Parts of this work appear as extended abstracts in ePrint reports 2022/1593 and 2023/1406. This is the full version.

Beyond the circuit:

How to Minimize Foreign Arithmetic in ZKP Circuits

Michele Orrù
CNRS*
m@orru.net

George Kadianakis
Ethereum Foundation
george.kadianakis@ethereum.org

Mary Maller
Ethereum Foundation, PQShield
mary.maller@ethereum.org

Greg Zaverucha
Microsoft Research
gregz@microsoft.com

Abstract

Zero-knowledge circuits are frequently required to prove gadgets that are not optimised for the constraint system in question. A particularly daunting task is to embed foreign arithmetic such as Boolean operations, field arithmetic, or public-key cryptography. We construct techniques for offloading foreign arithmetic from a zero-knowledge circuit including (i) equality of discrete logarithms across different groups; (ii) scalar multiplication without requiring elliptic curve operations; (iii) proving knowledge of an AES encryption. To achieve our goal, we employ techniques inherited from rejection sampling and lookup protocols. We implement and provide concrete benchmarks for our protocols.

1 Introduction

Zero-knowledge proofs [GMR89] allow a prover to convince a verifier about the truth of a statement without revealing more information than its validity. They are a core tool in complexity theory, cryptography, and security. Over the decades, the cryptographic community has witnessed a transformative evolution of zero-knowledge proofs, from theoretical tools to practical systems, with a focus on proving statements about NP relations. The surge of interest in real-world applications, such as blockchain scalability [BCL+21], private payments [BCG+14], and more recently authenticity of images and documents [KHSS22, BCG+22], has catalyzed the development of efficient zero-knowledge proof across a variety of systems.

Most real-world proof systems today express statements in terms of arithmetic circuits over finite fields, and the engineering effort involved is significant. For instance, programming a zero-knowledge statement in rank-1 constraint systems [BCG+13] (R1CS) requires to formulate the statements as an arithmetic circuit where the only operations allowed are additions and multiplications in the field. This process is highly inefficient, non-trivial and prone to errors. Plonkish relations [GP20] introduced the notion of *custom gates*, which improve efficiency by introducing reusable gadgets but require careful protocol engineering and bloat the proof size. Recent works designing and relying on algebraic hashes [AGR+16, GKR+21, GKL+22] and curve cycles [BCTV14, BCG+20b, KS23] try to reduce the burden of embedding foreign field arithmetic inside the zero-knowledge statement. Additionally, mistakes in the instantiations might result in either efficiency losses, or void the security guarantees of the proof system itself [zkb].

In practice, a large engineering effort has been put into securely programming zero-knowledge statements for computations over (i) symmetric-key operations, (ii) public-key operations, and (iii) arithmetic over a *foreign* finite field. All above operations often come at a high cost: as an example, expressing a non-algebraic cipher like AES using Marlin [CHM+20] in BLS12-377 [BLS04] costs 212848 constraints, consuming around 18s on modern hardware; alternatively, the jSNARK library (which internally relies on [BCTV14]) implements AES in 14240 R1CS constraints.

Similarly, expressing elliptic-curve operations inside circuits is extremely expensive and error-prone.[1] Specifically, performing arithmetic on the non-native field is multiple orders of magnitude more expensive than doing native field arithmetic due to using bit-decomposition to simulate the non-native field [azt]. For instance, to simulate a non-native field multiplication for the BN254 curve base field using the BN254 scalar field, there is a 600x overhead.[2] For this reason, recursive protocols typically adopt elliptic curve cycles [BGH19, KS23]; however these techniques can lead to more complicated protocols with subtle vulnerabilities [NBS23].

*Part of this work was conducted while the author was at UC Berkeley.

[1] https://penumbra.zone/pdfs/zksecurity_penumbra_2023.pdf

[2] https://github.com/arkworks-rs/nonnative

In sum, the engineering effort to express zero-knowledge statements is significant, and the resulting statements are often inefficient or insecure. In this paper we attempt to address this problem by providing a set of techniques that allow to express zero-knowledge statements without the need for foreign arithmetic. We provide three techniques that respectively avoid: (i) foreign group arithmetic, (ii) foreign field arithmetic, and (iii) foreign Boolean arithmetic.

1.1 Our contributions

We provide the following techniques to remove the use of foreign arithmetic in zero-knowledge circuits:

(i) Rejection sampling can be used to prove that two secrets, committed across different groups, are equal. We will use $\mathbb{G}_p$ and $\mathbb{G}_q$ to denote groups of prime order p and q, with generators (G_p, H_p) and (G_q, H_q) and constant $b_x > 0$ such that $b_x < \lceil \log_2(\min(p,q)) \rceil - 2$. We prove the following theorem.

Theorem 1 (informal)**.** *If discrete logarithm is hard in $\mathbb{G}_p$ and $\mathbb{G}_q$, and $0 \leq x < 2^{b_x}$, then Π_{dleq} of Figure 1 is a zero-knowledge argument of knowledge for the relation*

$$\mathsf{R}_{dleq} \coloneqq \{((x, r_p, r_q), X_p, X_q) : X_p = xG_p + r_pH_q \ \wedge \ X_q = xG_q + r_qH_q\}\,.$$

Intuitively, we require that x is bounded ($0 \leq x < 2^{b_x}$) to simplify the reasoning about the relation above, and work with x over $\mathbb{Z}$. We formalize this precondition by requiring a zero-knowledge proof be provided as input to the protocol. This can be guaranteed via an explicit range proof either in $\mathbb{G}_p$ or $\mathbb{G}_q$, or one of the input commitments may already be known to satisfy the length constraint (e.g. because of an earlier range proof). Efficient range proofs exist and for the explicit range proof we directly use prior works without modifications (see Section 3.2). Our protocol can be extended to prove discrete log relations of vectors $\vec{x}$ using standard composition techniques.

While the above relation can always be proven with zero-knowledge proofs for arbitrary NP statements, generic approaches must embed foreign group arithmetic in the proof statement and fail to deliver prover efficient protocols, whereas proofs with our protocol Π_{dleq} require on average 3–6 scalar multiplications in each of $\mathbb{G}_p$ and $\mathbb{G}_q$ (more details in Table 2) and as little as 81 bytes.

(ii) A (perhaps surprisingly) simple Σ-protocol allows circuit designers to relocate elliptic curve group operations outside the circuit. The approach involves performing a Σ-protocol outside the circuit and subsequently "binding" it inside the circuit.

Modern zero-knowledge protocols encounter significant prover computational overhead due to the execution of elliptic curve group operations within SNARK circuits. These operations are required for aggregating proofs, opening algebraic commitments, or verifying signatures inside circuits. It is possible to perform these elliptic curve group operations using non-native field arithmetic, but the costs in terms of field operations is multiple orders of magnitude slower than native field operations: over a million R1CS constraints are needed for expressing a single BN254 scalar multiplication in R1CS, and 14,000 rows are needed in Plonk using custom gates. Furthermore, implementing these operations inside circuits is complicated resulting in code that is prone to errors and difficult to audit. By comparison, modern algebraic hash function [AGR+16, GKR+21, BBC+22] have small circuit representations and are two orders of magnitude faster: for instance, Poseidon requires roughly 220 constraints to be expressed in R1CS and 240 rows to be expressed in Plonk. We prove the following theorem:

Theorem 2 (informal)**.** *Let $M \in \mathbb{G}_p^{m \times n}$ be a matrix over some group $\mathbb{G}$ of polynomial size. If H is a collision-resistant hash function and M is collision-resistant, then Π_{dlhash} of Figure 2 is a zero-knowledge argument of knowledge for the relation*

$$\mathsf{R}_{dlhash} \coloneqq \left\{\left(\vec{x}, (\vec{X}, x_h)\right) \ : \vec{X} = M\vec{x} \ \wedge \ x_h = \mathsf{H}(\vec{x})\right\}.$$

Informally, the requirement on M of collision-resistance means that is hard to find two distinct vectors $\vec{x}, \vec{x}'$ such that $M\vec{x} = M\vec{x}'$ (cf. the Kernel-Matrix Diffie-Hellman problem [MRV16]). The Π_{dlhash} protocol effectively avoids non-native group operations by performing algebraic hash invocations. This results in big improvements on the number of constraints. We provide concrete benchmarks for our protocol Π_{dlhash} in Section 4.2.

(iii) Boolean arithmetic can be implemented efficiently with lookup protocols. To demonstrate this, we provide a protocol to prove in zero-knowledge that the Rijndael (AES) cipher is correctly encrypting a committed value, using a single *lookup protocol*. We prove the following theorem:

Theorem 3 (informal)**.** *If discrete logarithm is hard in* $\mathbb{G}$*, then* Π_{aes} *of Figure 3 is a zero-knowledge argument of knowledge for the relation*

$$\mathsf{R}_{aes} := \left\{ \left((\vec{m}, \mu, \vec{k}, \kappa), ctx, M, K \right) : M = \sum_i m_i G_i + \mu H_i \ \wedge \ K = \sum_i k_i G_i + \kappa H_i \ \wedge \ ctx = \mathsf{AES}(\vec{k}, \vec{m}) \right\}.$$

where AES is the Rijndael block cipher, and $\vec{m}, \vec{k}$ are the bit-strings of (respectively) message and key. The protocol has linear prover and verifier cost, and internally relies on a lookup protocol Π_{lup} without requiring any range proof or boolean arithmetic. In practice the AES circuit is small thus linear-time verification is concretely efficient and saves the prover significant overhead. We introduce a novel lookup protocol that relies solely on Σ-protocols, implement and benchmark the resulting protocol, providing concrete estimates and guidelines for implementors. Our code is open-source and released under BSD license.[3]

The overall prover cost of our lookup techniques for AES is much lower than the application of a generic-purpose SNARK. We provide an implementation and concrete benchmarks for Π_{aes} in Section 5.2, for AES-128 and AES-256, and include comparisons with state-of-the-art MPC-in-the-Head based proofs, which are generally considered to be well-suited for Boolean computations. Our AES-128 zero-knowledge proof runs in 6 ms to reduce an encryption proof into a single batch lookup relation of less than 2000 elements in a table of 768 elements. Benchmarks for the entire proof creation and verification can be seen in Table 4.

1.2 Applications

We list below some applications that can benefit from our techniques.

Linking credentials and assets inside proofs. One overarching theme of our contributions is the ability to easily and efficiently *link* commitments and credentials inside zero-knowledge proofs.

- **Linking commitments.** Consider a PKI system which stores Alice's record (for instance, the IP file system IPFS [Ben14]). The PKI (that is the case for IPFS) requires us to commit to Alice's data using a Merkle-based commitment scheme to get commitment c_m. Now consider that we want to prove in zero-knowledge some property of Alice's record (e.g. a range check) using a proof system that expects a Pedersen commitment c_p. A naive approach would involve the prover to open c_m and c_p inside the circuit to prove it opens to the same witness data, however opening c_p inside the circuit is very expensive because of non-native field arithmetic. Using Π_{dlhash}, linking c_m and c_p becomes efficient by avoiding non-native field arithmetic.

- **Linking Assets.** Consider interaction between two blockchains that use different elliptic curves. This can happen for example, if an L2 zkEVM needs to create a proof about data posted on an L1 (e.g. Bitcoin [Nak08], Ethereum [But14]). If those two systems use different elliptic curves, linking a commitment between them can be done with Π_{dleq} rather than a generic SNARK with non-native field arithmetic which would result in a larger set of assumptions, attack surface, and engineering costs.

- **Linking anonymous credentials.** By *credential*, we refer to anonymous credentials defined in groups of prime order such as CL [CL04], BBS+ [LKWL22, BBS04, ASM06], U-Prove [PZ13] and Brands [Bra94], PS signatures [PS16], Coconut [SAB+19] and keyed-verification credentials such as those used in the Signal private groups system [CMZ14, BBDT16, CDDH19, CPZ20]. For instance, consider a credential that contains a user's account ID, e-mail address, phone number, and social security number. A second credential can now be linked to the first by including the user ID attribute in both credentials. This is possible both for credentials provided by the same issuer, as well as across multiple issuers.

  Linking credentials allows to essentially join authorization attributes via a unique linking attribute (say, a small 128-bit scalar), and giving the relying party assurance that both credentials were issued to the same user. In the case of sharing credentials across issuers, the issuers must rely on each other's credential security for the attributes being issued. The second issuer may, without coordination with any other party, use a blind issuance protocol to use a unique attribute from the first credential, as a linking attribute in the second credential and use Π_{dleq} to prove that the linking attribute in the second credential is the same as the linking attribute in the first credential.

[3] https://github.com/mmaker/tinybear

Concurrently-secure blind signatures. Fuchsbauer and Wolf [FW22] recently constructed an efficient blind signature scheme that produces valid Schnorr signatures, as supported by Bitcoin [Nak08], and enables for concurrently-secure blind coin swaps.[4] Roughly speaking, in order to provide concurrent security for Schnorr blind signatures and avoid ROS attacks [Sch91, BLL+21] the user commits and proves knowledge of the blinding factors and the message when producing the signature. This proof requires two ingredients: proving that scalar multiplication in an elliptic curve has been done correctly, and proving that the hash function has been correctly evaluated. Since no obvious choice of proof system can help here, the authors of [FW22] resorted to a generic SNARK, and report a running time of more than 3 hours of proving time using Groth16 [Gro16], or 2.5 minutes in Plonk [GWC19]. Using the techniques from Section 5, (or tweaking AES and Π_{aes} to be used as a collision-resistant hash function) it is possible to remove the scalar multiplication from the zero-knowledge circuit (at the cost of a small soundness error) and rely on a zero-knowledge circuit that solely proves knowledge of a hash pre-image and a linear relation of field elements. We estimate an overall runtime of less than a second for proving the same relation, and a much more slim circuit to audit.

Verifiable encryption. Commit-and-Prove Zero-Knowledge Proof systems (CP-ZKPs), introduced by Kilian [Kil90], later by Canetti et al. [CLOS02], and more recently by Benarroch et a. [BCF+21b] are a generalization of zero-knowledge proofs in which the prover proves statements about values that are committed.

The protocol Π_{aes} can be used to link an AES-encrypted message and an AES key to a Pedersen commitment. This can be used to prove that a ciphertext is a valid encryption of an authenticated data structure, or prove validity of some algebraic properties of the message, while maintaining post-quantum privacy.

