparison.

Note that we have not applied special optimisations to the verification algorithms and therefore show verification times in gray. Using *delayed (batch) verification*, e.g. as in [16], significantly improves verifier performance. Optimised *verification* performance of Bulletproofs and our proof systems is almost identical.[48] This was also verified in independent benchmarks by Noether and Shamir.[49] We are not aware of similar optimisations for the prover.

Table 4 gives execution times for our shuffle proofs. They are an instantiation of [5], c.f. Appendix C, and we project them to be 2–3× more computationally expensive than [5], but they are size $O(\log(N))$ instead of $O(\sqrt{N})$ for N ciphertexts. Again the very high execution times compared to [5] are caused by the underlying library.

Acknowledgements. We thank the anonymous reviewers of CRYPTO '19 and CCS '19 for helpful comments, which improved the overall quality and presentation of this work. Furthermore, we thank Ken Shamir for helpful editorial discussions. This work is supported by DFG grants PA 587/10-1 and RU 1664/3-1.

References

[1] Shashank Agrawal, Chaya Ganesh, and Payman Mohassel. "Non-Interactive Zero-Knowledge Proofs for Composite Statements". In: *CRYPTO 2018, Part III.* 2018.

[2] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. "Ligero: Lightweight Sublinear Arguments Without a Trusted Setup". In: *ACM CCS 17.* 2017.

[3] Oleg Andreev, Henry de Valence, and Cathie Yun. *dalek-cryptography bulletproofs.* `https://github.com/dalek-cryptography/bulletproofs`.

[4] D. F. Aranha and C. P. L. Gouvêa. *RELIC is an Efficient LIbrary for Cryptography.* `https://github.com/relic-toolkit/relic`.

[5] Stephanie Bayer and Jens Groth. "Efficient Zero-Knowledge Argument for Correctness of a Shuffle". In: *EUROCRYPT 2012.* 2012.

[6] Mihir Bellare and Oded Goldreich. "On Defining Proofs of Knowledge". In: *CRYPTO'92.* 1993.

[7] Mihir Bellare and Oded Goldreich. "On Probabilistic versus Deterministic Provers in the Definition of Proofs of Knowledge". In: *Studies in Complexity and Cryptography.* 2011.

[8] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. "Aurora: Transparent Succinct Arguments for R1CS". In: *IACR Cryptology ePrint Archive* (2018).

[9] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. "SNARKs for C: Verifying Program Executions Succinctly and in Zero Knowledge". In: *CRYPTO 2013, Part II.* 2013.

[10] David Bernhard, Olivier Pereira, and Bogdan Warinschi. "How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios". In: *ASIACRYPT 2012.* 2012.

[11] Nir Bitansky, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Huijia Lin, Aviad Rubinstein, and Eran Tromer. "The Hunting of the SNARK". In: *Journal of Cryptology* 4 (Oct. 2017).

[12] Jonathan Bootle and Jens Groth. "Efficient Batch Zero-Knowledge Arguments for Low Degree Polynomials". In: *PKC 2018, Part II.* 2018.

[47] To justify short exponents, concrete security estimates are needed. We are not aware of results justifying any concrete instantiations. If our conjectures in Appendix D hold, we can justify at least 80 bit security for witness size $n \leq 2^{16}$.

[48] Application of delayed batch verification with multi-exponentiation to our setting is slightly different. However, compared to the costs of the multi-exponentiation, the difference is likely not noticeable.

[49] Code available at `https://github.com/kenshamir/qesa/blob/nozk-verifier/src/ipa/no_zk.rs`

[13] 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 2016, Part II*. 2016.

[14] Jonathan Bootle, Andrea Cerulli, Essam Ghadafi, Jens Groth, Mohammad Hajiabadi, and Sune K. Jakobsen. "Linear-Time Zero-Knowledge Proofs for Arithmetic Circuit Satisfiability". In: *ASIACRYPT 2017, Part III*. 2017.

[15] Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. "Compressing Vector OLE". In: *ACM CCS 18*. 2018.

[16] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. "Bulletproofs: Short Proofs for Confidential Transactions and More". In: *2018 IEEE Symposium on Security and Privacy*. 2018.

[17] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. "Post-Quantum Zero-Knowledge and Signatures from Symmetric-Key Primitives". In: *ACM CCS 17*. 2017.

[18] Michele Ciampi, Giuseppe Persiano, Luisa Siniscalchi, and Ivan Visconti. "A Transform for NIZK Almost as Efficient and General as the Fiat-Shamir Transform Without Programmable Random Oracles". In: *TCC 2016-A, Part II*. 2016.

[19] Ronald Cramer and Ivan Damgård. "Zero-Knowledge Proofs for Finite Field Arithmetic; or: Can Zero-Knowledge Be for Free?" In: *CRYPTO'98*. 1998.

[20] Ivan Damgård. "Efficient Concurrent Zero-Knowledge in the Auxiliary String Model". In: *EUROCRYPT 2000*. 2000.

[21] George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. "Square Span Programs with Applications to Succinct NIZK Arguments". In: *ASIACRYPT 2014, Part I*. 2014.

[22] Alex Escala and Jens Groth. "Fine-Tuning Groth-Sahai Proofs". In: *PKC 2014*. 2014.

[23] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. "An Algebraic Framework for Diffie-Hellman Assumptions". In: *CRYPTO 2013, Part II*. 2013.

[24] Rosario Gennaro, Michele Minelli, Anca Nitulescu, and Michele Orrù. "Lattice-Based zk-SNARKs from Square Span Programs". In: *ACM CCS 18*. 2018.

[25] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. "Quadratic Span Programs and Succinct NIZKs without PCPs". In: *EUROCRYPT 2013*. 2013.

[26] Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. "ZKBoo: Faster Zero-Knowledge for Boolean Circuits". In: *USENIX Security Symposium*. 2016.

[27] Jens Groth. "Honest verifier zero-knowledge arguments applied". PhD thesis. Aarhus University, 2004.

[28] Jens Groth. "Linear Algebra with Sub-linear Zero-Knowledge Arguments". In: *CRYPTO 2009*. 2009.

[29] Jens Groth. "On the Size of Pairing-Based Non-interactive Arguments". In: *EUROCRYPT 2016, Part II*. 2016.

[30] Jens Groth. "Short Non-interactive Zero-Knowledge Proofs". In: *ASIACRYPT 2010*. 2010.

[31] Jens Groth and Yuval Ishai. "Sub-linear Zero-Knowledge Argument for Correctness of a Shuffle". In: *EUROCRYPT 2008*. 2008.

[32] Jens Groth and Amit Sahai. "Efficient Non-interactive Proof Systems for Bilinear Groups". In: *EUROCRYPT 2008*. 2008.

[33] Jens Groth et al. *Security Track Proceeding*. Tech. rep. `https://zkproof.org/documents.html`. ZKProof Standards, 2018.

[34] Ryan Henry and Ian Goldberg. "Batch Proofs of Partial Knowledge". In: *ACNS 13*. 2013.

[35] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. "Zero-knowledge from secure multiparty computation". In: *39th ACM STOC*. 2007.

[36] Ahmed E. Kosba, Zhichao Zhao, Andrew Miller, Yi Qian, T.-H. Hubert Chan, Charalampos Papamanthou, Rafael Pass, Abhi Shelat, and Elaine Shi. "$C\emptyset C\emptyset$: A Framework for Building Composable Zero-Knowledge Proofs". In: *IACR Cryptology ePrint Archive* (2015).

[37] Yehuda Lindell. "An Efficient Transform from Sigma Protocols to NIZK with a CRS and Non-programmable Random Oracle". In: *TCC 2015, Part I*. 2015.

[38] Yehuda Lindell. "Parallel Coin-Tossing and Constant-Round Secure Two-Party Computation". In: *Journal of Cryptology* 3 (June 2003).

[39] Helger Lipmaa. "Succinct Non-Interactive Zero Knowledge Arguments from Span Programs and Linear Error-Correcting Codes". In: *ASIACRYPT 2013, Part I*. 2013.

[40] Ueli Maurer. "Zero-knowledge proofs of knowledge for group homomorphisms". In: *Des. Codes Cryptography* 2-3 (2015).

[41] Paz Morillo, Carla Ràfols, and Jorge Luis Villar. "The Kernel Matrix Diffie-Hellman Assumption". In: *ASIACRYPT 2016, Part I*. 2016.

[42] C. Andrew Neff. "A Verifiable Secret Shuffle and Its Application to e-Voting". In: *ACM CCS 01*. 2001.

[43] Bryan Parno, Craig Gentry, Jon Howell, and Mariana Raykova. *Pinocchio: Nearly Practical Verifiable Computation*. Cryptology ePrint Archive, Report 2013/279. `http://eprint.iacr.org/2013/279`. 2013.

[44] Kun Peng, Colin Boyd, and Ed Dawson. "Batch zero-knowledge proof and verification and its applications". In: *ACM Trans. Inf. Syst. Secur.* 2 (2007).

[45] Björn Terelius and Douglas Wikström. "Proofs of Restricted Shuffles". In: *AFRICACRYPT 10*. 2010.

[46] Riad S. Wahby, Ioanna Tzialla, abhi shelat, Justin Thaler, and Michael Walfish. "Doubly-Efficient zkSNARKs Without Trusted Setup". In: *2018 IEEE Symposium on Security and Privacy*. 2018.

[47] *What is Jubjub?* `https://z.cash/technology/jubjub`.

[48] Douglas Wikström. "Special Soundness Revisited". In: *IACR Cryptology ePrint Archive* (2018). URL: `https://eprint.iacr.org/2018/1157`.

A Omissions

This section is only for appendix numbering compatibility with the extended version.

B Batch proofs of knowledge

By applying the "linear combination of protocols" technique, to multiple "trivial proofs of knowledge" (c.f. Fig. 2) we obtain batch verification of statements $([\boldsymbol{A}], [\boldsymbol{t}_i])$, $i = 1, \dots, N$, i.e. the setting of [44], in a straightforward way.

Protocol B.1. The following is a protocol to prove: $\exists \boldsymbol{w}_i \colon [\boldsymbol{A}]\boldsymbol{w}_i = [\boldsymbol{t}_i]$ for $i = 1, \dots, N$. Let χ_{N+1} be a testing distribution for challenges, such that $\boldsymbol{x} \leftarrow \chi_{N+1}$ has $x_{N+1} \neq 0$ always. Common input is $([\boldsymbol{A}], ([\boldsymbol{t}_i])_i) \in \mathbb{G}^{m \times n} \times \mathbb{G}^n$. The prover's witness are some $\boldsymbol{w}_i \in \mathbb{F}_p^n$.

- $\mathcal{P} \to \mathcal{V}$: Pick $\boldsymbol{r} \leftarrow \mathbb{F}_p^n$ and let $[\boldsymbol{a}] = [\boldsymbol{A}]\boldsymbol{r}$. Send $[\boldsymbol{a}] \in \mathbb{G}^m$.
- $\mathcal{V} \to \mathcal{P}$: Pick $\boldsymbol{x} \leftarrow \chi_{N+1}$. Send $\boldsymbol{x} \in \mathbb{F}_p$.
- $\mathcal{P} \to \mathcal{V}$: Compute $\boldsymbol{z} = \boldsymbol{x}^\top \begin{pmatrix} \boldsymbol{w}_1 \\ \vdots \\ \boldsymbol{w}_N \\ \boldsymbol{r} \end{pmatrix} = \sum_{i=1}^N x_i \boldsymbol{w}_i + x_{N+1}\boldsymbol{r}$. Send $\boldsymbol{z} \in \mathbb{F}_p^n$.
- $\mathcal{V}$: Check $[\boldsymbol{A}]\boldsymbol{z} \overset{?}{=} \sum_{i=1}^N x_i [\boldsymbol{t}_i] + x_{N+1}[\boldsymbol{a}]$, and accept/reject if true/false.

Lemma B.2. *Protocol B.1 is a* HVZK*-PoK for* $\exists w \colon [\boldsymbol{t}] = [\boldsymbol{A}]\boldsymbol{w}$. *It is perfectly complete, has perfect* HVZK *and is* $(N+1)$*-special sound.*

Proof. **Completeness** is straightforward. **Extraction** uses $N+1$ accepting transcripts $([\boldsymbol{a}], \boldsymbol{x}_j, \boldsymbol{z}_j)$. Let $[\boldsymbol{T}] \coloneqq [\boldsymbol{t}_1, \dots, \boldsymbol{t}_N, \boldsymbol{a}]$ and $\boldsymbol{Z}$, $\boldsymbol{X}$ be appropriate matrices built from the $N+1$ transcripts. Since $[\boldsymbol{A}]\boldsymbol{Z} = \boldsymbol{X}$, we find $(\boldsymbol{w}_1, \dots, \boldsymbol{w}_N, \boldsymbol{r}) \coloneqq \boldsymbol{Z}\boldsymbol{X}^{-1}$ is a valid witness. For **HVZK** note that $x_{N+1} \neq 0$. Hence $\boldsymbol{z}$ is uniformly distributed for any honest execution. Thus, we can pick $\boldsymbol{z} \leftarrow \mathbb{F}_p^m$ and let $[\boldsymbol{a}] \coloneqq [\boldsymbol{A}]\boldsymbol{z} - [\boldsymbol{T}]\boldsymbol{x}$ as usual. □

Using vectors of vectors and matrices of matrices, we can write the above as

$$\boldsymbol{x}^\top \otimes \mathrm{id} \begin{bmatrix} \boldsymbol{A} & & \\ & \ddots & \\ & & \boldsymbol{A} \end{bmatrix} \begin{bmatrix} \boldsymbol{w}_1 \\ \vdots \\ \boldsymbol{r} \end{bmatrix} = [\boldsymbol{A}] \begin{bmatrix} \boldsymbol{w}_1 \\ \vdots \\ \boldsymbol{r} \end{bmatrix}^\top \boldsymbol{x} = [\boldsymbol{A}](\sum_{i=1}^N x_i \boldsymbol{w}_i + x_{N+1}\boldsymbol{r})$$

$$= \sum_{i=1}^N x_i [\boldsymbol{t}_i] + x_{N+1}[\boldsymbol{a}] = \boldsymbol{x}^\top \begin{bmatrix} \boldsymbol{t}_1 \\ \vdots \\ \boldsymbol{a} \end{bmatrix} = \begin{bmatrix} \boldsymbol{t}_1 \\ \vdots \\ \boldsymbol{a} \end{bmatrix}^\top \boldsymbol{x}$$

In a sense, we run $\mathsf{LMPA}_{\mathrm{batch}}$, but exploit the structure (namely block-diagonality) to "commute" $\boldsymbol{x}$ and $\mathrm{diag}([\boldsymbol{A}], \dots, [\boldsymbol{A}])$. Linear combination also yields efficient k-out-of-N proofs, by having the verifier only partially fix the challenge. However, this must be done carefully or it is unsound, see [34].

C An efficient proof of correctness of a shuffle

A proof of correctness of a shuffle is a proof that two (multi-)sets of ciphertexts decrypt to the same multi-set of plaintexts. This is especially interesting in settings with rerandomisable ciphertexts, because the "shuffling party" does not need to decrypt. For electronic voting, a shuffle achieves a certain unlinkability between the originally encrypted votes, and the (in a final step) decrypted votes, while the proofs of correctness of the shuffle ensure that the voting result is unaffected.

With our tools, it is possible to prove the correctness of a shuffle in logarithmic communication for ElGamal ciphertexts in a very naive manner. Namely, we commit to a permutation matrix (as part of $\boldsymbol{w}$) and rerandomisation randomness for the ElGamal ciphertexts (also part of $\boldsymbol{w}$). Then we prove that $[\boldsymbol{A}]\boldsymbol{w} = [\vec{\boldsymbol{c}}]$, where $[\boldsymbol{A}]$ is constructed from the old ciphertexts and the ElGamal public key, and $[\vec{\boldsymbol{c}}]$ is the vector of shuffled ciphertexts. We also add a proof that (the relevant part of) $\boldsymbol{w}$ commits to a permutation matrix, as sketched in Section 3.5. This all neatly fits into our framework, giving a logarithmic size proof overall. However, there is a huge drawback: The size of the permutation matrix, hence $\boldsymbol{w}$, is N^2 for N ciphertexts. Thus, the *computation* grows quadratically in N. This is unacceptable in practice.

Remark C.1. Shuffle arguments for (Pedersen) committed values were already constructed in [16], by using a sorting circuit and comparing the sorted sequences. In [3], an improved shuffle argument (for commitments) is constructed. It relies on the techniques in [5, 42]. More generally, [3] allows *randomized R1CS*, by using a commit-and-prove structure of Bulletproofs.

In Appendix C.1 below, we rely essentially on the same techniques and properties as [3]. However, our setting concerns (ElGamal) *encrypted* values (e.g. votes), not (Pedersen) committed values. While it is likely that Bulletproofs can be suitably modified to cover this setting as well, it is not immediately obvious how.

Lastly, we note that given any log-communication proof system, theoretically "efficient" (polynomial time) and short shuffle proofs are essentially a triviality. Just prove the relevant statements (e.g. ElGamal randomisation) in a non-black-box manner. Since we have $O(\log(\mathsf{poly}(\kappa, n))) = O(\log(\kappa) + \log(n))$, communication is almost logarithmic in n (and if $n \in \Omega(\kappa)$, it is logarithmic).

C.1 Adapting the shuffle argument of Bayer–Groth

The shuffle argument of Bayer and Groth [5] is built from two sub-arguments, a "product argument" and a "multi-exponentiation argument". A generic proof of security is given in [5, Theorem 5]. The former argument can be instantiated by $\mathsf{QESA}_{\mathrm{ZK}}$, or more precisely, $\mathsf{QESA}_{\mathrm{Copy}}$. The latter argument can be instantiated by $\mathsf{LMPA}_{\mathrm{ZK}}$. Since our arguments have logarithmic communication and need linearly many exponentiations, so does the resulting shuffle argument. We give a more detailed instantiation below.

- CRS: $\mathrm{ck} = (\mathrm{ck}_Q, \mathrm{ck}_L)$, where $\mathrm{ck}_Q = ([\boldsymbol{g}', \boldsymbol{g}'', Q])$ is the commitment key for $\mathsf{QESA}_{\mathrm{ZK}}$ and $\mathrm{ck}_L = [\boldsymbol{h}]$ is the commitment key for $\mathsf{LMPA}_{\mathrm{ZK}}$ (or empty if a simple zero-knowledge LMPA version is used). Here $[\boldsymbol{g}'] \in \mathbb{F}_p^n$, where $n \geq N + 2$ is a (suitably large) power of 2. Note that our commitment keys consist of random group elements.
- Common input: Old and new ciphertexts $[\mathsf{ct}_i^{\text{old}}], [\mathsf{ct}_i^{\text{new}}] \in \mathbb{G}^2$ for $i = \{0, \dots, N-1\}$ and ElGamal public key $[\mathrm{pk}] \in \mathbb{G}^2$.
- Prover's witness: The random permutation $\boldsymbol{\pi} \in \{0, \dots, N-1\}^N$ and rerandomisation randomnesses $\rho_i \in \mathbb{F}_p$ such that $[\mathsf{ct}_i^{\text{new}}] = [\mathsf{ct}_{\pi_i}^{\text{old}}] + \rho_i[\mathrm{pk}]$. (Note that $\mathsf{Enc}(0; \rho_i) = \rho_i[\mathrm{pk}]$ for ElGamal.)
- $\mathscr{P} \to \mathscr{V}$: Compute and send the commitment $[c_{\boldsymbol{\pi}}]$ to π:

$$\begin{aligned}[c_{\boldsymbol{\pi}}] &= \mathsf{Com}_{\mathrm{ck}_Q}(\boldsymbol{\pi}; 0, r_{\boldsymbol{\pi}}) \\ &= [g'_1 \mid g'_2, \dots, g'_{N+1} \mid g'_{N+2}, \dots, g'_{n-2} \mid g'_{n-1}, g'_n] \begin{pmatrix} 0 \\ \hline \boldsymbol{\pi} \\ \hline \boldsymbol{0} \\ \hline 0 \\ r_{\boldsymbol{\pi}} \end{pmatrix}\end{aligned}$$