Future work. We believe that some of our techniques can be extended and leave them as open problems. The protocol Π_{dlhash} is linear in the size of the linear map M, but we wonder if other techniques could be employed to reduce the asymptotic proof size. Our techniques for proving Rijndael encryption are sufficiently modular and could be implemented using a post-quantum polynomial commitment scheme such as FRI [BBHR18], or linear-evaluation proofs using MPC-in-the-Head techniques [IKOS07]. We provide data suggesting that using lookups will lead a more efficient post-quantum candidate than the current state-of-the-art NIST candidate based on Aurora [BCR+19].

1.3 Related work

Discrete Logarithm EQuality (DLEQ) proofs. The special case where $p = q$ has been studied by Chaum and Pedersen [CP93]. Benarroch et al. [BCF+21a] provide a protocol for proving equality of commitments over $\mathbb{Z}_N^*$ and elliptic-curve groups. The problem of efficiently proving discrete logarithm equality across different groups can be found in Camenisch and Lysyanskaya [CL02], who describe an efficient zero-knowledge proof of knowledge that a committed value is in an accumulator. Values are committed in a group where the discrete logarithm (DL) is hard, while the accumulator is constructed in an RSA group. The problem considered in this work is slightly different, because we consider two groups where DL is hard. The problem of proving discrete logarithm equality across two generic DL groups was addressed by Agrawal, Ganesh, and Mohassel [AGM18], who also underline the applications for extending SNARKs. A protocol directly comparable to ours is given in [AGM18], however the protocol is more involved than ours (e.g., it requires commitments to the bits of x in both $\mathbb{G}_p$ and $\mathbb{G}_q$ and performs a range check on each of them) and thus more expensive. zWe note that bit decomposition and range checking can be omitted in most circumstances, as they are already performed within a larger proof system, or the one of the commitment is already trusted (this is the case of all examples in the applications section).

In the cryptocurrency area, the problem was already highlighted in Zerocoin [MGGR13], where they use the same techniques of Camenisch and Lysyanskaya [CL02] to provide an anonymous cryptocurrency. Dagher et al. [DBB+15] provide proofs of assets, solvency and non-collusion for Bitcoin, evoking the need of zkSNARKs for efficiency but the associated cost in expressing a large circuit. Sun et al. [SSS+22] formulate the problem of proving discrete logarithm equality across pairing-friendly and non-pairing-friendly groups.

The aborting technique we use to avoid leaking information about the secret when the prover sends a response computed over the integers originates in [Lyu08, Lyu09], where it was used in the context of lattice-based signatures. It then was adapted to signatures based on the short discrete log problem in Abdalla et al. [AFLT12]. The setting of this latter work is closer to ours and we use the main lemma from it in our analysis.

Public-key operations. Ben-Sasson, Chiesa, and Tromer [BCTV14] introduced the notion of *curve cycles*, elliptic curves where the scalar field of one curve is equal to the coordinate field of another. Note that the case where the scalar field is the coordinate field is called *anomalous curve* and they are susceptible to attacks [Sma99, Yas12]. This approach

[4] https://jonasnick.github.io/blog/2018/07/31/blind-signatures-in-scriptless-scripts/

has been shown itself extremely powerful, but also dangerous: little is known about, and deferred computations in recursive settings [Val08, CT10, BGH19] have already had a history of subtle vulnerabilities arising only when the elliptic curve is instantiated with cycles [NBS23].

GoblinPlonk[5] introduces a mechanism for deferring expensive operations in SNARK circuits. For instance, when encountering an expensive operation $X = xG$, the prover defers the actual computation and directly provides the final result for X. Once multiple such operations have been deferred, a specialized circuit verifies the correctness of all deferred operations in a single step. Π_{dlhash} can be effectively utilized in a GoblinPlonk final circuit to expedite the verification process. Furthermore, when integrating Π_{dlhash} into a larger circuit, the prover can provide the final result for X and defer its actual computation by pushing it to a Sigma protocol, following a similar approach as in GoblinPlonk.

Sun et al. [SSS+22] introduced the *delegated Schnorr protocol* technique enabling the efficient use of Pedersen commitments on a different elliptic curve from the one employed in the SNARK. Our Π_{dlhash} is similar to this technique, but extends it to accommodate Pedersen commitments of vectors rather than just single elements. Additionally, we provide a security analysis of our generic protocol.

Chase et al. [CGM16] introduced a technique that combines algebraic-based proof protocols, such as Σ-protocols, with proofs based on garbled circuits. This integration efficiently handles algebraic operations in the former and non-algebraic operations (e.g. hash functions) in the latter. The linkage between these proof systems relies on using a *private* garbling scheme to compute a one-time MAC of the witness and then proving the correctness of the MAC using a Σ-protocol. However, this technique requires garbled circuits with privacy properties, as the verifier learning the MAC value directly reveals the witness. As a result, the approach is not immediately applicable to proof systems that do not employ private garbled circuits. In contrast, Π_{dlhash} can be utilized by any circuit-based proof system without being restricted by the need for private garbled circuits.

LegoSNARK [CFQ19] introduced a generic framework for linking different proof systems. Using the commit-and-prove paradigm [Kil90, CLOS02], it provides a framework and generic compiler to facilitate the generic integration of proof systems and demonstrates its applicability across various use cases. Π_{dlhash} can be seen as providing an efficient specialized LegoSNARK link between Σ-protocols and generic SNARK protocols.

Symmetric-key operations. Previous efforts implementing symmetric primitives (like hash functions) within zero-knowledge SNARKs faced significant computational overhead due to their non-algebraic nature. While advancements have been made to accelerate these implementations (e.g., [zca]), many optimizations are tightly coupled to specific proof systems, limiting their broader applicability across ZK frameworks. Alternative approaches to eliminating foreign arithmetic from symmetric-key operations include:

- *Algebraic Symmetric-Key Primitives.* A recent research direction aims at building symmetric-key primitives that are "algebraic" or "zk-friendly" to alleviate the burden of embedding Boolean arithmetic in zero-knowledge statements [AGR+16, GKR+21, GKL+22, BBC+22]. While this avoids the foreign arithmetic challenge entirely, these primitives received little cryptanalytic effort so far, are still evolving, and are not easily interoperable with other systems: while efficient in terms of operations over a large-characteristic field, algebraic hash functions are significantly slower than native implementations of standard encryption and hashing primitives.

- *MPC-in-the-Head approaches.* A long-standing research line leveraging techniques from secure multi-party computation (MPC) has produced zero-knowledge proofs for symmetric-key operations. It's widely acknowledged in the cryptographic community that MPC-in-the-Head techniques offer the best performance for proving small Boolean circuits. Proof systems tailored specifically to proving AES pre-images gained recent interest due to the NIST call for post-quantum cryptography.[6]. However, these digital signatures generated using MPC/VOLE-in-the-head techniques have asymptotically large proof sizes. Efficiently combining these techniques with IOP-based proof systems like Plonk [GWC19], Halo2, STARKs, and Marlin [CHM+20] remains a challenge, as the underlying commitment schemes of these systems are vastly different. We provide a more detailed comparison in Section 5.2.

An independent and concurrent work by Arun, Setty, and Thaler (*Jolt* [AST23]) underscores our same core concept (Section 5): lookup protocols offer significant utility for performing custom gate computations and eliminating the need for foreign arithmetic. Their approach relies internally on a generic lookup protocol implemented using Merkle Trees (requiring proofs of knowledge for hash pre-images), permutation checks, or a novel lookup argument called Lasso [STW23]. Our protocol, Π_{aes}, leverages the same observation but focuses specifically on the context of an AES circuit. Lasso, introduced concurrently by Setty, Thaler, and Wahby [STW23], inherits techniques from memory-checking [BEG+91] and multivariate sumcheck, while our protocol draws upon *logUp* [Hab22]. Both approaches exploit tables with special structures to achieve

[5] https://hackmd.io/@aztec-network/B19AA8812

[6] https://csrc.nist.gov/Projects/post-quantum-cryptography

similar efficiency optimizations described in Section 5.2. Lasso operates in the preprocessing model to provide logarithmic verifiers, whereas our proposed lookup instantiation offers a linear-time verifier without requiring an offline phase, removes the need of shuffle proofs, and presents stronger soundness properties. Despite our approach having worse asymptotics by requiring a linear-time verifier, we believe the concrete performances are better than what was previously known. We delve into a more in-depth comparison of these two lookup protocols in Section 5.2.

2 Preliminaries

We denote by $(\mathbb{G}, p, G, H)$ the description of a group $\mathbb{G}$ of prime order p, with two "nothing-up-my-sleeve" generators G, H (that is, two generators in $\mathbb{G}$ such that the discrete logarithm of H to the base G is not known to anyone). We denote group operations additively, and given a scalar $x \in \mathbb{Z}_p$ we denote with xG scalar multiplication. When needing multiple "nothing-up-my-sleeve" (NUMS) generators (that is, generators whose respective DL is not known), we will consider $G_1, G_2, \ldots, G_n, H$. We denote probabilistic algorithms in sans-serif, and by writing $y \leftarrow \mathsf{M}(x)$ we denote the act of sampling the value y from the probabilistic algorithm M on input x. We assume that probabilistic algorithms run in time polynomial in the security parameter λ (abbrev p.p.t.) and have the security parameter implicitly as input. We use standard vector notation: by $\vec{x} \in \mathbb{Z}_p^n$ we refer to elements $(x_1, x_2, \ldots, x_n)$, with $\langle \vec{x}, \vec{y} \rangle$ we denote the inner-product $\sum_i x_i y_i$ and by $\vec{x} \otimes \vec{y}$ the "vectorized" tensor product $[x_i y_j]_{i \cdot n + j}$.

DL assumption. The Discrete Logarithm problem asks, given a group generator GrGen, a group description $(\mathbb{G}, p, G) \leftarrow \mathsf{GrGen}(1^\lambda)$ and a uniformly-random group element $X \leftarrow\!\!\$\ \mathbb{G}$, to find $x \in \mathbb{Z}_p$ such that $X = xG$. The *discrete logarithm (DL) is hard* for GrGen if no p.p.t. algorithm solves the discrete logarithm problem with more than $\mathsf{negl}(\lambda)$ advantage.

Pedersen commitments. Pedersen's commitment scheme [Ped92] lets us *commit* to a value $x \in \mathbb{Z}_p$. To do so, sample $r \leftarrow\!\!\$\ \mathbb{Z}_p$ and set

$$C \coloneqq xG + rH\,.$$

We say that C is a Pedersen commitment. A pair $(x, r) \in \mathbb{Z}_p^2$ is a *valid opening* if $C = xG + rH$. Pedersen commitments are *perfectly hiding* and *computationally binding* under the discrete logarithm assumption.

Informally, perfectly hiding means that no information about the pair (x, r) is revealed by C. Computationally binding means that no efficient adversary can produce two different valid openings (x, r) and (x', r') for a commitment C. Any adversary that given as input a group description is able to output a commitment C along with two distinct valid openings immediately gives a solution to an instance of DL. In fact, if (x, r) and (x', r') are a pair of valid openings, then $\log_G H = (r - r')^{-1}(x - x')$.

We will also use the well-known fact that Pedersen commitments are *additively homomorphic*: given commitments C, C', the sum of the openings $(x + x',\ r + r')$ is valid for the sum of the commitments $C_p + C'_p$. In addition, when committing to multiple elements $x_1, x_2, x_3, \cdots, x_n$ we will use the notation $C = \sum_i x_i G_i + rH$ as the commitment to the vector $x = (x_1, x_2, x_3, \ldots, x_n)$.

Σ-protocols. We briefly recap Σ-protocols. Our definition is a slight variation of the standard Σ-protocol definition from Cramer [Cra97] (as described in Boneh–Shoup [BS20, §19.4]), except we make a few minor changes to model the prover's ability to abort the protocol. Let R be a binary relation between statements denoted by ϕ and witnesses denotes by w. By $\mathsf{R}(\phi)$ we denote the set of possible witnesses for the statement ϕ in R. A Σ-protocol for the relation R is a three-move protocol between a prover (with inputs ϕ and w) and a verifier (with input ϕ) consisting of a triple of efficient algorithms $(\mathsf{Com}, \mathsf{Ch}, \mathsf{Resp})$ run as follows:

- the prover executes $(a, \rho) \leftarrow \mathsf{Com}(\phi, w)$, sends a and internally stores the state ρ. Com is a randomised algorithm and may have additional inputs such as the group description and security parameter
- the verifier sends $c \leftarrow \mathsf{Ch}()$ to the prover; c is distributed uniformly at random from a fixed set of possible challenges
- the prover calls $\mathsf{Resp}(\phi, w, \rho, c)$ which may return some value z or abort (in which case we consider $z = \perp$)
- finally, the verifier calls $\mathsf{Verify}(\phi, (a, c, z))$ which returns a bit $b \in \{0, 1\}$. If $b = 1$ the verifier accepts the proof, otherwise rejects.

The tuple of exchanged messages (a, c, z) is called *transcript*; a is called commitment, c is called challenge, and z response. An *accepting transcript* (a, c, z) for ϕ is a transcript for which $\mathsf{Verify}(\phi, (a, c, z)) = 1$. Σ-protocols must satisfy:

- **Completeness:** A Σ-protocol is δ-complete if honestly-generated transcripts always verify, except when the prover aborts (with probability δ). More formally, for all honestly generated transcripts (a, c, z) and $(\phi, w) \in \mathsf{R}$ we have that
$$\Pr[\mathsf{Verify}(\phi, a, c, z) = 1 \mid z \neq \perp] = 1\,, \text{ and } \Pr[z = \perp] = \delta$$
over the choice of prover randomness.

- **Special soundness:** A Σ-protocol is (computationally) special sound if there exists an efficient extractor Ext such that for any p.p.t. adversary outputting a statement ϕ and two (non-aborting) accepting transcripts $(a, c, z), (a, c', z')$ for ϕ such that $c \neq c'$, $\mathsf{Ext}(\phi, (a, c, z), (a, c', z'))$ returns a valid witness $w \in \mathsf{R}(\phi)$ except with probability ϵ. The probability ϵ is called the *knowledge error* of the protocol.

- **Honest verifier zero-knowledge:** A Σ-protocol is honest verifier zero-knowledge (HVZK) if there exists an efficient simulator algorithm Sim such that for all $(\phi, w) \in \mathsf{R}$ the distributions
$$\{(a, z) \mid c \leftarrow \mathsf{Ch}(); (a, z) \leftarrow \mathsf{Sim}(\phi, c)\}, \text{ and}$$
$$\{(a, z) \mid c \leftarrow \mathsf{Ch}(); (a, \rho) \leftarrow \mathsf{Com}(\phi, w);\ z \leftarrow \mathsf{Resp}(\phi, w, \rho, c)\}$$
are indistinguishable. Our definition is sometimes referred to as *special HVZK* – *special* since the challenge is input to the simulator, as opposed to being chosen by the simulator. If the two distributions are perfectly indistinguishable (which can be the case for all our protocols), we will say the protocol enjoys perfect special HVZK since the simulated distribution is identical to the real one.

Two example Σ-protocols relevant to our protocol are Schnorr's protocol [Sch91], which proves knowledge of a discrete logarithm and Okamoto's protocol [Oka93], which proves knowledge of the opening of a Pedersen commitment. The protocols and their knowledge extractors are well-known in the literature, see for example the description in the textbook of Boneh and Shoup [BS20, §19.1, 19.5.1].

Subprotocols and sequential composition. We often reduce an involved relation into an easier sub-claim that is then deferred to another proof. Instantiations of these sub-protocols as Σ-protocols is also provided, but we provide separate proofs of security for each component.

If we consider the overall interactive protocol, the resulting proof is $(2, \ldots, 2)$-special sound: it is possible to build a set of accepting transcripts, arranged in a (binary) tree structure, where every branching node at layer i and index j splits into the two transcripts demanded for the i-th layer at the j-th round, for which we provide an extractor. In section Section 5 we slack the above notion slighly proving $(3, \ldots, 3, 2)$-special soundness. We note that we compose at most a constant number of special-sound protocols. Bootle's et al. [BCC+16, Lemma 1] show that a $(n_1, \ldots, n_k)$-special sound protocol satisfies witness-extended emulation [Lin03, Def. 10] if $\prod_i^k n_i = \mathsf{poly}(\lambda)$. (In our case $\prod_i n_i < 3^{\log p}$ and the logarithmic factor is involved only when calling the sumcheck protocol.) A similar approach to ours has been taken by Attema and Cramer [AC20]. For honest-verifier zero-knowledge, it is possible to consider the transcripts generated by each honest-verifier zero-knowledge simulator.

We expect our system to be modular and security to hold also when using different zero-knowledge proof systems. This is particularly relevant as we expect sub-claims to be shown as a part of a larger proof being performed in an outside protocol. In these cases it is typical to consider other notions of knowledge soundness and zero-knowledge, especially for non-interactive protocols: knowledge soundness referring to the existence of a p.p.t. extractor that can extract a witness from a proof using a trapdoor, and zero-knowledge referring to the existence of a simulator that can generate a proof without knowledge of the witness [GOS06, GS08]. In these cases, we expect the overall protocol to maintain (computational) 2-special soundness: the special-sound extractor will internally run the extractor of the non-interactive proof, and the honest-verifier zero-knowledge its simulator to produce the simulated proof transcript for the sub-proof.

Non-interactive proofs. As is common in the the literature on Σ-protocols and identification schemes, we present and analyze the interactive version of our protocol with the understanding that can be easily made non-interactive using the Fiat-Shamir (FS) transform [FS87]. In the FS transform, the prover computes $(a, \rho) \leftarrow \mathsf{Com}(\phi, w)$ as usual, then computes the challenge as $c \leftarrow \mathsf{H}(\phi \| a)$ where H is a cryptographic hash function whose image is in the codomain of Ch. The response is computed as before, and the output is (a, c, z), which can usually be compressed to (c, z) (as in our protocol). The resulting protocol is secure in the random oracle model, via the forking lemma [PS00]. Again, since the FS transform and the related analysis are well-known, we refer to Boneh and Shoup [BS20] for additional details

Table 1: Summary of notation and variables names used throughout Section 3.

p, q	Order of the groups $\mathbb{G}_p$ and $\mathbb{G}_q$
G_p, G_q	Generators of $\mathbb{G}_p$ and $\mathbb{G}_q$
H_p, H_q	Additional generators of $\mathbb{G}_p$ and $\mathbb{G}_q$, independent of G_p, G_q
x, x_p, x_q	The witness as an integer x, or a value mod p or q
b_g	bit-length of the smaller group, i.e., $b_g = \lceil \log_2(\min(p, q)) \rceil$
b_c	bit-length of the challenge c
b_x	bit-length of the witness x
b_f	Parameter controlling the probability of aborts

3 General discrete logarithm equality

Notation. Since we will have two groups in our protocol, we use the subscripts p and q to indicate that an element or scalar belongs to $\mathbb{G}_p$ or $\mathbb{G}_q$. That is, we denote by $(\mathbb{G}_p, p, G_p, H_p)$ the description of a group $\mathbb{G}_p$ of prime order p. We will often lift scalars from $\mathbb{Z}_p$ to $\mathbb{Z}$ in the canonical way, and when we say that values $x_p \in \mathbb{Z}_p$ and $x_q \in \mathbb{Z}_q$ are equal we mean they are the same as integers. In Table 1 we summarize the variable names and notation used in this work.

Protocol. Our protocol is described in Figure 1, and is parametrized on values $b_x, b_c, b_f > 0$ such that $b_x + b_c + b_f < b_g$ with $b_g \coloneqq \lceil \log_2(\min(p, q)) \rceil$. It has a similar structure to Okamoto's identification protocol [Oka93] and Chaum–Pedersen's representation proof [CP93]. The main differences are: we require a range proof to ensure that the discrete log "fits" in both groups, and the response value is computed over the integers, so that a single value is used in both groups during verification.

The verifier's input to the protocol are two Pedersen commitments $(X_p, X_q) \in \mathbb{G}_p \times \mathbb{G}_q$ committing to values $(x_p, x_q) \in \mathbb{Z}_p \times \mathbb{Z}_q$. The verifier is also given a range proof π_{rp} that the committed value x_p is in $\{0, \ldots, 2^{b_x} - 1\}$. The relation being proven can be expressed as

$$\mathsf{R}_{dleq} \coloneqq \{((X_p, X_q)\,,\ (x, r_p, r_q)) : X_p = xG_p + r_pH_p \ \wedge\ X_q = xG_q + r_qH_q\} \tag{1}$$

(where (x, r_p, r_q) is the witness) and holds under the precondition that π_{rp} is valid. Our analysis will require that the range proof be knowledge-sound, since in our analysis we need to extract the opening of the Pedersen commitment X_p from both π_{rp} and from our new protocol, to ensure that both proofs are about the same opening of X_p (which holds since Pedersen commitments are binding). In practice, π_{rp} can be realized for example with Bulletproofs [BBB+18] when G_p is a prime-order group (we discuss some options in Section 3.2).

We describe the protocol as an interactive Σ-protocol (with aborts) with the understanding that it can be directly made non-interactive with the Fiat-Shamir transform [FS87] (with aborts [Lyu09]). Provers in this class will abort the protocol with a bounded probability: intuitively, the prover will abort when providing a response would leak information about the witness. When this occurs, the prover and verifier restart the protocol from the beginning. In the non-interactive version, the prover repeats locally, and only outputs a non-aborting transcript.

Range proofs. In Figure 1 we require a range proof π_{rp}, parametrized by the group description $(\mathbb{G}_p, p, G_p, H_p)$ and the bound b_x. It proves the relation

$$\mathsf{R}_{rp} \coloneqq \left\{((x, r), X_p, b_x) : X_p = xG_p + rH_p \ \wedge\ 0 \le x < 2^{b_x}\right\}.$$

We ask the range proof to satisfy *honest verifier zero-knowledge* and *knowledge soundness*.

Parameter selection. In Table 2 we give some possible parameters when $b_g = \min(253, 255) = 253$, where the bit-lengths 253 and 255 correspond to the group orders of the Ristretto [HdVLA22] group and the BLS12-381 group [BLS03, Bow17]. We must choose parameters so that $b_x + b_c + b_f < b_g$ so that the response is an integer and no modular reduction occurs in either group. We must also choose the number of parallel repetitions τ so that $\tau \cdot b_c \ge 128$, for non-interactive security.

Performance. For τ repetitions, the size of the proof after applying the Fiat-Shamir transform and compressing the transcript into $\pi = (c, z, s_p, s_q)$ is $\tau(b_c + b_f + \lceil \log_2 p \rceil + \lceil \log_2 q \rceil)$ bits. The prover and verifier computational costs are 2τ multi-scalar multiplications (τ in each of $\mathbb{G}_p$ and $\mathbb{G}_q$, each with three terms) when there is no abort (in general the expected cost depends on b_f). Some proof size estimates are given in Table 2.

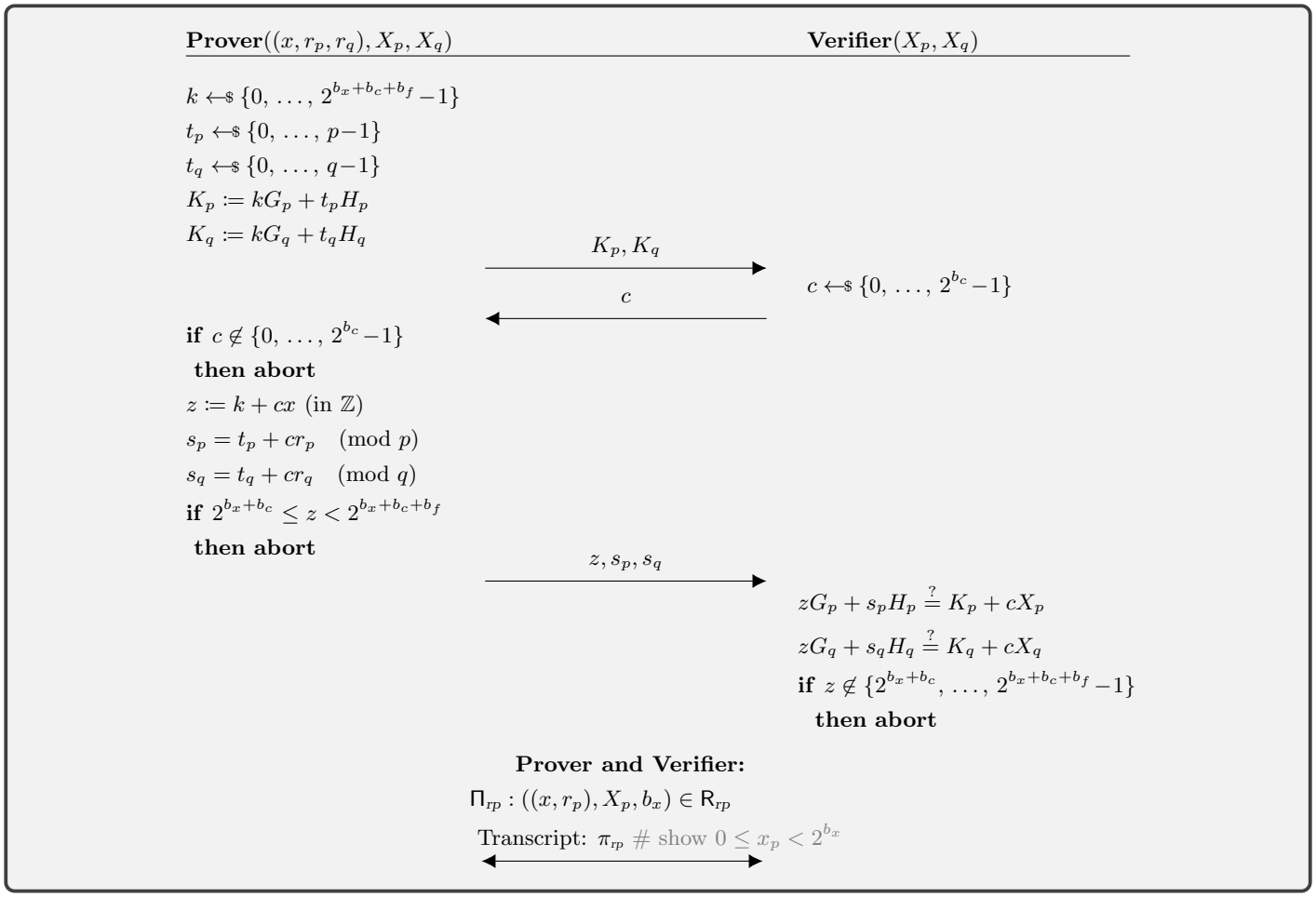

Figure 1: Protocol Π_{dleq}, a Σ-protocol for equality of committed values across groups. The input commitments are $X_p = xG_p + r_pH_p \in \mathbb{G}_p$ and $X_q = xG_q + r_qH_q \in \mathbb{G}_q$ for $0 \le x < 2^{b_x}$, $r_p \in \mathbb{Z}_p$ and $r_q \in \mathbb{Z}_q$.

3.1 Analysis

In this section, we prove Theorem 1 by showing that Π_{dleq} satisfies completeness, special soundness, and honest-verifier zero-knowledge. We introduce a lemma, which is essentially the same as [AFLT12, Lemma 1], propaedeutic for the proofs of completeness and zero-knowledge. The protocols are different, but this lemma applies almost exactly because of the way the response is computed over $\mathbb{Z}$ and the aborting condition.

Lemma 4 ([AFLT12])**.** *In an honest execution of Π_{dleq} the probability that the prover aborts is $1/2^{b_f}$. If the prover does not abort, the value z in the transcript is uniformly distributed in $\{2^{b_x+b_c}, \ldots, 2^{b_x+b_c+b_f}-1\}$.*

Proof. In the response value $z = k + cx_p$, since k and c are independent and k is distributed uniformly at random, the value z is distributed uniformly at random in the set

$$Z_0 = \{cx, cx+1, \ldots, cx + 2^{b_x+b_c+b_f}-1\} .$$

Let $Z = \{2^{b_x+b_c}, \ldots, 2^{b_x+b_c+b_f}-1\}$ be the set of responses for which the prover does not abort, and note that Z is properly contained in Z_0. The probability that $z \in Z$ is

$$|Z|/|Z_0| = \frac{2^{b_x+b_c+b_f} - 2^{b_x+b_c}}{2^{b_x+b_c+b_f}} = 1 - 1/2^{b_f}$$

and hence the probability that the prover aborts is $1/2^{b_f}$. Consider a fixed response $z_0 \in Z$, we have

$$\Pr[z = z_0 | z \in Z] = \frac{\Pr[z = z_0]}{\Pr[z \in Z]} = \frac{1/2^{b_x+b_c+b_f}}{|Z|/2^{b_x+b_c+b_f}} = \frac{1}{|Z|}$$

Table 2: Possible parameter choices for 128-bit security when $\mathbb{G}_p$ is Ristretto and $\mathbb{G}_q$ is BLS12-381. Column τ is the number of parallel repetitions; $|\pi| = \tau(b_c + b_f + \lceil \log_2 p \rceil + \lceil \log_2 q \rceil)$ is the proof size in bytes after applying the Fiat-Shamir transform and excluding the size of the range proof; all other columns are in bits.

b_c	b_x	b_f	τ	$\lvert\pi\rvert$	Notes
192	52	8	1	89	
128	112	12	1	81	Ideal for the credential linking application
64	128	60	2	158	Increase b_f since $\tau = 2$ means we can reduce b_c
64	180	8	2	145	
32	212	8	4	274	See alternative approach for large x in Section 3.2
16	228	8	8	532	See alternative approach for large x in Section 3.2

and so the response is uniformly distributed in the set of responses that do not cause the prover to abort. □

Given the above lemma, completeness is straightforward.