  (Remember that $[\boldsymbol{g}'_{n-1}]$ and $[\boldsymbol{g}'_n]$ are reserved for randomness in $\mathsf{QESA}_{\mathrm{ZK}}$ commitments, and $[\boldsymbol{g}_1]$ is also reserved (for the constant 1).)
- $\mathscr{V} \to \mathscr{P}$: Send $\boldsymbol{x} = (x_0, \dots, x_{N-1}) \leftarrow \chi_N$.
- $\mathscr{P} \to \mathscr{V}$: Send $[c_{\boldsymbol{y}}] = \mathsf{Com}_{\mathrm{ck}_Q}(\boldsymbol{y}; 0, r_{\boldsymbol{y}})$, where $[\boldsymbol{y}] \coloneqq \boldsymbol{\pi}(\boldsymbol{x}) = (\boldsymbol{x}_{\pi_i})_i = (x_{\pi_0}, \dots, x_{\pi_{N-1}})$.
- $\mathscr{V} \to \mathscr{P}$: Send $\zeta, z \leftarrow \mathbb{F}_p$.
- $\mathscr{P} \leftrightarrow \mathscr{V}$: Prove following statements using (logarithmic communication) sub-protocols $\mathsf{QESA}_{\mathrm{Copy}}$ and $\mathsf{LMPA}_{\mathrm{ZK}}$:
  - **$[c_{\boldsymbol{\pi}}]$ is a permutation and $[c_{\boldsymbol{y}}]$ is a commitment to $\boldsymbol{\pi}(\boldsymbol{x})$:** The prover shows (in zero-knowledge) that

    $$\prod_{i=0}^{N-1} (\zeta \pi_i + y_i - z) = \prod_{i=0}^{N-1} (\zeta i + x_i - z).$$

    Note that $\zeta[c_{\boldsymbol{\pi}}] + [c_{\boldsymbol{y}}]$ is a commitment to $\zeta\boldsymbol{\pi} + \boldsymbol{y}$, which can be used for $\mathsf{QESA}_{\mathrm{ZK}}$, or more precisely, $\mathsf{QESA}_{\mathrm{Copy}}$. Also note that the right-hand side is computable from public information.

- $[\vec{\mathsf{ct}}^{\text{new}}]$ **is a rerandomised permutation of** $[\vec{\mathsf{ct}}^{\text{old}}]$**:** The prover shows (in zero-knowledge) that
$$\sum_i [\mathsf{ct}_i^{\text{old}}] y_i + [\text{pk}] \sum_i \rho_i x_i = \sum_i [\mathsf{ct}_i^{\text{new}}] x_i.$$
This fits into our matrix multiplication proofs (with witness $\left(\begin{smallmatrix} \boldsymbol{y} \\ \boldsymbol{x}^\top \rho \end{smallmatrix}\right) \in \mathbb{F}_p^{N+1}$). Concretely, the prover commits to $\sigma := \boldsymbol{x}^\top \rho$ via $[c_\sigma] = \mathsf{Com}_{\text{ck}_Q}((\begin{smallmatrix} \boldsymbol{0} \\ \sigma \end{smallmatrix}); r_\sigma, 0) = [g'_{N+2}, g'_{n-1}](\begin{smallmatrix} \sigma \\ r_\sigma \end{smallmatrix})$ for $r_\sigma \leftarrow \mathbb{F}_p$. He sends c_σ to the verifier and engages in a $\mathsf{LMPA}_{\text{ZK}}$ protocol for
$$\left[\begin{array}{c|c|cc} g'_2, \dots, g'_{N+1} & g'_{N+2} & g'_{n-1} & g'_n \\ g'_2, \dots, g'_{N+1} & 0 & 0 & g'_n \\ \boldsymbol{0} & g'_{N+2} & g'_{n-1} & 0 \\ \mathsf{ct}_0^{\text{old}} \dots \mathsf{ct}_{N-1}^{\text{old}} & \text{pk} & 0 & 0 \end{array}\right] \begin{pmatrix} \boldsymbol{y} \\ \hline \sigma \\ \hline r_\sigma \\ r_{\boldsymbol{y}} \end{pmatrix} = \begin{bmatrix} c_{\boldsymbol{y}} + c_\sigma \\ c_{\boldsymbol{y}} \\ c_\sigma \\ \boldsymbol{u} \end{bmatrix}$$
where $[\boldsymbol{u}] := \sum x_i [\mathsf{ct}_i^{\text{new}}]$. The top row is added so one can run $\mathsf{LMPA}_{\text{batch}}$, reducing to a $2 \times n$ matrix. Since $[\boldsymbol{g}']$ has hard kernel relation, so has $[\boldsymbol{A}]$. (This is a "commitment-extension", see Remark 3.6.) Also note that this LMPA proof ensures the requirements of $\mathsf{QESA}_{\text{Copy}}$ on the opening of $[c_{\boldsymbol{y}}]$, hence no additional subprotocol $\mathcal{S}$ is necessary in this instance.

Honest verifier zero-knowledge of this protocol follows from honest verifier zero-knowledge of the subprotocols. Soundness (and extraction) follows from soundness (and extraction) of the subprotocols. Namely, for fixed $\boldsymbol{\pi}$, randomly chosen $\boldsymbol{x}$ and arbitrary $\boldsymbol{y}$, the probability that $\prod_{i=0}^{N-1}(\zeta\pi_i + y_i - z) = \prod_{i=0}^{N-1}(\zeta i + x_i - z)$ holds for $\zeta, z \leftarrow \mathbb{F}_p$ if $y_i \neq x_{\pi(i)}$ is negligible thanks to the Schwartz–Zippel lemma.[50]

In [5], intuition and a detailed security argument is given. Despite our minor modifications, their proof adapts seamlessly to our setting. The idea of using (permutation invariant sets of) roots of polynomials to prove that one set of roots is a permutation of another goes back to [42] and was extended to restricted permutations in [45].

A rough efficiency estimate of our scheme is $30N$ exponentiations for the prover and $10N$ exponentiations for the verifier. These are roughly twice the numbers of [5], when trading interaction for efficiency. However [5] has $O(\sqrt{N})$ size proofs, while we have $O(\log(N))$ size proofs.

D Witness-extended emulation and $\mathsf{TreeFind}$

D.1 Witness-extended emulation

We define (black-box) witness-extended emulation following [13, 31, 38]. But we separate extraction and emulation, and allow emulation (or rather extraction) to fail with probability depending on the extraction error. This is somewhat inconvenient, redundant, and yields yet another definition of "of knowledge". Fortunately, the extraction (i.e. the "knowledge" parts) are equivalent to standard formulations. However, combined with *emulation*, the prior work we are aware of was only concerned with negligible extraction errors.

Definition D.1 (Witness-extended emulation). Let $(\mathcal{P}, \mathcal{V})$ by an interactive argument system for $\mathcal{R}$. We say that $(\mathcal{P}, \mathcal{V})$ is an **argument of knowledge** with **witness-extended emulation** and **extraction error** δ_{ext} if there exists a (universal) *expected* polynomial-time emulator Emu. The emulator takes as input the CRS, a statement st and a rewindable deterministic[51] proof-oracle $\mathcal{P}^*(state)$, written $\mathsf{Emu}(\text{crs}, st, \mathcal{P}^*(state))$. (As usual, we suppress crs in the following.) It outputs a pair (tr, w) of emulated transcript and purported witness. We require following properties. For every adversary given by a pair of efficient algorithms $(\mathcal{A}, \mathcal{P}^*)$ we have:

[50] In more detail: The degree of the (difference) polynomial in z is at most N. The two polynomials are equal if and only if they have the same roots (with multiplicity). So the sets $\{\zeta\pi_i + y_i\}_i$ and $\{\zeta i + x_i\}_i$ must be equal. The probability that $\zeta i + y = \zeta j + x$ if $i \neq j$ is negligible (for any fixed choice of x, y). Hence if the sets are equal, with overwhelming probability we find that the sets $\{(\pi_i, y_i)\}_i$ and $\{(j, x_j)\}_i$ are equal. In other words, π is a permutation of the roots. With probability $1 - \delta_{\mathsf{snd}}(\chi_N)$ all x_j are distinct, see Remark E.6. Hence π is a permutation of $\{0, \dots, N-1\}$.

[51] We refer to [7] for a comparison of deterministic and probabilistic proof-oracles.

- **(Computational) Emulation**:

$$\mathbb{P}\left(\begin{array}{l}\mathrm{crs} \leftarrow \mathsf{GenCRS}(1^\kappa); (st, state) \leftarrow \mathcal{A}(\mathrm{crs}); \\ tr \leftarrow \langle \mathcal{P}^*(state), \mathcal{V}(st)\rangle \colon \mathcal{A}(state, tr) \overset{?}{=} 1\end{array}\right)$$
$$\overset{c}{\approx} \mathbb{P}\left(\begin{array}{l}\mathrm{crs} \leftarrow \mathsf{GenCRS}(1^\kappa); (st, state) \leftarrow \mathcal{A}(\mathrm{crs}); \\ (tr, w) \leftarrow \mathsf{Emu}(st, \mathcal{P}^*(state)) \colon \mathcal{A}(state, tr) \overset{?}{=} 1\end{array}\right)$$

- **Extraction**: For all crs $\leftarrow \mathsf{GenCRS}(1^\kappa)$ and all $(st, state) \leftarrow \mathcal{A}(\mathrm{crs})$ we have

$$\mathbb{P}\left(\mathsf{Emu}(st, \mathcal{P}^*(state)) \text{ fails}\right) \leq \frac{\delta_{\mathrm{ext}}}{\mathbb{P}(\underbrace{\langle \mathcal{P}^*(state), \mathcal{V}(st)\rangle = 1}_{\mathcal{P}^* \text{ succeeds}})} \tag{D.1}$$

By “Emu fails”, we mean that the extracted witness w does not satisfy $(st, w) \in \mathcal{R}$. An equivalent formulation of the inquality is

$$\mathbb{P}\left(\mathsf{Emu}(state, \mathcal{P}^*(state)) \text{ fails } \mid \langle \mathcal{P}^*(state), \mathcal{V}(st)\rangle = 1\right) \leq \delta_{\mathrm{ext}}.$$

where we assume $\mathcal{V}$ uses independent randomness.

If δ_{ext} is negligible, then the definition is very similar to [13, 31, 38]. However, we can deal with non-negligible δ_{ext}, e.g. fixed to 2^{-80} independent of the security parameter, essentially having the (partial) definition of an extractor (not emulator) due to Eq. (D.1). In any case, we require the emulated transcript to be (computationally) indistinguishable from a real transcript.