Theorem 5. *The protocol Π_{dleq} for the relation R_{dleq} is 2^{-b_f}-complete.*

Proof. By Lemma 4, we have that the prover aborts with probability 2^{-b_f}. When the prover does not abort, the verification equation is always satisfied, since $0 \leq c < 2^{b_c}$ and

$$zG_p + s_pH_p = (k + cx)G_p + (t_p + cr_p)H_p = (kG_p + t_pH_p) + c\,(xG_P + r_pH_p) = K_p + cX_p\,.$$

Similarly, one proves that also (ii) is satisfied. □

3.1.1 Soundness

Our soundness analysis reduces to the binding property of Pedersen commitments, and establishes the constraints on the protocol parameters b_x, b_c, and b_f.

Theorem 6. *If $b_x + b_c + b_f < \lceil \log_2(\min(p, q)) \rceil$, the protocol Π_{dleq} is computational special sound for the relation R_{dleq} with knowledge error $\epsilon = 2^{-b_c+1} + \epsilon_{rp} + \epsilon_{DL}$, where ϵ_{rp} is the knowledge error of π_{rp} and $\epsilon_{DL} = \epsilon_{DL_p} + \epsilon_{DL_q}$ is the advantage in solving the discrete logarithm problem in $\mathbb{G}_p$ or $\mathbb{G}_q$.*

Proof. We describe an extractor algorithm Ext, that on input π_{rp}, $(X_p, X_q) \in \mathbb{G}_p \times \mathbb{G}_q$, and accepting transcripts $((K_p, K_q), c, z, s)$ and $((K_p, K_q), c', z', s'_p, s'_q)$ with must recover x, r_p, r_q, such that $X_p = xG_p + r_pH_p$ and $X_q = xG_q + r_qH_q$.

Since π_{rp} is assumed to be valid, using the knowledge extractor for that proof we can extract (x^*_p, r^*_p) such that $X_p = x^*_pG_p + r^*_pH_p$ and $x^*_p < 2^{b_x}$. We denote by ϵ_{rp} the probability that the range proof extractor fails in providing a valid witness (x^*_p, r^*_p). From special soundness, we have two pairs of accepting transcripts proving knowledge of the opening of a Pedersen commitment (or, transcript of Okamoto's identification protocol [Oka93]) in $\mathbb{G}_p$ and $\mathbb{G}_q$, namely $((K_p, c, z, s_p), (K_p, c', z', s'_p))$ and $((K_q, c, z, s_q), (K_q, c', z', s'_q))$. Ext internally runs the Okamoto extractor for special soundness using the transcripts above, which succeeds with probability 2^{-b_c} (since $c \neq c'$) in producing witnesses (x_p, r_p) and (x_q, r_q), such that $X_p = x_pG + r_pH_p$ and $X_q = x_qG + r_qH_q$.[7]

Then, the extractor checks that all commitment openings are consistent between each other, and that the adversary did not manage to change the committed values in the second transcript. The extractor aborts if $(x_p, r_p) \neq (x^*_p, r^*_p)$ or $(z - cx_p, s_p - cr_p) \neq (z' - c'x_p, s'_p - c'r_p)$. Then, it makes a similar check for $\mathbb{G}_q$ too: if $(z - cx_q, s_q - cr_q) \neq (z' - c'x_q, s'_q - c'r_q)$, abort. This happens with negligible probability, by the binding property of Pedersen commitments X_p, K_p and K_q (that is, hardness of DL in $\mathbb{G}_p$ and $\mathbb{G}_q$).

Finally, the extractor returns (x_p, r_p, r_q). We must now argue that $x_p = x_q$, when seen as integers. From the verification checks (i) and (ii) we have that $\exists\, k, k', a, a', b, b' \in \mathbb{Z}$ such that

$$z = k + cx_p + ap \qquad z = k' + cx_q + bq$$
$$z' = k + c'x_p + a'p \qquad z' = k' + c'x_q + b'q$$

Note that k and k' are well-defined, since the check above establishes a single commitment opening for K_p, K_q in each pair of transcripts. The integers (a, a', b, b') are non-negative because verification checks that $2^{b_x+b_c} \leq z < 2^{b_x+b_c+b_f}$ and

[7] We stress here that we are proving *special soundness* and that therefore rewinding is not needed.

parameters are chosen such that $b_x + b_c + b_f < \lceil \log_2(\min(p,q)) \rceil$. By subtracting the responses corresponding to the mod p and mod q equations, we have

$$(z - z') = (c - c')x_p + (a - a')p \qquad (z - z') = (c - c')x_q + (b - b')q,$$

Without loss of generality, assume that $z - z'$ is positive. Since π_{rp} ensures that x_p is "small" and $|c - c'|$ is also "small", then $(a - a') = 0$. More precisely, $z - z'$ has bit-length less than $b_g \leq \lceil \log_2(p) \rceil$ by our choice of parameters (namely the constraint $b_x + b_c + b_f < b_g$), and check (iii) during verification, which ensures that $z < 2^{b_x + b_c + b_f}$.

Equating the two representations of $z - z'$, and noting that $(a - a') = 0$ we have (still over $\mathbb{Z}$)

$$\begin{aligned}(c - c')x_p &= (c - c')x_q + (b - b')q \\ (c - c')(x_p - x_q) &= (b - b')q\end{aligned}$$

Since q is prime, it must divide $(c - c')$ or $(x_p - x_q)$. But since the bit-length of q is at least b_g, and $b_g > b_c$, then q is too large to divide $|c - c'|$. Therefore $q \mid (x_p - x_q)$ which means that $x_p = x_q \pmod q$. Since x_p and x_q are equal mod q, and the bit-length of x_p is strictly less than $\lceil \log_2(q) \rceil$, it must be that $x_p = x_q$ over $\mathbb{Z}$ as well. To conclude, Ext extracts a valid witness with error $\epsilon = 2^{-b_c+1} + \epsilon_{rp} + \epsilon_{DL_p} + \epsilon_{DL_q}$. □

Parallel repetitions. The knowledge error might not be negligible depending on the choice of b_c. Generically, τ repetitions result in a knowledge error ϵ^τ, but in this case the extractor for the range proofs needs to be run only once for all repetitions, and the reductions to commitment binding can be done all at once. This means that τ repetitions of Π_{dleq} lead to a knowledge error $2^{(-b_c+1)\tau} + \epsilon_{rp} + \epsilon_{DL_p} + \epsilon_{DL_q}$.

3.1.2 Zero-knowledge

Zero-knowledge with aborts. Identification schemes where the prover may abort [Lyu09, AFLT12] are generally not honest-verifier zero-knowledge (HVZK). The challenge in proving HVZK is in simulating the prover's commitment message in aborting transcripts. However, it is often possible to prove the schemes satisfies a relaxed notion of HVZK, sometimes called no-abort honest-verifier zero-knowledge (naHVZK) [KLS18]. In naHVZK, the simulator either returns a valid transcript, or returns $\perp$ and the verifier forgets about the incomplete session made only of commitment and challenge. Since naHVZK is sufficient to simulate non-interactive proofs (or signatures) when the Fiat-Shamir transform is applied, naHVZK is still a useful notion. Our protocol in Figure 1 is not affected by this limitation: intuitively, the responses s_p, s_q, which are distributed uniformly at random in $\mathbb{Z}_p$ and $\mathbb{Z}_q$, guarantee that the commitment message is always uniformly random, both in aborting as well as succeeding transcripts. Thus, we prove standard honest-verifier zero-knowledge, and our protocol may also be used interactively.

Theorem 7. *The protocol Π_{dleq} for the relation R_{dleq} is perfectly honest-verifier zero-knowledge.*

Proof. On input c, the simulator samples z uniformly at random from $\{2^{b_x+b_c}, \ldots, 2^{b_x+b_c+b_f}-1\}$ and s_p and s_q uniformly from $\mathbb{Z}_p$ and $\mathbb{Z}_q$. Then the simulator solves for K_p, as $K_p := (zG_p + s_pH_p) - cX_p$ (similarly for K_q). With probability $1/2^{b_f}$ the simulator outputs $(K_p, K_q, c, \perp)$ (the abort case) and otherwise outputs $(K_p, K_q, c, (z, s_p, s_q))$.

We now argue that the real and simulated transcripts are identically distributed. For the prover's first message, since s_p was chosen uniformly by the simulator, then $K_p = kG_p + t_pH_p = kG_p + (s_p - zc)H_p$ is distributed uniformly at random in $\mathbb{G}_p$, regardless of whether the response is $\perp$ or (z, s_p, s_q). We note that in the abort case k will be distributed differently in real and simulated transcripts, but because K_p and K_q are perfectly hiding commitments they are identically distributed. In non-aborted transcripts, both real and simulated transcripts have uniform z value (in the given range), by Lemma 4 and (s_p, s_q) are sampled uniformly at random in both cases. The abort probability of the simulator is the same as the honest prover, by Lemma 4 honest transcripts are aborted with probability $1/2^{b_f}$ exactly as in the simulated case. □

3.1.3 Equality of simple discrete logarithms

Our protocol takes Pedersen commitments to (x_p, x_q) as input. A natural question is whether a variant of this protocol also works when the inputs are simple discrete logarithm commitments, namely $X_p = x_pG_p$ and $X_q = x_qG_q$. We investigated this question and believe it has a positive answer, subject to some technicalities and limitations. We did not formalize this section as our main motivation of linking credentials requires Pedersen commitments, so that they can be re-randomised by the credential holder before each presentation proof, in order to make repeated proofs with the same credentials unlinkable. We list some of the issues that must be addressed.

Concrete DL hardness. Our protocol allows x to be short (e.g., 64 or 128 bits), but solving for x given $X = xG$ is easier when x is short. For generic groups, the best-known attack cost (Pollard's lambda algorithm [Pol78]) is $2^{(\log_2 x)/2}$. Therefore, x must be large enough so that the DL instance is hard, and this restricts the choices available for parameter selection (cf. Table 2): in order to keep the response size below the group order, we must use smaller challenges, and this increases the number of parallel repetitions required, or the abort probability and consequently the proving time. With Pedersen commitments, instead, x is unconditionally hidden.

Cross-group DL hardness. Again, when the commitments to x are simple discrete log instances, we have to make a new hardness assumption. Namely, we must assume that given short DL instances $X_p = xG_p$ and $X_q = xG_q$ with the same x, the advantage ϵ_{DL} of finding x is as hard as the short DL in either $\mathbb{G}_p$ or $\mathbb{G}_q$. That is, $\epsilon_{DL} \leq \max(\epsilon_{DL_p}, \epsilon_{DL_q})$. This seems reasonable when x is large enough and the DL problem is hard in both $\mathbb{G}_p$ and $\mathbb{G}_q$, but is not a common cryptographic assumption as secrets are almost universally used only in one primitive, and not across groups.

Simuation of aborting transcripts. Another issue is how to (perfectly) simulate the prover's first message in aborted transcripts. In Section 3.1 we discuss how our protocol avoids this challenge (in short, the first message consists of Pedersen commitments, which are always uniformly random, independent of whether the prover aborts). For a variant of our protocol that does not use Pedersen commitments, the weaker notion of *no-abort HVZK* [KLS18] (discussed in Section 3.1.2) should be achievable, and while weaker, this notion is still sufficient for non-interactive proofs, which are suitable for the applications we consider.

3.2 Efficiency

While the asymptotic efficiency of Π_{dleq} is trivial, in this section we discuss some of the considerations for concretely realizing and implementing it.

Handling larger values. One limitation of the protocol presented above is that it requires that x be b_x bits or fewer, and b_x cannot be as large as the group order (of the smaller group). Here we describe how to address this, by breaking x into chunks and proving the relation on each chunk using Π_{dleq}. Let C_p and C_q be commitments to the same value $x \in \{0, \ldots, \min(p, q)-1\}$, and suppose b_x has been chosen subject to the constraints given above. Define $\ell := \lceil (\log_2 x)/b_x \rceil$. Denote by $(x^{(0)}, \ldots, x^{(\ell-1)})$ the representation of x in base 2^{b_x} that is, $x = \sum_{i=0}^{\ell-1} 2^{i \cdot b_x} x^{(i)}$. Sample random $r_p^{(i)}$ such that $r_p = \sum_i 2^{i \cdot b_x} r_p^{(i)} \pmod p$. Construct the commitments $C_p^{(0)}, \ldots, C_p^{(\ell-1)}$ as $C^{(i)} := x^{(i)} G_p + r_p^{(i)} H_p$. Proceed in the same way for C_q. They satisfy

$$C_p = \sum_{i=0}^{\ell-1} 2^{i \cdot b_x} C_p^{(i)} \qquad C_q = \sum_{i=0}^{\ell-1} 2^{i \cdot b_x} C_q^{(i)} \tag{2}$$

The prover sends $C_p^{(0)}, \ldots C_p^{(\ell-1)}$ and $C_q^{(0)}, \ldots, C_q^{(\ell-1)}$, along with ℓ range proofs,[8] to prove that each $x^{(i)} \in \{0, \ldots, 2^{b_x}-1\}$. Then the prover and verifier invoke the protocol in Figure 1 for each $i \in \{0, \ldots, \ell-1\}$ to prove that $C_p^{(i)}$ and $C_q^{(i)}$ commit to the same short value. The verifier additionally checks Equation (2) holds.

Range proofs. Range proofs may not be necessary if the application provides assurance that x is in the correct range. For example, in the credential linking application, we can trust that the issuer only issues credentials with a valid x. In systems using keyed-verification anonymous credentials [CMZ14], this is especially reasonable since the issuer and verifier are the same party. When the credential is presented in order to produce X_p, our soundness analysis of Theorem 6 can be modified to extract x from the presentation proof, rather than the range proof.