Typically, the emulator proceeds in two steps, c.f. [38]. First, it uses the honest verifier's strategy to obtain a transcript. If this is not accepting, we're done. Otherwise, the emulator runs an extractor to obtain the witness (with suitable probability). In such a two-phase setting, one can amplify the probability to obtain a witness, e.g. by retrying the extraction step often enough.[52] For example, if we have acceptance probability $\varepsilon \geq \delta_{\mathrm{ext}}(1 + \frac{1}{\mathsf{poly}})$, then by $O(\kappa \cdot \mathsf{poly}(\kappa))$-fold repetition, we achieve overwhelming extraction probability. Thus, switching to a (sufficiently large) upper bound $\delta'_{\mathrm{ext}} \geq \delta_{\mathrm{ext}}$, we can obtain overwhelming extraction for this (weaker) extraction error.

We choose not to require such amplification for two reasons: First, a “one-shot” extractor with extraction error δ_{ext} is quite natural in our applications. Second, one might arguably want to call the “minimal” δ_{ext} *the* extraction error. Since amplification requires $\delta'_{\mathrm{ext}} > \delta_{\mathrm{ext}}$, we can only approximate δ_{ext} for such extractors, see [48] for similar considerations.

The focus of this paper is not the definition of witness-extended emulation with extraction error. Hence we stop the discussion of possible variations here.

D.1.1 Relating knowledge errors

Our definition of extraction error should be viewed as “per extractor”, not “per protocol” (see [48] for a similar point of view). Moreover, we chose a definition of extraction error which implies soundness, unlike Bellare and Goldreich [6], where soundness and extraction are explicitly separated. We ignore these differences here.

The (alternative) definition of knowledge error [6, Section 6] fits nicely into our setting. Indeed, from $\mathbb{P}(\mathsf{Ext}\text{ succeeds}) \geq 1 - \frac{\delta_{\mathrm{ext}}}{\varepsilon}$ we see that by first generating a (random) transcript, and only invoking Ext if $\mathcal{P}^*$ was successful, we get an extractor Ext' with

$$\mathbb{P}(\mathsf{Ext}' \text{ succeeds}) \geq \varepsilon(1 - \frac{\delta_{\mathrm{ext}}}{\varepsilon}) = \varepsilon - \delta_{\mathrm{ext}}$$

and expected polynomial runtime. Here, we considered an *extractor* sastifying Eq. (D.1), not an emulator. Thus, our notion implies that of [6]. (As noted before, witness-extended emulators can be constructed as Ext'.)

[52] Doing so by retrying the whole emulation must be done with care. Simply taking the “first successful run” can skew the distribution of the emulated transcript.

The converse regarding extraction is also evident from this inequality. Emulation *almost* follows from [38]. The construction and proof in [38] is only for negligible extraction error (and only guarantees negligible soundness error).[53] However, by choosing a weaker extraction error $\delta'_{\mathrm{ext}} \geq \alpha\delta_{\mathrm{ext}}$ with a sufficient gap $\alpha > 1$, the proof strategy should generalise.[54] Here, α should be constant or at least of the form $\alpha = 1 + \frac{1}{\mathsf{poly}}$. Due to runtime requirements, it is not obvious whether the (stronger) extraction error δ_{ext} can be preserved by some (improved) construction.

Again, we stop the discussion here, since this is not the focus of this work.

D.1.2 Lower bounds for black-box extraction

We pose following natural question.

Question D.2. Let $\mathcal{R}$ be a witness relation and Π_{Arg} be an argument system for $\mathcal{R}$. Suppose $\mathcal{R}$ has "short" witnesses of size n. In particular, $\mathcal{R}$ defines a hard language. Suppose a transcript of Π_{Arg} has size $s_{\mathcal{P}} + s_{\mathcal{V}}$. Does a black-box extractor (or emulator) need $n/s_{\mathcal{P}}$ transcripts for extraction? (Here $s_{\mathcal{P}}$ is the size of the total communication sent by $\mathcal{P}$, and likewise $s_{\mathcal{V}}$.)

If above assumption is true, it gives a lower-bound on the number of necessary rewinds of any efficient emulator. Indeed, it guarantees that small communication implies large black-box extraction overhead.

D.2 Modular extraction from μ-special soundness

Witness-extended emulation for μ-special sound protocols can be constructed as a two-stage process: First, run the protocol and keep the transcript (for emulation). If it is a successful transcript, find a good μ-tree. Second, apply the extractor from Definition 2.17 to obtain a witness. To find a good μ-tree with acceptable runtime, the straightforward "follow your nose" approach actually works, c.f. the general forking lemma in [13]. An alternative, with better guarantees but worse runtime estimates, is given in [48].

Given such a $\mathsf{TreeFind}$, we get the following:

Lemma D.3. *Let* $\Pi_{\mathrm{Arg}} = (\mathsf{GenCRS}, \mathcal{P}, \mathcal{V})$ *be a public coin interactive argument system with* μ-*special soundness and extractor* Ext. *Let* $\mu = (\mu_1, \ldots, \mu_\ell)$ *and* $\delta_i = \delta_{\mathsf{snd}}(\chi^{(i)})$ *the soundness error of the i-th testing distribution. Suppose* $\mathsf{TreeFind}$ *is a tree-finding algorithm with expected runtime* $t_{\mathsf{TreeFind}}/\varepsilon \in O(\mathsf{poly}(\kappa))/\varepsilon$, *where* ε *is the probability the oracle* $\mathcal{P}^*(state)$ *convinces the honest verifier. Let* η *be the probability that* $\mathsf{TreeFind}$ *outputs a* bad μ-*tree. Then* Π_{Arg} *has a witness-extended emulator* Emu *(as described above) with expected runtime roughly* $O(t_{\mathcal{V}} + t_{\mathsf{TreeFind}} + t_{\mathsf{Ext}})$ *and extraction error* η.

(For more precise runtime estimates, $\mathsf{TreeFind}$ should be modelled as a transcript oracle for Ext. Otherwise, short-circuit extraction is not useful. We chose to simplify here.)

The $\mathsf{TreeFind}$ algorithm of [13] outputs the first μ-tree it finds, if it is good. It generates trees recursively, depth-first, always branching paths with accepting transcripts, and aborts if it rewinds more than $1/\alpha$ times (for suitable negligible α). By a union bound, the probability that such a tree is bad is at most $\eta = \sum_{i=1}^{n} \frac{\delta_i}{\alpha} \prod_{j=1}^{i-1} k_j$. Here $\delta_i = \sigma_\infty(\chi^{(i)})$. Unfortunately, negligible α enforces negligible soundness errors for all i. Thus one cannot fix the soundness level of the testing distributions to, say 2^{-100}.

The algorithm in [48] can be configured to do better, e.g. to attain $\eta = \sum_{i=1}^{n} \delta_i \nu^i$ for any choice of $\nu > 1$. The choice $\nu = 2$ is quite natural, but yields relatively large bounds on runtime.

We have following question/conjecture which we also leave for future work:

[53] The proof in [38] is in the plain model, but it translates to the CRS model. Recall that our notion of extraction (error) is "unconditional". However, we consider statements such as "either $st \in \mathcal{L}$ *or a hard problem was solved*" and thus still incorporate hardness assumptions. This follows [13] and is convenient to use.

[54] Namely, we estimate the success probability ε of the prover precise enough (depending on α). Except with probability $2^{-\kappa}$, the approximation ε' is so close that $\frac{\varepsilon'}{\delta'_{\mathrm{ext}}}$ is bounded by constants small enough w.r.t. α. Thus $\frac{1}{\varepsilon' - \delta_{\mathrm{ext}}}$ is also (polynomially) bounded. (Note the use of δ'_{ext} and δ_{ext} here.)

Question D.4. Consider μ-special soundness with $\mu = (\mu_1, \dots, \mu_\ell)$. Suppose the respective testing distributions have soundness errors $\delta_i = \delta_{\mathsf{snd}}(\chi^{(i)})$. Hence, we expect[55] the "overall extraction error" δ_{snd} to be bounded by $\sum_i \delta_{\mathsf{snd}}^{(i)}$. Does there exist an efficient $\mathsf{TreeFind}$ algorithm such that

- $\mathsf{TreeFind}$ has runtime roughly $\widetilde{O}(n/\varepsilon)$ where $n = \prod_i \mu_i$ and ε is the probability for the verifier to accept. (By $\widetilde{O}(f)$, we denote asymptotic behaviour up to polylogarithmic factors.)
- $\mathsf{TreeFind}$ returns a *good* tree with probability at least $1 - \delta_{\mathsf{snd}}/\varepsilon$.

If the above is not satisfiable, how close can we get?

The algorithms from [13, 48] do not achieve this. The $\mathsf{TreeFinder}$ in [13] has weak soundness guarantees, but satisfies the runtime of first point. As noted above, [48] achieves the second point arbitrarily close by sacrificing runtime. Moreover, we note that [48] generalises testing distributions by working with matroids. Improving or strengthening results [48] could settle some of our questions and conjectures.

Remark D.5. We note that [48] relies on $\sigma_\infty(\chi)$ instead of $\delta_{\mathsf{snd}}(\chi)$. We hope that both measures are identical, see Conjecture E.1. We also note that our definition(s) of extraction error differs slightly from those in [48].

E Testing distributions

In this section, we state some simple but helpful insights on testing distributions. We note that linear independence can be generalised and used instead. For example, [48] uses a generalised setting.

E.1 Conjectures and computational soundness errors

We first conjecturally characterise the soundness error by a different measure, namely we define

$$\sigma_\infty(\chi_m) \coloneqq \max_{H \leq \mathbb{F}_p^n} \mathbb{P}(\boldsymbol{x} \leftarrow \chi_m \colon \boldsymbol{x} \in H)$$

where H ranges over all $(n-1)$-dimensional subspaces. We have following lemma.

Conjecture E.1. *Let χ be a testing distribution on $\mathbb{F}_p^m$. Then*

$$\delta_{\mathsf{snd}}(\chi_m) = \sigma_\infty(\chi_m).$$

Partial proof. The subdistribution ψ_H over the maximising H always yields n linearly dependent vectors (i.e. determinant 0), Moreover, ψ_H has weight $\varepsilon = \chi_m(H) = \sigma(\chi_m)$. By definition of δ_{snd}, we find $1 \leq \frac{1}{\varepsilon}\delta_{\mathsf{snd}}(\chi_m)$. Therefore $\delta_{\mathsf{snd}}(\chi_m) \leq \sigma_\infty(\chi_m)$.

To prove the claim, we need to show that $\sigma_\infty(\chi_m)$ is admissible as a soundness error, i.e.

$$\mathbb{P}(\boldsymbol{x}_i \leftarrow \psi \colon \det(\boldsymbol{x}_1, \dots, \boldsymbol{x}_m) = 0) \leq \frac{1}{\varepsilon}\sigma_\infty(\chi_m)$$

for all subdistributions ψ (with weight ε).