When a range proof is necessary, Bulletproofs [BCC+16, BBB+18] give a practical solution. For example, creating range proofs for 64-bit values using the Rust crate `bulletproofs` from the Dalek project [dVYA], the prover time is about 7.3 ms, the verifier time is 1 ms and the proof size is 672 bytes. See [dVYA] for details of the benchmark platform, and benchmarks of other libraries offering range proofs.The library does not support 128-bit ranges, but we expect prover and verifier times to roughly double, and the range proof size to increase to 704 bytes.We also note that when using the strategy given above that breaks x into ℓ pieces, range proofs for each of the pieces can be grouped together into a single

[8] It is not secure to send a single range proof for x instead of ℓ proofs for each $x_p^{(i)}$. Consider commitments C_p, C_q to different values $x_p < p$ and $x_q < q$, and π_{rp} a range proof of C_p with witness x_p. By the Chinese remainder theorem there exists a unique integer $x < pq$ such that $x_p = x \pmod p$ and $x_q = x \pmod q$. An attacker is able to freely choose ℓ values $x_p^{(0)}, \ldots x^{(\ell-1)}$ larger than b_x such that $\sum_i 2^{ib_x} x_i = x$, and commit to them both in $\mathbb{G}_p$ and $\mathbb{G}_q$, passing the verification procedure.

proof, which will be shorter than ℓ individual proofs, for example, a proof of four 64-bit ranges is only 800 bytes (but prover and verifier times are only slightly better than four individual proofs).

When one of the groups is a pairing-based group, one could alternatively do π_{rp} in that group using a zkSNARK with constant size and concretely very short proofs, e.g., [Gro16]. Since our analysis requires a range proof for x in *either one of* $\mathbb{G}_p$ or $\mathbb{G}_q$, applications can choose to implement the range proof in the group that offers better performance.

Constant-time implementation. Depending on the abort probability $1/2^{b_f}$, implementations may leak the number of times the protocol was aborted, since the prover's time is directly proportional to the number of aborts (in a direct implementation). If the number of aborts depends on the secret, this would be sensitive information. However, from Lemma 4 we can see that the abort probability is the same for any secret, and therefore independent of the secret. Therefore, it is not required that implementations attempt to hide the number of aborts that occur when generating a proof.

Parallel repetition. In Theorem 6 it is shown that Π_{dleq} has knowledge error 2^{-b_c} and because of our constraints on parameter selection, b_c may not be as large as the security parameter λ, so the soundness error may be non-negligible. In practice, assuming the hash function of the FS transform has output bit-length $b_c\tau$, the challenges are obtained by considering each of the τ chunks of b_c bits.[9] The well-known approach to boost soundness of the protocol is to repeat it τ times in parallel, so that the soundness error is $2^{-b_c\tau}$ such that $b_c\tau \geq \lambda$. Note that we exclude π_{rp} from the parallel repetitions, since we consider it to be part of the input statement and have negligible soundness error. We also require that none of the τ repetitions abort, which increases the abort probability from $1/2^{b_c}$ to $\tau/2^{b_c}$, so to hold the abort probability constant b_f should be increased by $\lceil \log_2(\tau) \rceil$. Since Π_{dleq} is zero-knowledge, it is also witness hiding [FS90, Theorem 3] and therefore parallel composition is also witness hiding (at least witness hiding; we expect Theorem 7 can be generalized to handle parallel repetition).

Denote the challenge with parallel repetition as $c = (c_1, \ldots, c_\tau)$. Special soundness provides two transcripts with $c \neq c'$, and when both transcripts are different everywhere, that is $c_i \neq c'_i$ for $i \in [\tau]$, we have τ transcripts where Ext succeeds with probability 2^{-b_c}. Therefore soundness is boosted as expected to $2^{-\tau b_c}$ in this case. More generally, when c is chosen at random (either by an honest verifier or a hash function) there may be a small loss in concrete soundness, since some c_i may be equal. While this loss is in our analysis, we do not know of an attack matching it.

When there are multiple instances of the protocol, like in the variant described above where the protocol is run ℓ times, the witnesses are independent and protocols may be run in parallel. As a minor optimization, each of the ℓ instances may share the same random challenge from the verifier.

Ignoring aborts safely? For some secret lengths and group sizes, it is possible to choose parameters such that the abort probability 2^{-b_f} is statistically negligible. In such cases an implementation that ignores aborts will leak a small amount of information occasionally. For example, suppose we have $b_x = 128$, $b_c = 64$, $b_g = 253$ and $b_f = 60$. Then we expect one in 2^{60} proofs to output a "leaky" response; a response that would have caused an abort, but that we output anyway. In the case of Schnorr and ECDSA signatures slight biases in the nonce appear to be difficult to exploit, see e.g., [ANT+20]. However, since our setting is somewhat different and we do not have a detailed analysis we do not recommend ignoring aborts, but encourage future work on this question. Avoiding the abort path simplifies writing and testing of implementations.

4 Trading group operations for hash evaluations

Our protocol for trading elliptic curve group operations for hash evaluations is described in Figure 2. Informally, it is parametrized by a linear morphism $M \in \mathbb{G}^{m \times n}$ denoting the linear relation to be proven. Valid choices include $M = [G]$ for discrete logarithm relations $xG = X$, or $M = [G, H]$ for Pedersen commitments $[G, H] \cdot [x_0, x_1]^t = x_0 G + x_1 H$, but at the core it should be hard to find $\vec{x}, \vec{x}'$ such that $M\vec{x} = M\vec{x}'$. The verifier's inputs are $(\vec{X}, x_h)$, respectively commitment and hash of the same value $\vec{x}$. The relation being proven can be expressed as

$$\mathsf{R}_{dlhash} \coloneqq \{((\vec{x}), x_h, \vec{X}) : \vec{X} = M\vec{x} \ \wedge \ x_h = \mathsf{H}(\vec{x})\}.$$

which is (implicitly) parametrized on the group and the matrix distribution from which M is selected.

[9] In particular, we ask not to use the same hash function for each repetition to mitigate grinding attacks, also known in the literature as precomputation attacks.

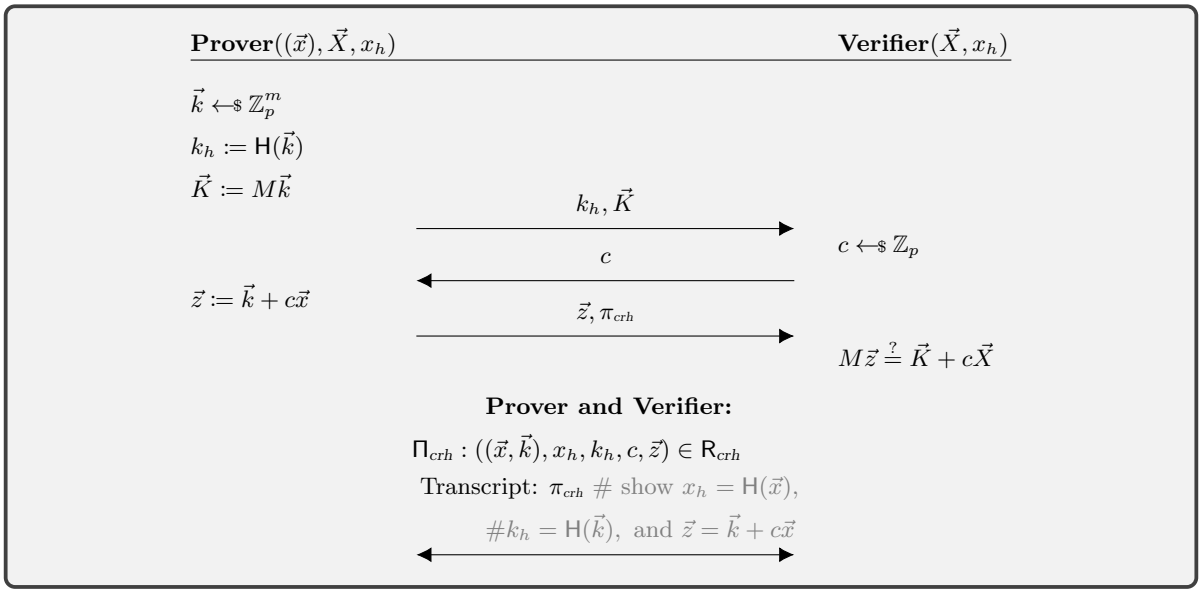

Figure 2: Protocol Π_{dlhash}, a Σ-protocol for proving knowledge of $\vec{x}$ such that $\vec{X} = M\vec{x}$ and $x_h = \mathsf{H}(\vec{x})$.

Proofs for CRH. Π_{dlhash} requires at the end a proof for the pre-image of a collision-resistant hash function H, parametrized by the field $\mathbb{Z}_p$. More precisely, we assume the existence of a proof for the relation

$$\mathsf{R}_{crh} := \{((\vec{x}, \vec{k}), x_h, k_h, c, \vec{z}) : x_h = \mathsf{H}(\vec{x}) \ \wedge \ k_h = \mathsf{H}(\vec{k}) \ \wedge \ \vec{z} = \vec{k} + c\vec{x}\}$$

This protocol can be instantiated (for instance) using our Π_{aes} from Section 5, taking particular care in tweaking the block size to be large enough in order to provide sufficient collision resistance,[10] but this proof can of course be provided with any other general-purpose proof system for collision-resistant hash functions, algebraic or boolean.

Collision resistance. Roughly speaking, a function $f : \mathcal{X} \to \mathcal{Y}$ is *collision-resistant* if no p.p.t. adversary $\mathcal{A}$ given as input f can produce $\vec{x}, \vec{x}' \in \mathcal{X}$ such that $f(\vec{x}) = f(\vec{x}')$. We require this property to hold both for H as well as for the linear map associated to M. While the first is a standard notion in cryptography [KL07], in the context of linear function this problem reduces to finding non-trivial elements of the kernel of M, which has been formulated in the past by Morillo, Ràfols, and Villar [MRV16, Def. 13]. For the concrete examples that we gave above and we will study, the above assumption reduces to the hardness of the DL on GrGen.

Definition 8 (Kernel-Matrix Diffie-Hellman [MRV16])**.** *KMDH is hard for a group generator* GrGen *and a matrix distribution* D *if it is infeasible, given a group description* $\Gamma := (\mathbb{G}, p, G) \leftarrow \mathsf{GrGen}(1^\lambda)$ *and a matrix* $M \leftarrow \mathsf{D}(\Gamma)$ *in* $\mathbb{G}^{n \times m}$ *to find non-trivial elements of the null space, that is, to exhibit an* $\vec{x} \in \mathbb{Z}_p^n$ *such that* $M\vec{x} = 0$ *and* $\vec{x} \neq 0$*.*

We will say that KMDH is hard in $\mathbb{G}$ for a matrix M if we consider the distribution D to be the distribution of matrices M parametrized solely by the group description output of GrGen. The simplest examples of the above is the group mapping $M = [G]$, which is into and thus perfectly collision resistant. Another valid example are Pedersen commitments, for which $M = [G, H]$ $(m = 1, n = 2)$ and the binding property follows straightforwardly from hardness of DL in $\mathbb{G}$.

Compatibility. In order to provide HVZK, the hash function H must be compatible with the group GrGen, in the sense that it should be computationally hard to distinguish the pair $(M\vec{x}, \mathsf{H}(\vec{x}))$ from the pair $(M\vec{x}', \mathsf{H}(\vec{x}'))$ for $\vec{x} \neq \vec{x}'$. We call this notion *hiding-compatibility.*

Definition 9. *Let* $f = \{f_\lambda\}_\lambda, h = \{h_\lambda\}_\lambda$ *be two function families indexed in* $\lambda \in \mathbb{N}$ *with domain* $\mathcal{X}_\lambda$*. We say that* (f, h) *are* hiding-compatible *if for all p.p.t. (in* λ*) adversaries* $\mathcal{A}$*, the distributions*

$$\{x \leftarrow\$ \mathcal{X}_\lambda : (f_\lambda(x), h_\lambda(x))\} \qquad \textit{and} \qquad \{x, s \leftarrow\$ \mathcal{X}_\lambda : (f_\lambda(x), h_\lambda(s))\}$$

are distinguishable with probability negligible in λ*.*

[10] A secure hash mode for AES, derivative of the Davies-Meyer construction, has been proposed in https://csrc.nist.rip/groups/ST/toolkit/BCM/documents/proposedmodes/aes-hash/aeshash.pdf.

The above assumption can be considered valid for most practical applications when setting f to be a linear elliptic-curve morphism, and h a one-way function, but seems hard to prove in the standard model without a joint computational assumption. More specifically, it's difficult to prove that for instance, given a hash function $h : \{0,1\}^{2\lambda} \mapsto \{0,1\}^{2\lambda}$, there does not exist an elliptic curve such that $h(x) = xG = f(x)$ where G is the generator resulted from $\mathsf{GrGen}(1^\lambda)$. Yet, the above definition trivially holds in the generic group model, if h is a one-way hash function, and in the random oracle model. Another valid case would be picking $f(\vec{x}) = x_0G_1 + x_1G_2$ and $h(\vec{x}) = x_0G_3 + x_1G_4$, with $G_1, \ldots, G_4$ independent (i.e., *NUMS*, cf. Section 2) generators where hiding-compatibility holds under the DL assumption.

Batching. When batching multiple Π_{dlhash} instantiations, one can trivially extend the matrix relying on standard parallel execution of Σ-protocols. A single separate proof for R_{crh} may be used if the vector is the same. The proof π_{crh} may be integrated within a more proof statement. For instance, when proving knowledge of a secret key x such that $X = xG$ and membership in a Merkle tree of the given public key, our soundness analysis of Theorem 10 can be adapted to extract the witness x from the Merkle proof.

4.1 Analysis

In this section, we prove Theorem 2 by showing that Π_{dlhash} satisfies completeness, special soundness, and honest-verifier zero-knowledge. In fact, we focus on special soundness and zero-knowledge, as completeness is straightforward: the response $\vec{z} = \vec{k} + c\vec{x}$ satisfies $M\vec{z} = M\vec{k} + c(M\vec{x}) = \vec{K} + c\vec{X}$ by linearity of M, and validity of the proof π_{crh} relies on completeness of the underlying proof for the relation R_{crh}. We then discuss the concrete efficiency of our protocol when instantiated for simple discrete logarithm relations.

4.1.1 Soundness

Our soundness analysis assumes that the sub-proof π_{crh} is a knowledge-sound proof for the relation R_{crh}, collision-resistance of H, and that M is "binding" (i.e., KMDH is hard for M).