Note that the lefthand side is $\mathbb{P}(\boldsymbol{X} \leftarrow \psi^m \colon \boldsymbol{X} \in T)$ where $T \coloneqq \{\boldsymbol{X} \in \mathbb{F}_p^{m \times m} \mid \det(\boldsymbol{X}) = 0\}$. Equivalently $T = \cup_H H^m$ where H ranges over all hyperplanes. For convenience, we write $\psi(M) \coloneqq \mathbb{P}(M \in \psi)$. Thus, we find

$$\begin{aligned}
\mathbb{P}(\boldsymbol{X} \leftarrow \psi^m \colon \boldsymbol{X} \in T) &= \psi^m(\cup_H H^m) \\
&= \sum_{H_1} \psi^m(H_1^m) - \frac{1}{2!} \sum_{H_1 \neq H_2} \psi^m(H_1^m \cap H_2^m) + \dots \\
&= \sum_{H_1} \psi(H_1)^m - \frac{1}{2!} \sum_{H_1 \neq H_2} \psi(H_1 \cap H_2)^m + \dots
\end{aligned}$$

[55] We have not formally defined an extraction error for sequences of testing distributions. Indeed, a definition is non-trivial. We conjecture for some natural generalisation of testing distribution (covering such sequences) natural results hold, e.g. the sum of the soundness errors bounds the soundness error of the sequential composition of (generalised) testing distributions. The results in [48] support this (and might even partially prove it).

by application of the inclusion-exclusion principle, probabilistic independence (for $\psi^m(H^m) = \psi(H)^m$) and subspace properties (i.e. $H_1^m \cap H_2^m = (H_1 \cap H_2)^m$). Here all H_i range over all hyperplanes.

Heuristically, the higher order term should only decrease the sum. Moreover, a sum $\sum x_i^m$ under constraints $\sum x_i = 1$, $x_i \in [0, \sigma_\infty(\chi)]$ is maximised by maximising all x_i (i.e. to $\sigma_\infty(\chi)$, or whatever is "left" for the last one). Thus, heuristically, we have an upper bound $N\sigma_\infty(\chi)$ where $N = \frac{1}{\varepsilon}$ (the "number" of x_i's). This is exactly our claim.

However, the heuristic oversimplifies possible interdependencies of higher order terms (i.e. hyperplanes sharing lower-dimensional subspaces). As is, this is *not a proof.* $\sqsubset$

The above conjectured characterisation of the soundness error allows to prove and argue much easier. Most of the following results are stated w.r.t. to $\sigma_\infty(\chi)$.

The impact of the (or a) "favoured hyperplane", that is a hyperplane H with $\mathbb{P}(H \in \chi) = \sigma_\infty(\chi)$, is evident in following example.

Example E.2. Fix some hyperplane $H \leq \mathbb{F}_p^m$. Consider the distribution χ over $\mathbb{F}_p^m$ induced by following algorithm: Pick $\boldsymbol{x}_0 \leftarrow \mathbb{F}_p^m$ and $\boldsymbol{x}_1 \leftarrow H$ uniformly at random. Pick $b \leftarrow \{0, 1\}$ uniformly at random. Output $\boldsymbol{x}_b$.

This has following characteristics: With probability at most $2^{-m} + 3m^2p^{-1}$ a sample of m elements is linearly dependent.[56] But the soundness error, or rather σ_∞, is (slightly greater than) $\frac{1}{2}$. This is because the subdensity ψ_H, which is χ conditioned on H, has weight (slightly greater than) $\frac{1}{2}$ and m samples are always linearly dependent.

Remark E.3. If the halfplane H in Example E.2 is chosen uniformly at random and *secret*, and m *grows* in κ fast enough, then it is probably a hard problem to differentiate the distribution from a uniformly random one. See [15], where a similar (even more structured) assumption is used constructively.

Note that for constant m, one can give a (very inefficient) algorithm which recovers H given enough (e.g. $2m$) samples. Namely, try every subset of $m - 1$ indices, compute a candidate H', and check if about m samples $\boldsymbol{x}_i$ lie in H. This recovers H with high probability, thus distinguishing the distribution from random. (The effort to try all subsets is exponential im m, which by assumption is constant. Thus the overall algorithm is still polynomial in κ.)

The definition of soundness error $\delta_{\mathsf{snd}}(\chi)$ of χ is a "perfect unconditional" notion. It assigns to the distribution in Example E.2 a soundness error which is greater than $\frac{1}{2}$, even when $m = \mathsf{poly}(\kappa)$ grows with with the security parameter κ, and the distribution is assumed to be pseudorandom.

This motivates a relaxation of the soundness error. There are different ways to define a(n admissible) computational soundness.[57] The cleanest one is by comparison to a (unconditionally) secure distribution, similar to computational entropy. Namely, we say $\delta_{\mathsf{snd}}^{\mathsf{comp}}(\chi)$ is a(n admissible) computational soundness error if there exists a distribution χ' such that $\chi \overset{c}{\approx} \chi'$ and $\delta_{\mathsf{snd}}(\chi') = \delta_{\mathsf{snd}}^{\mathsf{comp}}$. (Recall that whenever we say "distribution" we actually mean *probability ensemble* or *family of distributions* (paramterised over κ).)

While this definition is elegant and resembles pseudoentropy, it has limited use: We would like to replace uniformly random samples by a PRG and *give away the seed.* Replacing uniform randomness with a PRG works nicely and yields a computational soundness error which is identical to the statistical one, according to the previous definition. However, giving away the seed makes no sense in that model. There is no indistinguishability involved.

Reminder. The soundness error is a combinatorial property. There is no need for pseudorandomness, as testing with powers of x shows. However, since we do not know a simple example of a distribution with (small) exponents $x_i \in \mathcal{S}$ for general $\mathcal{S} \subseteq \mathbb{F}_p$, it is natural to turn to PRG's. It is also a

[56] Split $\mathbb{P}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ lin. dep.) depending on the number k of picks with $b = 1$. For fixed k and $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m) \leftarrow H^k \times (\mathbb{F}_p^m)^{m-k}$ (which is enough due to symmetry), we find $\mathbb{P}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ lin. dep.) $\leq \mathbb{P}(\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k \in H\} \cup \{\boldsymbol{x}_{k+1}, \ldots, \boldsymbol{x}_m \in \mathbb{F}_p^m$ lin. dep.$\} \cup \{\exists k+1 \leq i \leq m \colon \boldsymbol{x}_i \in H\})$ which is easily bounded above by $3m^2p^{-1}$ for $k < m$. For $k = m$, linear dependence is guaranteed, but this only happens with probability 2^{-m}. Thus, $2^{-m} + \sum_{k=0}^{m-1} \binom{m}{k} 2^{-m} 3m^2/p \leq 2^{-m} + 3m^2/p$ is the desired bound.

[57] We speak of *admissible*, because there may be different computational soundness errors which are satisfied, depending on the choice of negligible functions. We do not know whether there is a (uniquely) well-defined minimal one. (Unlike *statistical* soundness, where a unique (minimal) error is given essentially by definition, and where Conjecture E.1 would imply very simple characterisation of it.)

plausible assumption, that non-pathological PRG's have a (statistical) soundness error close to the uniform choice. Otherwise, assuming Conjecture E.1, the PRG would have to have hidden favoured hyperplanes.

To define computational soundness which can encompass the setting where a PRG seed is sent (as a compressed challenge), we need a few definitions. Since this is not the focus of the paper, we will only sketch a possible choice. For this, we have to make encoding and decoding of a testing distributions test vector explicit.

A testing distribution χ over $\mathbb{F}_p^m$ with decoding decode is a distribution χ^{enc} over some set of encodings, such that $\chi := \mathsf{decode}(\chi^{\text{enc}})$. The (encoded) challenge is $s \leftarrow \chi^{\text{enc}}$, which the actual (decoded) challenge *vector* is $\mathsf{decode}(s) \in \mathbb{F}_p^m$. Note that *encoding* the challenge (i.e. recovering s) need not be efficient,[58] e.g. if $\mathsf{decode} = \mathsf{PRG}$ and χ^{enc} draws uniformly from $\{0,1\}^\kappa$.

A subdistribution ψ of a testing distribution χ over $\mathbb{F}_p^m$ (with decoding) is called *efficiently samplable* if there is an algorithm Rej, the rejection sampler, such that $\mathsf{decode}(\mathsf{Rej}(\chi^{\text{enc}}))$ has the distribution of ψ, conditioned on $\mathsf{Rej}(\chi^{\text{enc}}) \neq \bot$ (i.e. Rej not rejecting). Note that Rej is given the *encoding*, e.g. the seed of the PRG challenge.

By taking a brief look at the previous definition of computational soundness error, and noting that the weights of efficiently samplable subdistributions of a computationally indistinguishable distributions must be close (up to negligible error), one sees the following: To relax the notion of computational soundness, one can allow a computational soundness error of $\delta_{\mathsf{snd}}^{\mathsf{comp}}$ if *for all efficiently samplable subdisitributions* ψ there exist negligible functions negl_1, negl_2 such that $\mathbb{P}(\boldsymbol{x}_i \leftarrow \psi\colon \det(x_1, \dots, x_m) = 0) \leq (\varepsilon(\psi) + \mathsf{negl}_1(\kappa))\delta_{\mathsf{snd}}^{\mathsf{comp}} + \mathsf{negl}_2(\kappa)$ where $\varepsilon(\psi)$ is the weight of ψ.

This definition is somewhat unwieldy. But, to the best of our knowledge, it is appropriate and we have no simpler notion.

Example E.4. Let PRG by a *non-uniformly* secure PRG. Suppose Conjecture E.1 holds. Then PRG has *statistical* soundness error negligibly close to $\delta_{\mathsf{snd}}(\chi_{\mathsf{uniform}})$. Otherwise, due to Conjecture E.1, there must be a favoured hyperplane H (with $\sigma_\infty(\mathsf{PRG})$ non-negligibly greater than $\sigma_\infty(\chi_{\mathsf{uniform}})$). Encoding this hyperplane as the non-uniform advice z_κ, we can constructed a distinguisher with non-negligible advantage. (If $\boldsymbol{x} \in H$, say PRG. Else, return a random guess.)

Remark E.5. It is not immediately clear how to generalise Example E.4 to other settings, such as families of PRG's or *uniform* security assumptions. Hence, it is rather a testament to the strength of non-uniform security assumptions. However, by using *computational* soundness, the idea may be salvageable.

However, any successful "adversary" Rej induces some ψ with non-negligible weight and non-negligible deviation from $(\varepsilon(\psi) + \mathsf{negl}_1(\kappa))\delta_{\mathsf{snd}}^{\mathsf{comp}} + \mathsf{negl}_2(\kappa)$, where we set $\delta_{\mathsf{snd}}^{\mathsf{comp}}(\mathsf{PRG}) = \delta_{\mathsf{snd}}(\chi_{\mathsf{uniform}})$. Thus, using Conjecture E.1 in a computational setting, the non-negligibly more likely linear dependency of ψ may allow to *sample* a favoured hyperplane H' via ψ, as a preprocessing step. Then, one can use using this H', as the advice above to break the PRG. If this works, then the technique should also apply to families of PRG's (e.g. based on RSA). (This is only an *unfinished* sketch and *not a proof.*)

E.2 Properties of testing distributions

Remark E.6. Let χ_m be a testing distribution. Then the probability that $x_i = x_j$ for $\boldsymbol{x} \leftarrow \chi_m$ is smaller than $\delta_{\mathsf{snd}}(\chi_m)$. This is due to following observation: Let B be the set of all vectors with $x_i = x_j$ and let ε be the probability of B under χ_m, that is, ε is the weight of the subdistribution ψ_B belonging to B. Note that B contains no basis of $\mathbb{F}_p^m$. Thus, the soundness error of χ_m is bounded below by ε. In other words, $\varepsilon \leq \delta_{\mathsf{snd}}(\chi_m)$.

Remark E.6 above is another example demonstrating that the soundness error can be very far from probability that a $\boldsymbol{X} \leftarrow \chi_m^m$ is not invertible. For random *binary* $n \times n$ matrices over the *reals*, the conjecture is that only a $(1 + o(1))n^2 2^{-n}$ fraction is singular. But the probability that $x_i = x_j$ is $\frac{1}{4}$

[58] This is irrelevant for the stronger "perfect" notion of soundness error. Any "encoding" is equivalent in that setting.

in this case. In our case, the matrices are *not* over the reals, but modulo p. This makes a difference, e.g. for $p = 2$, the fraction of singular matrices is roughly $\frac{1}{2}$. But it is natural to assume that for large $p \gg n$ (e.g. an exponential gap as in our case), asymptotics which could "justify" random binary vectors modulo p as testing distributions do hold. Thus, there may be distributions, where $x_i = x_j$ with high probability, but where any n random vectors are independent with very high probabity. Again, this shows the importance of considering subdistributions for δ_{snd}.

Remark E.7. The above argument in Remark E.6 generalises to other relations/properties of vectors which affect invertibility. Thus, a testing distribution must be "well-spread" over a vector space to achieve high (computational) soundness. We note that relations which are computationally hard may affect the soundness heavily, while leaving *computational* soundness "unaffected" (up to a negligible loss).

E.3 Constructions of testing distributions

We consider the tensor product of testing distributions. In a sense, this construction is the unrolling of the recursive steps in our proof systems. The tensor product distribution $\chi = \chi_1 \otimes \ldots \otimes \chi_\ell$ is defined by sampling $\boldsymbol{z} \leftarrow \chi$ via $\boldsymbol{z} = \boldsymbol{z}_1 \otimes \ldots \otimes \boldsymbol{z}_\ell$ for $\boldsymbol{z}_i \leftarrow \chi_i$. Note that $\boldsymbol{z}$ is therefore always an elementary tensor.

Lemma E.8. *Let $\chi = \chi_1 \otimes \ldots \otimes \chi_\ell$ be the tensor product of ℓ testing distributions χ_i on $\mathbb{F}_p^{k_i}$ with $\sigma_\infty(\chi_i)$. Then χ has $\sigma_\infty(\chi) \leq \sum_{i=1}^{\ell} \sigma_\infty(\chi_i)$. If Conjecture E.1 holds, this translates to $\delta_{\mathsf{snd}}(\chi) \leq \sum_{i=1}^{\ell} \delta_{\mathsf{snd}}(\chi_i)$.*

Proof. By induction, it suffices to consider $\ell = 2$. Let $\delta_i := \sigma_\infty(\chi_i)$. Suppose $V = \ker(\varphi)$ is some hyperplane with $\sigma_\infty(\chi) = \mathbb{P}_{\boldsymbol{z}\leftarrow\chi}(\boldsymbol{z} \in V)$, where $\varphi\colon \mathbb{F}_p^{k_1} \otimes \mathbb{F}_p^{k_2} \to \mathbb{F}_p$ is a linear map. Recall that any element $\boldsymbol{z}$ in $\operatorname{supp}(\chi)$ is an *elementary tensor* $\boldsymbol{x} \otimes \boldsymbol{y}$ by definition of $\chi = \chi_1 \otimes \chi_2$.

Since $\varphi(\_ \otimes \boldsymbol{y})$ induces a linear map $\operatorname{Hom}(\mathbb{F}_p^{k_1}, \mathbb{F}_p) \cong \mathbb{F}_p^{k_1}$, we find that

$$\mathbb{P}_{\boldsymbol{x}\leftarrow\chi_1}(\varphi(\boldsymbol{x} \otimes \boldsymbol{y}) = 0) \leq \delta_1$$

for any choice of $\boldsymbol{y}$, except if $\varphi(\_ \otimes \boldsymbol{y}) = 0$ as a map. But $\varphi(\_ \otimes \boldsymbol{y}) = 0$ implies that $\boldsymbol{y} \in K$, where $K := \{\boldsymbol{b} \mid \varphi(\_ \otimes \boldsymbol{b}) = 0\} \leq \mathbb{F}_p^{k_2}$. which is at most a subspace of dimension $(k_2 - 1)$, (else $\varphi = 0$, a contradiction).[59] Thus, we get

$$\mathbb{P}_{\boldsymbol{y}\leftarrow\chi_2}(\varphi(\_ \otimes \boldsymbol{y}) = 0) = \mathbb{P}_{\boldsymbol{y}\leftarrow\chi_2}(\boldsymbol{y} \in K) \leq \delta_2.$$

Then we from $\boldsymbol{z} = \boldsymbol{x} \otimes \boldsymbol{y}$ that

$$\begin{aligned}
\mathbb{P}_{\boldsymbol{z}\leftarrow\chi}(\boldsymbol{z} \in V) &= \mathbb{P}_{\boldsymbol{x},\boldsymbol{y}}(\varphi(\boldsymbol{x} \otimes \boldsymbol{y}) = 0) \\
&\leq (1 - \mathbb{P}_{\boldsymbol{y}}(\boldsymbol{y} \in K)) \max_{\boldsymbol{y}\notin K} \mathbb{P}_{\boldsymbol{x}}(\varphi(\boldsymbol{x})) + \mathbb{P}_{\boldsymbol{y}}(\boldsymbol{y} \in K) \\
&\leq \delta_1 + \delta_2 - \delta_1\delta_2 \\
&\leq \delta_1 + \delta_2.
\end{aligned}$$

Consequently, $\sigma_\infty(\chi) \leq \sigma_\infty(\chi_1) + \sigma_\infty(\chi_2)$. This proves our claim. We stress the importance of using σ_∞, which allowed us to work directly with χ instead of subdistributions. While χ has a simple product structure, where $\boldsymbol{x}$, $\boldsymbol{y}$ are drawn independently, a subdistribution of χ can break this stochastic independence. □

Our recursive arguments actually have a tensor structure, namely they reduce $\mathbb{F}_p^n = (\mathbb{F}_p^k)^\ell$ to $(\mathbb{F}_p^k)^{\ell-1}$ in one step, i.e. they apply a linear map to one of the factors of the tensor product. It is not hard to see that in Section 3.5, Protocol 3.9, one applies $\boldsymbol{x}_1 \otimes \ldots \otimes \boldsymbol{x}_\ell$ to $[\boldsymbol{A}]$ and $\boldsymbol{y}_1 \otimes \ldots \otimes \boldsymbol{y}_\ell$ to $\boldsymbol{w}$ when all batching steps are taken together. It follows easily, that assuming quick-extraction in Lemma 3.10, this means that we can extract a witness by obtaining $n = k^\ell$ separate transcripts with

[59] K is a subspace because $\varphi(\boldsymbol{x} \otimes (\boldsymbol{b} + \gamma\boldsymbol{c})) = \varphi(\boldsymbol{x} \otimes \boldsymbol{b}) + \gamma\varphi(\boldsymbol{x} \otimes \boldsymbol{c})$.

challenges $\boldsymbol{y}_1 \otimes \ldots \otimes \boldsymbol{y}_\ell \leftarrow \chi_1 \otimes \ldots \otimes \chi_k$, one can invert the respective matrix $\boldsymbol{Y} \in \mathbb{F}_p^{n \times n}$ to recover the witness. This way of extraction only needs a $\mathsf{TreeFind}$ algorithm of depth 1. Therefore, simply rewinding until n transcripts are found is sufficient, giving us a runtime of $\mathsf{poly}(\kappa)/\varepsilon$ (where ε is the probability of convincing the verifier). Furthermore, since the adversary induces a subdistribution on the challenges, we obtain a knowledge error of $\ell\delta_{\mathsf{snd}}(\chi_k)$, which is almost optimal. Indeed, the emulator has rewinding-tightness of $O(n)$, which is almost best possible assuming the bound of $O(n/\log(n))$ from Question D.2 holds.

Even though the above is a very special situation, we take this as a hint that Question D.4 has a positive answer. Although the strategy must be quite different in that case.

F Further Remarks on our Implementation

F.1 Arithmetic Circuits

We use $\mathsf{QESA_{ZK}}$ to proof arithmetic circuits. In contrast to existing techniques, $\mathsf{QESA_{ZK}}$ is not restricted to R1CS circuits, but can also handle quadratic equations. Hence we include a preprocessing step in Python, which transforms arithmetic circuits generated by the Pinocchio compiler [43] or jsnark[60] into quadratic equations.

Preprocessing. We preprocess the arithmetic circuit in order to better make use of "quadratic equation gates" (quad gates in the following). To this end, we perform a series of transformations, which in the end yield an equivalent circuit comprised almost entirely of quad gates.