Theorem 10. *The protocol Π_{dlhash} is special sound for the relation R_{dlhash} with knowledge error $\epsilon = \epsilon_{crh} + \epsilon_{kmdh}$, where ϵ_{crh} is the knowledge error of the sub-proof π_{crh} and ϵ_{kmdh} is the hardness of KMDH in $\mathbb{G}$ for M.*

Proof. We prove computational special soundness extracting $\vec{x}$ such that $\vec{X} = M\vec{x}$ and $h_x = \mathsf{H}(\vec{x})$. Consider a p.p.t. adversary that outputs two valid transcripts

$$(\vec{K}, k_h, c_0, \vec{z}_0, \pi_{h,0}) \quad \text{and} \quad (\vec{K}, k_h, c_1, \vec{z}_1, \pi_{h,1}) \ , \tag{3}$$

with $c_0 \neq c_1$. For $b = 0, 1$ recover the witness $(\vec{x}_b, \vec{k}_b)$ from the knowledge-sound proof $\pi_{h,b}$. We claim that $\vec{x}_0$ is the witness, and now argue that it is indeed valid. Observe that (by definition of knowledge soundness), extracted $\vec{x}_0, \vec{x}_1, \vec{k}_0, \vec{k}_1$ satisfy

$$\begin{aligned} \mathsf{H}(\vec{x}_0) = \mathsf{H}(\vec{x}_1) = x_h \ , \qquad & \vec{k}_0 = \vec{z}_0 - c_0\vec{x}_0 \pmod p \ , \\ \mathsf{H}(\vec{k}_0) = \mathsf{H}(\vec{k}_1) = k_h \ , \qquad & \vec{k}_1 = \vec{z}_1 - c_1\vec{x}_1 \pmod p \ . \end{aligned} \tag{4}$$

The probability that the witnesses are not valid is bounded by ϵ_{crh}, the knowledge error of π_{crh}. If $\vec{k}_0 \neq \vec{k}_1$ or $\vec{x}_0 \neq \vec{x}_1$, we found a break for collision-resistance of H. Since the two transcripts of Equation (3) are valid, define $\vec{x} \coloneqq (c_0 - c_1)^{-1}(\vec{z}_0 - \vec{z}_1)$ satisfying

$$\vec{K} = M\vec{z}_0 - c_0M\vec{x} = M\vec{z}_1 - c_1M\vec{x} \ .$$

(Note $c_0 \neq c_1$ so the inverse always exists.) Since $\vec{k}_0 = \vec{k}_1$ and $\vec{x}_0 = \vec{x}_1$, we have

$$\begin{aligned} \vec{z}_0 - c_0\vec{x}_0 &= \vec{z}_1 - c_1\vec{x}_0 \pmod p \\ M\vec{z}_0 - c_0M\vec{x} &= M\vec{z}_1 - c_1M\vec{x} \end{aligned}$$

If $\vec{x}_0 \neq \vec{x}$, we have a non-trivial element of the kernel of M since $(c_0(\vec{x} - \vec{x}_0), c_1(\vec{x} - \vec{x}_0))$ are different ($c_0 \neq c_1$) and have the same image under M. Therefore, $x_0 = x = x_1$ and hence $M\vec{x}_0 = X$. In addition, from Equation (4), $\mathsf{H}(\vec{x}_0) = x_h$. Thus, $(\vec{x}_0, \vec{X}, x_h) \in \mathsf{R}_{dlhash}$ with knowledge error $\epsilon_{crh} + \epsilon_{kmdh}$. □

Table 3: The protocol Π_{dlhash} vs non-native scalar multiplication inside a Groth16 [Gro16] circuit ("Naive"). The hash function used is Poseidon [GKR+21] and the proof size is in bytes after applying the Fiat-Shamir transform, for two popular choices of elliptic curves. Benchmarks on a laptop equipped with an Intel i7-1370P CPU and 32GB of RAM running Debian Linux.

	$\lvert\pi\rvert$	Prover time	R1CS constraints
Π_{dlhash} (BN254)	256	20ms	325
Naive (BN254)	128	10.1s	1.7 million
Π_{dlhash} (BLS12-381)	336	21ms	325
Naive (BLS12-381)	192	17.6s	2.5 million

4.1.2 Zero-knowledge

Similarly to the case of soundness, in the statement below we assume that the proof π_{crh} is zero-knowledge. More information about the zero-knowledge property of π_{crh} can be found in Section 2.

Theorem 11. *If (M, H) are hiding-compatible, then the Π_{dlhash} is computationally honest-verifier zero-knowledge.*

Proof. The zero-knowledge simulator samples $c, \vec{z}, \vec{k}^* \leftarrow\$ \mathbb{Z}_p \times \mathbb{Z}_p^n \times \mathbb{Z}_p^n$ and computes $\vec{R} \coloneqq M\vec{z} - c\vec{X}$, and $r_h \coloneqq \mathsf{H}(\vec{k}^*)$. Then, simulates π_{crh} for the statement $(\tau, (\vec{x}, \vec{k}))$ and returns the transcript $(\vec{R}, r_h, c, \vec{z}, \pi_{crh})$ We show that it is difficult for an adversary to distinguish simulated transcripts from genuine transcripts generated by an honest prover via a hybrid argument on the distribution of prover transcripts:

- $\mathbf{H}_1$ An honestly-generated prover transcript is a tuple $(\vec{K}, k_h, c, \vec{z}, \pi)$ where $\vec{K} = M\vec{k}$ for some $\vec{k}$ uniformly distributed and $k_h \coloneqq \mathsf{H}(\vec{k})$, $\vec{z} = c\vec{x} + \vec{k}$. The proof π is honestly generated for $((\vec{x}, \vec{k}), (x_h, k_h)) \in \mathsf{R}_{crh}$.
- $\mathbf{H}_2$ This game behaves identically to the previous except that π_{crh} is now computed using the simulator for the statement $(x_h, k_h, c, \vec{z})$. The two distributions are indistinguishable by zero-knowledge of π_{crh}.
- $\mathbf{H}_3$ Replace the computation of k_h: instead of honestly computing it via $k_h \coloneqq \mathsf{H}(\vec{k})$, sample $\vec{k}^* \leftarrow\$ \mathbb{Z}_p^n$ and compute $k_h \coloneqq \mathsf{H}(\vec{k}^*)$. This follows directly from hiding-compatibility of (M, H).
- $\mathbf{H}_4$ Compute the elements $(\vec{K}, c, \vec{z})$ differently: instead of computing $\vec{K} = M\vec{k}$ and $\vec{z} \coloneqq \vec{k} + c\vec{x}$ for some uniformly distributed $c \in \mathbb{Z}_p$ and $\vec{k}$, we sample $c, \vec{z} \leftarrow\$ \mathbb{Z}_p \times \mathbb{Z}_p^n$ and compute $\vec{K} = M\vec{z} + c\vec{x}$. The two distributions are both uniformly distributed satisfying the relation $\vec{K} = M\vec{z} + c\vec{x}$ and perfectly indistinguishable. (The adversary cannot see the order in which values are sampled).

The simulated transcript is exactly the distribution output of the simulator. Therefore, the protocol Π_{dlhash} is honest-verifier zero-knowledge. □

4.2 Efficiency

Let $|\mathsf{H}|$ denote the size of the output of the hash function H and $|\pi_{crh}|$ the size of the proof π_{crh}. Then, for a linear relation $M \in \mathbb{G}^{n \times m}$, the prover time and proof size of Π_{dlhash} will scale linearly with the size of the witness: the proof size will be $|\pi| = (n+1)|\mathbb{F}| + |\mathsf{H}| + |\pi_{crh}|$ after applying the Fiat-Shamir transform, in *short form* (as challenge, response pair), and the prover time will be dominated by m multi-scalar multiplications of size n (that can be optimized depending on the structure and sparsity of the matrix M), plus the cost for computing the sub-proof π_{crh}.

Concrete efficiency. We instantiated the proof system for simple DL relations, using $M = [G]$, Poseidon [GKR+21] as the CRH function H, and π_{crh} using R1CS and Groth16 [Gro16]. In Table 3 we benchmark the performance of Π_{dlhash} against the naive approach of performing a non-native scalar multiplication xG using the double-and-add algorithm inside a SNARK circuit, for an $x \in \mathbb{Z}_p$ of 255 bits. Roughly speaking, for R1CS-based proof systems we note that deferring the scalar multiplication outside the circuit drastically reduces the number of constraints of about 4 orders of magnitude (We expect proof systems based on custom gates such as Plonk to have a lower gap.). The proof size overhead of Π_{dlhash} is due to the size of the additional Sigma transcript, which is large compared to the constant-sized Groth16 proof; the overhead ratio would be smaller if a less succinct proof system was used (e.g. Plonk or a log-sized proof system).

5 Rijndael via one lookup

The Rijndael cryptosystem [DR91] is a symmetric encryption algorithm, established as Advanced Encryption Standard (AES) by the U.S. National Institute of Standards and Technology (NIST) in 2001 [AES01], which has become a widely-used standard for securing electronic data globally.

Notation. The Rijndael cipher uses the following low-level operations: (1) the XOR operation, denoted with infix notation as the map $\oplus : \mathbb{F}_2^8 \times \mathbb{F}_2^8 \to \mathbb{F}_2^8 : (a, b) \mapsto a + b$; (2) multiplication by $\{2\}$ in Rijndael's Galois field $\mathsf{rj2} : \mathbb{F}_2^8 \to \mathbb{F}_2^8 : a \mapsto \alpha \cdot a$, where $\alpha^2 = 1$ is a non-trivial 2-root of unity the field $\mathbb{F}_2^8$; (3) the S-Box operation, denoted $\mathsf{sbox} : \mathbb{F}_2^8 \to \mathbb{F}_2^8 : a \mapsto a^{-1}$ if $a \neq 0$ otherwise 0. In the following we will sometimes consider component-wise applications of the above functions. In those cases, we will write $st' := \mathsf{sbox}(st)$ to denote the application of the S-Box operation to each element of the state st. The AES algorithm (denoted AES) consists of two parts: the key schedule (denoted $\mathsf{AES.Ksch}$) and cipher (denoted $\mathsf{AES.Cphr}$).

We denote by $(\mathbb{G}, p, \vec{G}, H)$ the description of a group $\mathbb{G}$ of order p, with independent (NUMS, cf. Section 2) generators $\vec{G} = (G_1, \ldots, G_n)$ and H (used to indicate the zero-knowledge elements). Pedersen commitments are denoted in uppercase, with randomness usually denoted in the respective greek letter, that is: $X = \sum_i x_i G + \chi H$ is a commitment to $\vec{x} \in \mathbb{Z}_p^n$ with opening information $\chi \in \mathbb{Z}_p$.

Our protocol reduces to so-called *lookup arguments*. A lookup argument takes as input a fixed table $\vec{t}$ of precomputed values. A prover can demonstrate to a verifier that a committed vector $\vec{f}$ contains only values in $\vec{t}$. In the remainder of this work, we will abuse notation and write $\vec{f} \subset \vec{t}$ to indicate $\forall i \in [1, n], f_i \in \{t_j\}_{j=1}^{|\vec{t}|}$. Informally, we will call $\vec{f}$ the *needles* and $\vec{t}$ the *haystack*.

3-Special soundness. In this section we will consider a (common) relaxation of special soundness, called 3-special soundness. Differently from classical special soundness used in Σ-protocols (cf. Section 2), in a k-special-sound protocol the extractor receives 3 transcripts, with the same commitment, but with all different challenges. Additionally, when the proof π_{lup} is instantiated using Π_{tsp} from Appendix B, the overall protocol will be $(3, \ldots, 3, 2)$-special sound (see Section 2) for an additional discussion about the soundness of composed protocols.

Lookups over structured tables. Boolean functions $B : \mathbb{F}_2^n \to \mathbb{F}_2^n$ where each component is evaluated independently, i.e. $B(\vec{v}) = (b(v_0), \ldots, b(v_n))$ for $b : \mathbb{F}_2 \to \mathbb{F}_2$ can (naïvely) be seen as a lookup table of size $N := 2^n$. To optimize the concrete efficiency of our protocol, for such functions we instead perform c lookups over tables of size $\sqrt[c]{N}$. This comes at the cost of committing to a larger vector of size $c \cdot N$. We delve into the concrete efficiency trade-offs in Section 5.2. Similar techniques have been already used by zCash [zca] and in Lasso [STW23].

Witness generation. Prover and verifier see the AES encryption as a circuit of three low level functions: $\oplus$, $\mathsf{rj2}$, sbox. The witness generation step, preceding the actual proof, consists into generating a vector $\vec{w}$ containing the concatenation of all intermediate computation on the AES state, grouped in bit-segments (in the implementation it will be grouped as 4-bit segments, but in Appendix C we study them over 8-bit segments for simplicity). Permutations of the state are not needed as they will be performed by the (linear-time) verifier. We denote with $(\vec{tr}, \vec{ksch}) := \mathsf{AesTrace}(\vec{m}, \vec{k})$ running the algorithms in Appendix C, then returning them as a pair. The prover computes the *witness vector* by constructing a vector $\vec{w} = (\vec{tr} \| \vec{ksch})$ that contains the concatenation of all intermediate computations. The final state is omitted from $\vec{tr}$ as it's equal to the ciphertext ctx, which can be added by the verifier itself.

Let us consider, as an example, the witness generation phase for the AES-128 cipher. The cipher takes as input the message $\vec{m}$ and the round keys $\vec{rk}$, incrementally defines the intermediate state values $st_{i,j}$ over $i = 0, \ldots, 10$ rounds, and returns their concatenation as the execution trace $\vec{tr}$. In each round, the state is altered across (some of the) sub-procedures $\mathsf{SubBytes}$, $\mathsf{ShiftRows}$, $\mathsf{MixColumns}$, and $\mathsf{AddRoundKey}$: $\mathsf{SubBytes}$ consists solely of one application of sbox, $\mathsf{ShiftRows}$ is a permutation, $\mathsf{MixColumns}$ is a linear transformation over $\mathbb{F}_{2^8}$ and as such can be written as $\oplus$ and $\mathsf{rj2}$ applications, and $\mathsf{AddRoundKey}$ is a simple XOR operation. In other words, the cipher circuit induces the following matrices and equations that hold for a valid cipher trace $\vec{tr}$:

- $S_{xor,L}, S_{xor,R}, S_{xor,O}$: select of the left inputs, right inputs, and outputs of XOR. We have that

$$S_{xor,L} \cdot \vec{tr} \ \oplus \ S_{xor,R} \cdot \vec{tr} = S_{xor,O} \cdot \vec{tr} \tag{5}$$

  if and only if all $\oplus$ operations over the AES trace are computed correctly.

- $S_{rj2,I}, S_{rj2,O}$: select the inputs and outputs of multiplication by $\{2\}$ in Rijndael's field. We have that

$$\mathsf{rj2}\left(S_{rj2,I} \cdot \vec{tr}\right) = S_{rj2,O} \cdot \vec{tr} \tag{6}$$

  if and only if all $\mathsf{rj2}$ operations over the AES trace are computed correctly.

- $S_{sbox,I}, S_{sbox,O}$: Matrices chosen by the verifier to select the inputs and outputs of the S-Box. We have that

$$\mathsf{sbox}\left(S_{sbox,I} \cdot \vec{tr}\right) = S_{sbox,O} \cdot \vec{tr} \tag{7}$$

  if and only if all sbox operations over the AES trace are computed correctly.

For instance, consider the example trace $\vec{tr} = (3, \dots, 3, \dots, 0, \dots)$ displaying the first element of the message, the first element of the state, and the first element of the AES key. The first row of the matrices $S_{xor,L}, S_{xor,R}, S_{xor,O}$, enforcing the 0-th AES round, will be all zeros except for the displayed positions, which will be respectively 1 for the left, right, and output XOR gates.

Protocol. The protocol is illustrated in Figure 3 and is similar to a batch lookup protocol. All core AES operations displayed can be treated as lookup evaluations: instead of checking Equations (5) to (7) directly, the verifier sends challenges $c_{\text{xor}}, c_{\text{rj2}}, c_{\text{sbox}}$ and we check that:

- the XOR constraints are all valid, with

$$S_{xor,L} \cdot \vec{tr} + c_{\text{xor}} S_{xor,R} \cdot \vec{tr} + c_{\text{xor}}^2 S_{xor,O} \cdot \vec{tr} \quad \subset \quad \vec{t}_{\text{xor}} \coloneqq \left\{ i + c_{\text{xor}} j + c_{\text{xor}}^2 \cdot (i \oplus j) \right\}_i \tag{8}$$

- the multiplications by $\{2\}$ in Rijndael's field are all valid, with

$$S_{rj2,I} \cdot \vec{tr} + c_{\text{rj2}} S_{rj2,O} \cdot \vec{tr} \quad \subset \quad \vec{t}_{rj2} \coloneqq \left\{ i + c_{\text{rj2}} \cdot \mathsf{rj2}(i) \right\}_i \tag{9}$$

- the S-Box is applied correctly, with

$$S_{sbox,I} \cdot \vec{tr} + c_{\text{sbox}} S_{sbox,O} \quad \subset \quad \vec{t}_{sbox} \coloneqq \left\{ i + c_{\text{sbox}} \cdot \mathsf{sbox}(i) \right\}_i \tag{10}$$

In other words, upon receiving a challenge c_{sbox}, the prover computes $x + c_{\text{sbox}} y$ and proves that it is contained in $i + c_{\text{sbox}} \cdot \mathsf{sbox}(i)$ (for all possible i's). We proceed similarly for the other operations, and perform a single check across all tables by concatenating needles and haystack.

The main differences are that lookup tables for boolean operations (e.g. the XOR operations) have a particular structure that allows the splitting of the table into smaller ones (in the case of 8-bit XOR, we can for instance consider two lookups of size 2^8 instead of a single one of size 2^{16}). Range-checks and shuffles are implicitly done within the lookup protocol, leveraging the small table size: overall, in the case of AES, we have a single table of size $3 \cdot 2^8 = 768$ elements. The relation being proven can be expressed as

$$\mathsf{R}_{aes} \coloneqq \left\{ \left((\vec{m}, \mu, \vec{k}, \kappa), ctx, M, K \right) : \; M = \sum_i m_i G_i + \mu H_i \;\wedge\; K = \sum_i k_i G_i + \kappa H_i \;\wedge\; ctx = \mathsf{AES}(\vec{k}, \vec{m}) \right\},$$

where $\vec{m}, \vec{k}$ are the bit-strings of (respectively) message and key for the AES block cipher, seen as small integers that are easier to store in lookup tables. For instance, in our implementation, for AES-128 and AES-256 we represent the key and the message split into 4-bit segments, i.e. $\vec{m} = (m_0, \dots, m_{31})$ with $0 \leq m_i < 16$ for all i's. If values are too large then decomposition will fail.[11] Overall, the protocol simply boils down to generating a witness vector of all intermediate computation results and looking up the elements of the computation trace in a table of 768 elements. In AES-128 the computation trace consists of 1232 elements and looks up 1808 elements in a table of 768; in AES-256 the computation trace consists of 1744 elements and 2576 elements to look up in a table of 768.

[11] From a practical perspective, this representation of the message does not change end-user applications: proving equality of messages represented differently boils doing to proving a simple Σ-protocol for proofs of representation. For instance, the linear map $\mathbf{I}_{16} \otimes (1, 2^4)$ maps the 4-bit message encoding into the 8-bit message encoding ($\mathbf{I}_n$ is the $n \times n$ identity matrix).

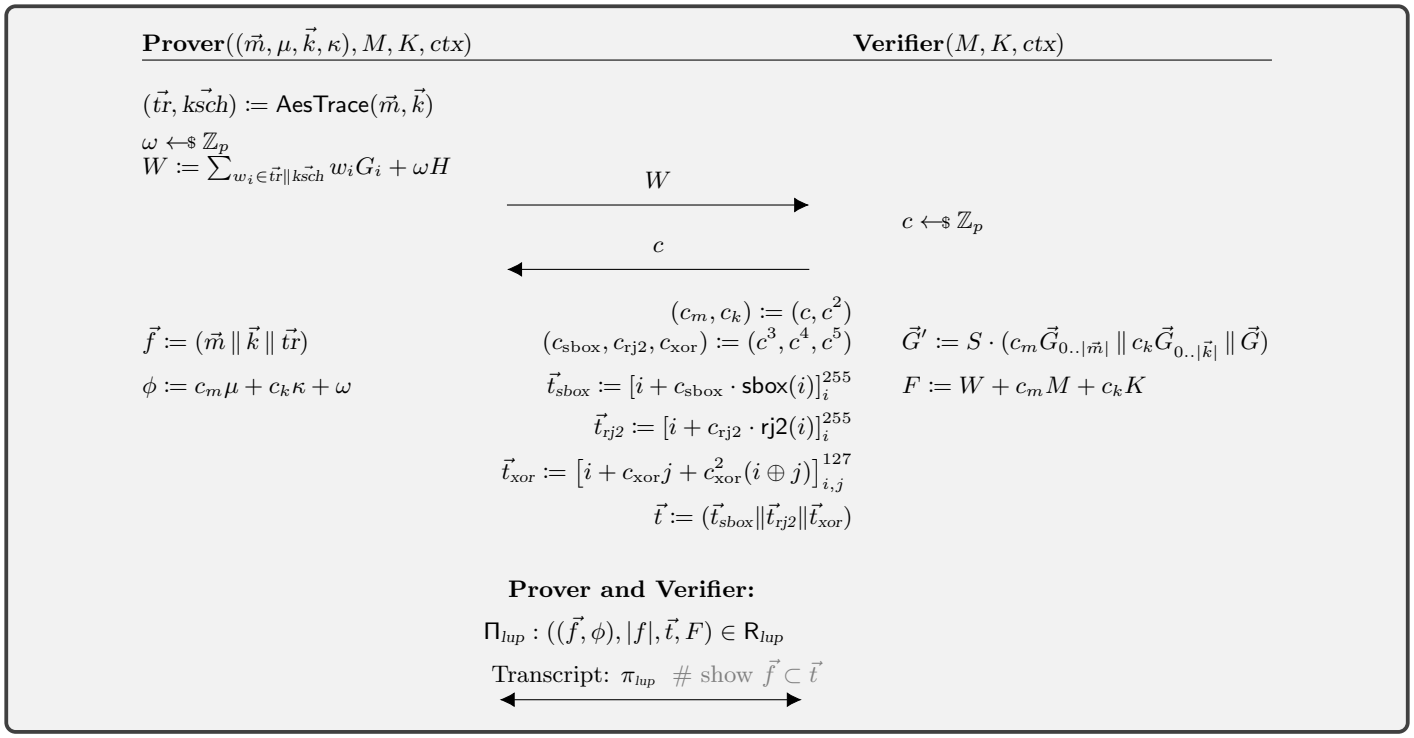

Figure 3: Π_{aes} for proving that ctx is the correct AES-encryption of message $\vec{m}$ with key $\vec{k}$ committed as M, K; the matrix S is defined in Equation (11).

Lookup protocol. In Figure 3 we require a lookup proof π_{lup}, parametrized by the group description $(\mathbb{G}, p, \vec{G})$. It proves the relation

$$\mathsf{R}_{lup} \coloneqq \left\{ \left((\vec{f},\phi), F, \vec{t} \right) : \ F = \textstyle\sum_i f_i G_i + \phi H \ \wedge \ \vec{f} \subset \vec{t} \right\}$$

In Appendix A we provide a lookup argument which internally relies solely on Σ-protocols or compressed Σ-protocols, but any other lookup protocol may be used. Note that the commitment F to $\vec{f}$ is not sent throughout Π_{aes} of Figure 3. This is because it can be obtained via a linear transformations of the generators used. Consider the matrix S parametrized by the challenges $c_{\text{rj2}}, c_{\text{sbox}}, c_{\text{xor}}$

$$S \coloneqq \begin{bmatrix} S_{sbox,I} + c_{\text{sbox}}\, S_{sbox,O} \\ S_{rj2,I} + c_{\text{rj2}}\, S_{rj2,O} \\ S_{\text{xor},L} + c_{\text{xor}}\, S_{\text{xor},R} + c_{\text{xor}}^2\, S_{\text{xor},O} \end{bmatrix} \tag{11}$$

and note that $\vec{f} = S\vec{w}$, and thus $F = \langle \vec{w}, S \cdot \vec{G} \rangle + \omega H$.

Modularity. In the actual implementation we split the relation R_{aes} into two relations $\mathsf{R}_{aes.ksch}$ and $\mathsf{R}_{aes.cphr}$. This allows the prover to provide two separate zero-knowledge proofs for the key schedule and the cipher. In cases where the key schedule has already been expanded, or provided by a trusted third party, then one only needs to prove the cipher, reducing costs. Let

$$\mathsf{R}_{aes.ksch} \coloneqq \left\{ \left((\vec{k},\kappa), K \right) : K = \textstyle\sum_i rk_i G_i + \kappa H \ \wedge \ \vec{rk} = \mathsf{AES.Ksch}(\vec{k}) \right\}$$

be the relation identifying a correct commitment to the AES key schedule (using the same bit-string encoding above) and

$$\mathsf{R}_{aes.cphr} \coloneqq \left\{ \left((\vec{m},\mu,\vec{rk},\kappa), ctx, K, M \right) : K = \textstyle\sum_i rk_i G_i + \kappa H \ \wedge \ M = \sum_i m_i G_i + \mu H \ \wedge \ ctx = \mathsf{AES.Cphr}(\vec{rk},\vec{m}) \right\}$$

be the relation identifying a correct AES encryption. The core of the two proofs is essentially the same: the prover sends a commitment to the computation trace of a circuit whose gates are XOR, field multiplication by $\{2\}$ in Rijndael's field, and S-Box operations. Then, the prover engages in a (batch) lookup protocol with the verifier for each operation, across each round, to show that the computation trace is correct.

Extensions. Minor modifications of the relation $\mathsf{R}_{aes.cphr}$ (adding one XOR constraint), will allow to prove correct evaluation of AES in same-key Even-Mansour mode (AES-EM) as described by Dunkelman, Keller, and Shamir [DKS12, Definition 4.1]. Informally, the statement of $\mathsf{R}_{aes.cphr}$ would have to change so that the round keys $\vec{rk}$ are public and the prover shows $ctx = \vec{x} \oplus \mathsf{AES.Cphr}(\vec{rk}, \vec{x})$. AES-EM has been used in the context of signatures based on MPC-in-the-Head approaches [DKR+22, BBdSG+23], and we expect similar results to apply here. The proof system requires only minor modifications, so that the prover can provide a proof for the additional XOR constraint and the overall efficiency should remain roughly the same to Table 4 (protocol $\Pi_{aec.cphr}$).

Trivially repeating the proof will lead to a proof for the AES cipher in electronic-codebook (AES-ECB) mode. With the addition of one XOR constraint per block, also other modes such as AES-CBC, AES-CTR, AES-GCM are possible.

5.1 Analysis

In this section, we prove Theorem 3 showing that it satisfies completeness, special soundness, and honest-verifier zero-knowledge. The analysis is fairly simple as the core of the protocol boils down to one batch lookup invocation. We present a security analysis of our example lookup protocol in Appendix A.

Showing completeness is fairly straightforward: the verifier of Π_{aes} internally computes the vector $\vec{t} := (\vec{t}_{sbox} \| \vec{t}_{rj2} \| \vec{t}_{xor})$, sees F as a Pedersen commitment under generators $S \cdot \vec{G}$, and internally invokes the lookup protocol verifier. Completeness of the whole protocol immediately follows from completeness of the lookup protocol verifier.

Zero-knowledge is guaranteed by the (perfect) hiding property of Pedersen commitment, and hinges on the underlying lookup argument's security. Specifically, the zero-knowledge simulator samples a random element $W \leftarrow\!\!\$\ \mathbb{G}$ and internally runs the simulator for π_{lup}. The result follows by a simple union bound.

Theorem 12. *The protocol Π_{aes} for the relation R_{aes} is honest-verifier zero-knowledge.*

We are left with soundness, which we prove below.