The transformations follow a few simple observations. Some gates can be represented directly by (quadratic) constraints. For example, $\mathsf{xor}(X, Y)$ can be represented as $(1-X)Y + X(1-Y) = 0$. We refer to these as isolated gates in the following. Other gates, such as pack with $\mathsf{pack}(x_1, \ldots, x_k) = \sum_1^k x_i 2^i = x_0 + 2(x_1 + 2(\ldots + 2x_k \ldots))$, can be decomposed into a series of arithmetic gates, hence we coin them decomposable gates. The remaining basic gates, i.e., add, sub, $\mathsf{const\text{-}mul}$, and $\mathsf{const\text{-}mul\text{-}neg}$, can be merged if they precede a mul gate, resulting in a quad gate computing $\sum_{i,j} w_i \Gamma_{i,j} w_j = w_k$. Such a quad gate $\mathfrak{g}$ can be represented by $\boldsymbol{\Gamma}_\mathfrak{g} = \sum_i \boldsymbol{a}_{\mathfrak{g},i} \boldsymbol{b}_{\mathfrak{g},i}^\top - \boldsymbol{e}_1 \boldsymbol{e}_\mathfrak{g}^\top \in \mathbb{F}_p^{n \times n}$, where $\boldsymbol{a}_{\mathfrak{g},i}$, $\boldsymbol{b}_{\mathfrak{g},i}$ are constants describing the gate. We find that $\boldsymbol{w}^\top \boldsymbol{\Gamma}_\mathfrak{g} \boldsymbol{w} = 0$ iff $\mathfrak{g}$ is satisfied by the wire assignment $\boldsymbol{w}$.

Based on these observations, our preprocessing applies the following steps: First, decomposable gates are replaced with other gates depending on their functionality.

Then, each wire w that is either a global output wire or an input wire of an isolated gate, is prepended with a new mul gate where one input is w and the other is the constant-1 wire. Naturally, this is only applied if w is not already the output of a mul gate. The insertion allows for later aggregation of preceding logic into a single quad gate.

Now, all remaining basic gates are merged into quad gates of the form $\sum_{i,j} a_i w_i \Gamma_{i,j} b_j w_j = w_k$. This aggressive optimisation may result in several gates with constant $w_k = 0$. Therefore, constant zeros are propagated through the circuit, eliminating affected gates and wires. Finally the circuit is stripped of floating gates where no output is connected any more and for each remaining gate the corresponding $\boldsymbol{\Gamma}_i$ is extracted.

Results. We evaluate $\mathsf{QESA_{ZK}}$ using the same 512-bit SHA256 circuit without padding as in [16]. The preprocessed circuit consists of 25657 wires, i.e., $\boldsymbol{w} \in \mathbb{F}_p^{25657}$ and 25840 matrices $\boldsymbol{\Gamma}_i \in \mathbb{F}_p^{25657 \times 25657}$. If the $\boldsymbol{\Gamma}_i$ would have been stored without the sparse matrix optimisation, this would require the implementation to hold $25840 \cdot 25657^2 > 2^{43}$ $\mathbb{F}_p$ elements in memory just for the matrices. The sparse representation reduces this to 197465 $\mathbb{F}_p$ elements. Since $\mathsf{QESA_{ZK}}$ expects n to be a power of two, we set $n = 2^{15} = 32768$ and the witness is zero-extended accordingly. As a result, the implementation took 84.2s for $\mathscr{P}$ and 38.1s for $\mathscr{V}$ on average.

[60] See: https://github.com/akosba/jsnark

Parameters	Bulletproofs		Bulletproofs with $\mathsf{IPA}_{\mathrm{noZK}}$	
	$\mathcal{P}$	$\mathcal{V}$	$\mathcal{P}$	$\mathcal{V}$
60 bit	0.26	0.17	0.23	0.11
60 bit × 2	0.47	0.29	0.42	0.21
60 bit × 32	7.4	4.5	6.3	3.7
60 bit × 128	28.9	17.9	26.6	14.2
60 bit × 512	116	78.7	105	55.5
124 bit	0.46	0.29	0.41	0.22
124 bit × 32	14.9	9.2	13.6	7.0
124 bit × 128	59.7	36.8	54.1	29.7
124 bit × 512	238	147	219	117
252 bit	0.95	0.59	0.79	0.46
252 bit × 32	30.2	18.6	26.1	14.3
252 bit × 128	121	74.3	105	58.4
252 bit × 512	484	297	426	227

Table 5: Comparison of non-optimised prover runtime in seconds of aggregate range proofs from [16] with the original IPA and with $\mathsf{IPA}_{\mathrm{noZK}}$. Verification times are only included for completeness. See Section 5 for details.

F.2 Bulletproofs with $\mathsf{IPA}_{\mathrm{noZK}}$

One of our main contributions is the improvement of the original IPA from [16]. In order to practically evaluate the impact of said improvements, we benchmarked Bulletproofs aggregate range proofs with the same parameters as in Table 3, but this time used $\mathsf{IPA}_{\mathrm{noZK}}$ instead. Table 5 shows the results.

G Overview of protocols

In the following, we give an overview of the protocols for with several choices fixed. In particular, we fix $k = 2$. Otherwise, the respective setting is as in the definition of the protocols. Let $\mathcal{S} \subseteq \mathbb{F}_p^\times$. Note that $\mathcal{S}$ are always non-zero. For simplicity, we use the testing distribution $\chi^{(\beta\neq 0)}$, which draws $\alpha \leftarrow \mathcal{S}$ and returns $(\alpha, 1)$. (In this case, $\chi^{(\beta\neq 0)} = \chi^{(\beta)}$.) Moreover, we write $\alpha \leftarrow \chi^{(\beta\neq 0)}$ instead. For other testing distributions χ_n, we consider $\boldsymbol{x} \leftarrow \{1\} \times \mathcal{S}^{n-1}$, that is $x_1 = 1$ always and the other components are random (small) exponents in $\mathcal{S}$. These choices are compatible with the restrictions posed in some protocols. For $\widetilde{\chi}_{2k-1}$ we use an explicit choice $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$, namely $(1, \beta) = \boldsymbol{x} \leftarrow \chi^{(\beta\neq 0)}$, $\boldsymbol{y} = (\beta, 1)$ and $\boldsymbol{z} = (1, \beta, \beta^2)$.

We deviate from the standard presentation of inputs as follows:

- Inputs, which *must* be known to *both* parties are *common* inputs.
- Inputs, which a party (generally the prover) can derive from other inputs, are removed from common inputs.

For example, the target value t in $\mathsf{IPA}_{\mathrm{almZK}}$ is not a treated as a common input, since $\mathcal{P}$ can recompute $t = \langle \boldsymbol{w}', \boldsymbol{w}'' \rangle$ via the witness. This makes data flow (and some optimisations) more explicit, e.g. Remark 4.8. The common input in the usual sense is always given by the "reduced" common input as above and the verifier's additional input. For this point of view to be essentially equivalent to standard zero-knowledge proofs, we merely require that the statement is fixed by the (reduced) common input and $\mathcal{P}$'s (resp. $\mathcal{V}$'s) additional input.

$\mathsf{IPA}_{\mathrm{noZK}}$(Protocol 4.1)

Common Input: $\mathrm{crs} = ([\boldsymbol{g}', \boldsymbol{g}'', Q])$

<u>Prover $\mathscr{P}$</u>		<u>Verifier $\mathscr{V}$</u>
Input: $\boldsymbol{w}', \boldsymbol{w}''$		Input: $[c]$, t
		$\alpha \leftarrow \chi^{(\beta \neq 0)}$
	$\xleftarrow{\alpha}$	
$[Q] \coloneqq \alpha^{-1}[Q]$		$[Q] \coloneqq \alpha^{-1}[Q]$
		$[c] \coloneqq [c] - (\alpha - 1)t[Q]$

Recursive step. Suppose $n > 1$

Prover		Verifier
split $\boldsymbol{w}'$ in halves $\boldsymbol{w}'_1, \boldsymbol{w}'_2$		
split $\boldsymbol{w}'', \boldsymbol{g}', \boldsymbol{g}''$ analogously		
$[\boldsymbol{u}'_{-1}] \coloneqq [\boldsymbol{g}'_2]\boldsymbol{w}'_1$, $[\boldsymbol{u}'_{+1}] \coloneqq [\boldsymbol{g}'_1]\boldsymbol{w}'_2$		
compute $[\boldsymbol{u}''_{\pm 1}]$ analogously		
$v_{-1} \coloneqq \langle \boldsymbol{w}'_2, \boldsymbol{w}''_1 \rangle$		
$v_{+1} \coloneqq \langle \boldsymbol{w}'_1, \boldsymbol{w}''_2 \rangle$		
$[\boldsymbol{u}_{-1}] \coloneqq [\boldsymbol{u}'_{-1}] + [\boldsymbol{u}''_{+1}] + v_{+1}[Q]$		
$[\boldsymbol{u}_{+1}] \coloneqq [\boldsymbol{u}'_{+1}] + [\boldsymbol{u}''_{-1}] + v_{-1}[Q]$		
	$\xrightarrow{[\boldsymbol{u}_{-1}],[\boldsymbol{u}_{+1}]}$	
		$\xi \leftarrow \chi^{(\beta \neq 0)}$
	$\xleftarrow{\xi}$	
$[\boldsymbol{g}'] \coloneqq [\boldsymbol{g}'_1] + \xi[\boldsymbol{g}'_2]$		$[\boldsymbol{g}'] \coloneqq [\boldsymbol{g}'_1] + \xi[\boldsymbol{g}'_2]$
$[\boldsymbol{g}''] \coloneqq \xi[\boldsymbol{g}''_1] + [\boldsymbol{g}''_2]$		$[\boldsymbol{g}''] \coloneqq \xi[\boldsymbol{g}''_1] + [\boldsymbol{g}''_2]$
$\boldsymbol{w}' \coloneqq \xi\boldsymbol{w}'_1 + \boldsymbol{w}'_2$		$[c] \coloneqq \xi^2[\boldsymbol{u}_{-1}] + \xi[c] + [\boldsymbol{u}_{+1}]$
$\boldsymbol{w}'' \coloneqq \boldsymbol{w}''_1 + \xi\boldsymbol{w}''_2$		
$n \coloneqq n/2$		$n \coloneqq n/2$

Start next recursion iteration.

Base case. Suppose $n = 1$

Prover		Verifier
	$\xrightarrow{\boldsymbol{w}',\boldsymbol{w}''}$	
		return true iff:
		$[c] \overset{?}{=} [\boldsymbol{g}']\boldsymbol{w}' + [\boldsymbol{g}'']\boldsymbol{w}'' + t[Q]$
		where $t \coloneqq \langle \boldsymbol{w}', \boldsymbol{w}'' \rangle$

$\mathsf{IPA}_{\mathrm{almZK}}$ (Protocol 4.3)

Common Input: $\mathrm{crs} = ([\boldsymbol{g}', \boldsymbol{g}'', Q])$

<u>Prover $\mathscr{P}$</u>		<u>Verifier $\mathscr{V}$</u>
Input: $\boldsymbol{w}', \boldsymbol{w}''$		Input: $[c_{\boldsymbol{w}}]$, t
$\boldsymbol{r}' \leftarrow \ker(\boldsymbol{w}''^{\top}) \cap \mathbb{M}_n^+$		
$\boldsymbol{r}'' \leftarrow \ker(\begin{pmatrix} \boldsymbol{w}'^{\top} \\ \boldsymbol{r}'^{\top} \end{pmatrix}) \cap \mathbb{M}_n^+$		
$[c_{\boldsymbol{r}}] \coloneqq [\boldsymbol{g}']\boldsymbol{r}' + [\boldsymbol{g}'']\boldsymbol{r}''$		
	$\xrightarrow{[c_{\boldsymbol{r}}]}$	
		$\beta \leftarrow \chi^{(\beta)}$
	$\xleftarrow{\beta}$	
		$t \coloneqq \beta^2 t$
$\boldsymbol{w}' \coloneqq \beta\boldsymbol{w}' + \boldsymbol{r}'$		$[c] = \beta[c_{\boldsymbol{w}}] + [c_{\boldsymbol{r}}] + t[Q]$
$\boldsymbol{w}'' \coloneqq \beta\boldsymbol{w}'' + \boldsymbol{r}''$		

Engage $\mathsf{IPA}_{\mathrm{noZK}}$(crs, $\mathscr{P}(\boldsymbol{w}', \boldsymbol{w}'')$, $\mathscr{V}([c], t)$)

$\mathsf{QESA}_{\text{Inner}}$ (part of Protocol 4.7)

Common Input: $\text{crs} = ([\boldsymbol{g}', \boldsymbol{g}'', Q]), \{\boldsymbol{\Gamma}_i\}$

Prover $\mathcal{P}$ Input: $\boldsymbol{w}, \boldsymbol{r}'$		Verifier $\mathcal{V}$ Input: $[c'_{\boldsymbol{w}}]$
$\boldsymbol{w}' := \binom{\boldsymbol{w}}{\boldsymbol{r}'}$		$\boldsymbol{x} \leftarrow \chi_N$
	$\xleftarrow{\boldsymbol{x}}$	
$\boldsymbol{\Gamma} := \sum x_i \boldsymbol{\Gamma}_i$ $\beta := x_2$ $[g'_1] := \beta^{-1}[g'_1]$ $\boldsymbol{w}'' := \binom{\boldsymbol{\Gamma w}}{\boldsymbol{R r}'}$ $[c''_{\boldsymbol{w}}] := [\boldsymbol{g}'']\boldsymbol{w}''$		$\boldsymbol{\Gamma} := \sum x_i \boldsymbol{\Gamma}_i$ $\beta := x_2$ $[g'_1] := \beta^{-1}[g'_1]$ $[c'_{\boldsymbol{w}}] := [c'_{\boldsymbol{w}}] - (\beta - 1)[g'_1]$
	$\xrightarrow{[c''_{\boldsymbol{w}}]}$	
		$(1, \boldsymbol{s}, \boldsymbol{b}) \leftarrow \chi_n$, $\boldsymbol{s}' := \binom{\boldsymbol{s}}{\boldsymbol{b}}$
	$\xleftarrow{\boldsymbol{s}'}$	
		$t := -\langle \boldsymbol{s}, \boldsymbol{\Gamma}^\top \boldsymbol{s}\rangle$
$\boldsymbol{w}' := \boldsymbol{w}' - \boldsymbol{s}'$ $\boldsymbol{w}'' := \boldsymbol{w}'' + \boldsymbol{\Gamma}'^\top \boldsymbol{s}'$		$[c_{\boldsymbol{w}}] := [c'_{\boldsymbol{w}}] - [\boldsymbol{g}']\boldsymbol{s}' + [c''_{\boldsymbol{w}}] + [\boldsymbol{g}'']\boldsymbol{\Gamma}'^\top \boldsymbol{s}'$

Engage $\mathsf{IPA}_{\text{almZK}}(\text{crs}, \mathcal{P}(\boldsymbol{w}', \boldsymbol{w}''), \mathcal{V}([c_{\boldsymbol{w}}], t))$

$\mathsf{QESA}_{\text{ZK}}$ (Protocol 4.7)

Common Input: $\text{crs} = ([\boldsymbol{g}', \boldsymbol{g}'', Q]), \{\boldsymbol{\Gamma}_i\}$

Prover $\mathcal{P}$ Input: $\boldsymbol{w}$		Verifier $\mathcal{V}$ Input: $\emptyset$
$\boldsymbol{r}' \leftarrow \mathbb{F}_p^2$ $[c'_{\boldsymbol{w}}] := [\boldsymbol{g}']\binom{\boldsymbol{w}}{\boldsymbol{r}'}$		
	$\xrightarrow{[c'_{\boldsymbol{w}}]}$	

Engage $\mathsf{QESA}_{\text{Inner}}((\text{crs}, \{\boldsymbol{\Gamma}_i\}), \mathcal{P}(\boldsymbol{w}, \boldsymbol{r}'), \mathcal{V}([c'_{\boldsymbol{w}}]))$

$\mathsf{QESA}_{\text{Copy}}$(Protocol 4.16)

Common Input: $\text{crs} = ([\boldsymbol{g}', \boldsymbol{g}'', Q]), \{\boldsymbol{\Gamma}_i\}, \{\widetilde{\text{ck}}^{(i)}\}, \{[\widetilde{c}^{(i)}]\}$

Prover $\mathcal{P}$ Input: $\boldsymbol{w}$, $\{\boldsymbol{v}^{(i)}\}$		Verifier $\mathcal{V}$ Input: $\emptyset$
$\boldsymbol{r}' \leftarrow \mathbb{F}_p^2$ $\boldsymbol{w}' := \binom{\boldsymbol{w}}{\boldsymbol{r}'}$ $[c'_{\boldsymbol{w}}] := [\boldsymbol{g}']\boldsymbol{w}'$		
	$\xrightarrow{[c'_{\boldsymbol{w}}]}$	
		$\boldsymbol{\alpha} \leftarrow \chi_{M+1}$ with $\alpha_0 = 1$
	$\xleftarrow{\boldsymbol{\alpha}}$	
$[c'_{\boldsymbol{w}}] := \alpha_0[c'_{\boldsymbol{w}}] + \sum_i \alpha_i[\widetilde{c}^{(i)}]$ $\{\boldsymbol{\Gamma}_i\} \cup= \{\boldsymbol{\Gamma}^{(k)}_{\text{copy}}$ for $k \in \mathcal{I}\}$ $\boldsymbol{w}' := \alpha_0\boldsymbol{w}' + \sum_i \alpha_i \boldsymbol{v}^{(i)}$ decompose $(\boldsymbol{w}, \boldsymbol{r}') := \boldsymbol{w}'$		$[c'_{\boldsymbol{w}}] := \alpha_0[c'_{\boldsymbol{w}}] + \sum_i \alpha_i[\widetilde{c}^{(i)}]$ $\{\boldsymbol{\Gamma}_i\} \cup= \{\boldsymbol{\Gamma}^{(k)}_{\text{copy}}$ for $k \in \mathcal{I}\}$

Engage $\mathsf{QESA}_{\text{Inner}}((\text{crs}, \{\boldsymbol{\Gamma}_i\}), \mathcal{P}(\boldsymbol{w}, \boldsymbol{r}'), \mathcal{V}([c'_{\boldsymbol{w}}]))$

$\mathsf{LMPA}_{\text{noZK}}$(Protocol 3.9)

Common Input: $[\boldsymbol{A}]$

Prover $\mathcal{P}$ | Verifier $\mathcal{V}$

Input: $\boldsymbol{w}$ | Input: $[\boldsymbol{t}]$

Recursive step. Suppose $n > 1$

$[\boldsymbol{u}_{-1}] \coloneqq [\boldsymbol{A}_1]\boldsymbol{w}_2$

$[\boldsymbol{u}_{+1}] \coloneqq [\boldsymbol{A}_2]\boldsymbol{w}_1$

$$\xrightarrow{[\boldsymbol{u}_{-1}],[\boldsymbol{u}_{+1}]}$$

$\xi \leftarrow \chi^{(\beta \neq 0)}$

$$\xleftarrow{\xi}$$

$[\boldsymbol{A}] \coloneqq [\boldsymbol{A}_1] + \xi[\boldsymbol{A}_2]$ | $[\boldsymbol{A}] \coloneqq [\boldsymbol{A}_1] + \xi[\boldsymbol{A}_2]$

$\boldsymbol{w} \coloneqq \xi\boldsymbol{w}_1 + \boldsymbol{w}_2$ | $[\boldsymbol{t}] \coloneqq [\boldsymbol{u}_{-1}] + \xi[\boldsymbol{t}] + \xi^2[\boldsymbol{u}_{+1}]$

$n \coloneqq n/2$ | $n \coloneqq n/2$

Start next recursion iteration.

Base case. Suppose $n = 1$

$$\xrightarrow{\boldsymbol{w}}$$

return true iff $[\boldsymbol{A}]\boldsymbol{w} \overset{?}{=} [\boldsymbol{t}]$

$\mathsf{LMPA}_{\text{simpleZK}}$

Common Input: $[\boldsymbol{A}]$

Prover $\mathcal{P}$ | Verifier $\mathcal{V}$

Input: $\boldsymbol{w}$ | Input: $[\boldsymbol{t}]$

$\boldsymbol{r} \leftarrow \mathbb{F}_p^n$

$[\boldsymbol{a}] \coloneqq [\boldsymbol{A}]\boldsymbol{r}$

$$\xrightarrow{[\boldsymbol{a}]}$$

$\beta \leftarrow \chi^{(\beta \neq 0)}$

$$\xleftarrow{\beta}$$

Engage $\mathsf{LMPA}_{\text{noZK}}([\boldsymbol{A}], \mathcal{P}(\beta\boldsymbol{w} + \boldsymbol{r}), \mathcal{V}(\beta[\boldsymbol{t}] + [\boldsymbol{a}]))$