Theorem 13. *The protocol Π_{aes} for the relation R_{aes} is 3-special sound with knowledge error $\epsilon = \epsilon_{dl} + 3\epsilon_{lup} + 10/p$, where ϵ_{lup} is the knowledge error of π_{lup} and ϵ_{dl} is the advantage in solving the discrete logarithm problem in $\mathbb{G}$.*

Proof Sketch. Consider an adversary that outputs valid transcripts $(W, c_0, \pi_{lup,0})$, $(W, c_1, \pi_{lup,1})$, and $(W, c_2, \pi_{lup,2})$, with $c_0 \neq c_1$, $c_1 \neq c_2$, and $c_0 \neq c_2$. For $i = 0, 1, 2$ recover the witness $\vec{f}_i, \phi_i$ from the knowledge-sound proof $\pi_{lup,i}$. Let J_1 and J_2 denote the indices of $\vec{f}$ encoding the XOR constraints of the initial state and the state at the end of the 0-th round. (Recall that the first round consists solely of $\mathsf{AddRoundKey}$ and the 0-th key is the AES key itself.) Consider the linear system in unknowns $x, y, z \in \mathbb{Z}_p$:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} 1 & c_0^5 & c_0^{10} \\ 1 & c_1^5 & c_1^{10} \\ 1 & c_2^5 & c_2^{10} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{12}$$

where $b_i \in \{f_{i,j}\}_{j \in J_1} \cup \{f_{i,j}\}_{j \in J_2} \cup \{\phi_j\}$ which admits one solution since $c_0 \neq c_1 \neq c_2$ and the thus the matrix is Vandermonde. (We assume p much larger than 5.) The extractor checks $x, y, z \in \{0, \ldots, 2^4 - 1\}$ and $z = x \oplus y$. If found, the extractor outputs the recovered values as the witness for the relation R_{aes}. The extractor outputs $\perp$ if no such values exist.

If the extractor outputs $\perp$, by soundness of $\pi_{lup,i}$ it follows that exists (different) $x_j, y_j, z_j \in \{0, \ldots, 2^4 - 1\}$ such that $x_j \oplus y_j = z_j$ and $x_j + y_j c_j^5 + z_j c_j^{10} = f_{j,0}$. (If that was not the case, then extraction failed for $\pi_{lup,i}$, as no valid witness was found.) However, this means that the adversary has found two different openings for the commitment $F = W + c_m M + c_k K$, which is a contradiction to the binding property of Pedersen commitment, which itself happens only with probability ϵ_{dl}. We are left with arguing that the values extracted are indeed from the commitments M and K (which follows from the Schwartz-Zippel lemma), and that the witness is indeed valid (which follows from the soundness of π_{lup}). Thus, the knowledge error of the extractor is at most $\epsilon_{dl} + 3\epsilon_{lup} + 10/p$. □

5.2 Efficiency

Asymptotic complexity. Π_{aes} is essentially a thin wrapper on top of the lookup protocol Π_{lup}. We denote the witness length (the number of XOR, S-Box, and {2}-mult. gates in the AES circuit) as n, and recall that the repeated structure of computation allows us to split the (large) XOR lookup table by a factor $c \in \{1, 2, 4\}$ with optimal value $c = 2$.

The overall prover's time cost consists of the cost of running the lookup protocol over $|\vec{f}|$ needles into a haystack vector $|\vec{t}|$ of $2^8 + 2^8 + 2^{16/c}$ elements (in our implementation, 768), plus a multi-scalar multiplication of size $c \cdot n$ for small elements (of size $2^{8/c}$) and one scalar multiplication (for zero-knowledge). When instantiated with Π_{lup} from Appendix A the prover

Table 4: Comparison between zero-knowledge AES-128 schemes and our protocol Π_{aes}. Proof size $|\pi|$ is in bytes. "Proof Type" indicates the techniques used among MPC-in-the-Head [IKOS07], FRI-based [BBHR18], or DL-based, and between parenthesis we indicate if they are plausibly post-quantum *for the zero-knowledge property*. Benchmarks on a laptop equipped with an Intel i7-1370P CPU and 32GB of RAM running Debian Linux.

	Proof type (PQ-ZK)	$\|\pi\|$	Prover time	Verifier time
PICNIC1-L3 [CDG+17]	MPCitH (✓)	74134	3.2ms	2.5ms
PICNIC2-L3 [CDG+17]	MPCitH (✓)	27173	123ms	41ms
FAEST [BBdSG+23]	MPCitH (✓)	6336	14ms	13ms
Preon128A [CCC+23]	FRI (✓)	139000	64s	414ms
Preon128B [CCC+23]	FRI (✓)	372000	65s	576ms
Lambdaclass [lam]	DL (✓)	855	34s	?
Ours (Σ-protocols)	DL (✓)	80864	37ms	13ms
Ours (compressed-Σ)	DL (✓)	2848	180ms	16ms

verifier time complexity are dominated by a linear number of group and field operations. Being based on Σ-protocols, the proof size is linear in the size of the witness. Additionally, we highlight that in the Σ-protocol Π_{lin} (Figure 7), the commitment operation (which is the most expensive of the whole protocol) is independent of the witness and can be precomputed in an offline phase.

Concrete efficiency. Concretely, the prover's computational cost is dominated by a multi-scalar multiplication (MSM) of size n over $\mathbb{Z}_p$ to commit to $\vec{q}$ (see Figure 5), and an MSM of size $\max(m, c \cdot n)$ for the commitment of $\vec{k}$ in the linear evaluation protocol in Figure 7 (providing zero-knowledge). The verifier's computational cost is dominated by an MSM of size $\max(m, c \cdot n)$ when verifying the linear evaluation proof. In practice, for the AES-128 cipher with $c = 2$, the prover has to do two MSMs, one of 1808 $\mathbb{Z}_p$ elements and one of 3616 elements (for zero-knowledge), and the verifier one MSM of 3616 elements; for the AES-256 cipher with $c = 2$, the prover has to do two MSMs, one of 2576 $\mathbb{Z}_p$ elements and one of 3488 elements. Enabling an offline phase, the prover's larger MSM can be precomputed and is not part of the final cost.

We implemented and made available Π_{aes} as an open-source library with tests and benchmarks, in Rust using the arkworks library,[12] and made it openly available under the BSD license.[13] As part of our implementation, we revisited arkworks' implementation of Pippenger's algorithm and optimized it for small scalars: in order to perform an MSM of the form $\sum_i x_i G_i$, we consider buckets $B_1, \ldots, B_8$ and add G_i to the bucket B_i if the i-th bit of x_i is set. Finally, we return $\sum_i 2^i B_i$. We also employ a batch version of the sumcheck protocol which is described in Appendix A. A summary of our results is shown in Table 4, where we benchmark Π_{aes} with `curve25519` [Ber06] along with alternative approaches. The table is indicative: the statements proven and the cryptographic assumptions used are different: FRI- and MPCitH-based proofs are for digital signatures and are thus proving equality of secret keys; Lambdaclass's implementation also considers the keyschedule (whereas we do consider only the cipher).Additional results are available along with the code source. All in all, we measure 32ms for proving relation $\mathsf{R}_{aes.cphr}$ on a Macboock M1 Pro when enabling multi-threading.

Other lookup protocols? Besides the protocol Π_{lup} of Appendix A, other lookup protocols may be used without harming soundness. The literature on the subject is quite large, and given the small table sizes of AES it is difficult to examine the most efficient protocol without looking at the fine-grained efficiency. At a high-level, if our protocol relates to the log-derivates approach of Habock [Hab22] in *logUp*, other approaches might be valid such as the memory-checking techniques [BEG+91, Set20, STW23], Merkle trees and accumulator-based techniques, or polynomial techniques [GW20]. The Merkle tree approach seems unpractical from a performance perspective since, in order to provide zero-knowledge, it requires an expensive sub-proof for knowledge of the pre-image of a hash function. The memory-checking techniques (used in e.g. [Set20, STW23]) and polynomial techniques (used in [GW20, BCHO22]) both internally rely on a sub-protocol called grand-product argument, respectively invoked 4 and 3 times.[14] Roughly speaking, a grand-product argument is a zero-knowledge proof that a vector $\vec{f} \in \mathbb{Z}_p^n$ satisfies $y = \prod_i f_i$. To do so, generally the prover commits to an auxiliary vector g embedding the partial products, and then proves that all partial products are computed correctly. We are not aware of any zero-knowledge grand-product argument for vectors of size n that does not rely on $O(n)$ multi-scalar multiplications of $\mathbb{Z}_p$ elements, and for this reason we think it's unlikely that other approaches will be more efficient than the one we

[12] https://arkworks.rs

[13] https://github.com/mmaker/tinybear

[14] The two techniques both present other important differences. For instance, the memory-checking techniques allow for a offline/preprocessing phase. Here, we focus on the prover's online time.

presented here with requires one single multi-scalar multiplication of $\mathbb{Z}_p$ elements of same size as the needles vector. We do believe however this to be an interesting avenue for future research.

6 Acknowledgements

The authors thank Trevor Perrin (Independent) and Melissa Chase (Microsoft Research), who took part to the initial writeup for Π_{dleq}. The authors would also like to express their gratitute towards Andrija Novakovic (Geometry), who was part of the initial writeup for Π_{dlhash}. Stephan Krenn (AIT) provided initial inputs and suggestions. arnaucube (0xPARC) authored the code used for benchmarking Π_{dlhash}.

References

[AC20] Thomas Attema and Ronald Cramer. Compressed Σ-protocol theory and practical application to plug & play secure algorithmics. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 513–543. Springer, Heidelberg, August 2020.

[AES01] Advanced Encryption Standard (AES). National Institute of Standards and Technology, NIST FIPS PUB 197, U.S. Department of Commerce, November 2001.

[AFLT12] Michel Abdalla, Pierre-Alain Fouque, Vadim Lyubashevsky, and Mehdi Tibouchi. Tightly-secure signatures from lossy identification schemes. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 572–590. Springer, Heidelberg, April 2012.

[AGM18] Shashank Agrawal, Chaya Ganesh, and Payman Mohassel. Non-interactive zero-knowledge proofs for composite statements. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 643–673. Springer, Heidelberg, August 2018.

[AGR+16] Martin R. Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. MiMC: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 191–219. Springer, Heidelberg, December 2016.

[ANT+20] Diego F. Aranha, Felipe Rodrigues Novaes, Akira Takahashi, Mehdi Tibouchi, and Yuval Yarom. LadderLeak: Breaking ECDSA with less than one bit of nonce leakage. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 225–242. ACM Press, November 2020.

[ASM06] Man Ho Au, Willy Susilo, and Yi Mu. Constant-size dynamic k-TAA. In Roberto De Prisco and Moti Yung, editors, *SCN 06*, volume 4116 of *LNCS*, pages 111–125. Springer, Heidelberg, September 2006.

[AST23] Arasu Arun, Srinath Setty, and Justin Thaler. Jolt: Snarks for virtual machines via lookups. Cryptology ePrint Archive, Paper 2023/1217, 2023. https://eprint.iacr.org/2023/1217.

[azt] Aztec emulated field and group operations. https://hackmd.io/@arielg/B13JoihA8.

[BBB+18] 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*, pages 315–334. IEEE Computer Society Press, May 2018.

[BBC+22] Clémence Bouvier, Pierre Briaud, Pyrros Chaidos, Léo Perrin, and Vesselin Velichkov. Anemoi: Exploiting the link between arithmetization-orientation and CCZ-equivalence. Cryptology ePrint Archive, Report 2022/840, 2022. https://eprint.iacr.org/2022/840.

[BBdSG+23] Carsten Baum, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß, Emmanuela Orsini, Lawrence Roy, and Peter Scholl. Publicly verifiable zero-knowledge and post-quantum signatures from vole-in-the-head. Cryptology ePrint Archive, Paper 2023/996, 2023. https://eprint.iacr.org/2023/996.

[BBDT16] Amira Barki, Solenn Brunet, Nicolas Desmoulins, and Jacques Traoré. Improved algebraic MACs and practical keyed-verification anonymous credentials. In Roberto Avanzi and Howard M. Heys, editors, *SAC 2016*, volume 10532 of *LNCS*, pages 360–380. Springer, Heidelberg, August 2016.

[BBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast reed-solomon interactive oracle proofs of proximity. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *ICALP 2018*, volume 107 of *LIPIcs*, pages 14:1–14:17. Schloss Dagstuhl, July 2018.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, Heidelberg, August 2004.

[BCC+16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 327–357. Springer, Heidelberg, May 2016.

[BCF+21a] Daniel Benarroch, Matteo Campanelli, Dario Fiore, Kobi Gurkan, and Dimitris Kolonelos. Zero-knowledge proofs for set membership: Efficient, succinct, modular. In Nikita Borisov and Claudia Diaz, editors, *Financial Cryptography and Data Security*, pages 393–414, Berlin, Heidelberg, 2021. Springer Berlin Heidelberg.

[BCF+21b] Daniel Benarroch, Matteo Campanelli, Dario Fiore, Jihye Kim, Jiwon Lee, Hyunok Oh, and Anaïs Querol. Proposal: commit-and-prove zero-knowledge proof systems and extensions. In *4th ZKProof Workshop*, 2021.

[BCG+13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: Verifying program executions succinctly and in zero knowledge. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 90–108. Springer, Heidelberg, August 2013.

[BCG+14] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE Computer Society Press, May 2014.

[BCG20a] Jonathan Bootle, Alessandro Chiesa, and Jens Groth. Linear-time arguments with sublinear verification from tensor codes. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 19–46. Springer, Heidelberg, November 2020.

[BCG+20b] Sean Bowe, Alessandro Chiesa, Matthew Green, Ian Miers, Pratyush Mishra, and Howard Wu. ZEXE: Enabling decentralized private computation. In *2020 IEEE Symposium on Security and Privacy*, pages 947–964. IEEE Computer Society Press, May 2020.

[BCG+22] Kenneth A Bamberger, Ran Canetti, Shafi Goldwasser, Rebecca Wexler, and Evan J Zimmerman. Verification dilemmas in law and the promise of zero-knowledge proofs. *Berkeley Tech. LJ*, 37:1, 2022.

[BCHO22] Jonathan Bootle, Alessandro Chiesa, Yuncong Hu, and Michele Orrù. Gemini: Elastic SNARKs for diverse environments. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 427–457. Springer, Heidelberg, May / June 2022.

[BCL+21] Benedikt Bünz, Alessandro Chiesa, William Lin, Pratyush Mishra, and Nicholas Spooner. Proof-carrying data without succinct arguments. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 681–710, Virtual Event, August 2021. Springer, Heidelberg.

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

[BCTV14] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Scalable zero knowledge via cycles of elliptic curves. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 276–294. Springer, Heidelberg, August 2014.

[BEG+91] Manuel Blum, William S. Evans, Peter Gemmell, Sampath Kannan, and Moni Naor. Checking the correctness of memories. In *32nd FOCS*, pages 90–99. IEEE Computer Society Press, October 1991.

[Ben14] Juan Benet. Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561*, 2014.

[Ber06] Daniel J. Bernstein. Curve25519: New Diffie-Hellman speed records. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *PKC 2006*, volume 3958 of *LNCS*, pages 207–228. Springer, Heidelberg, April 2006.

[BGH19] Sean Bowe, Jack Grigg, and Daira Hopwood. Halo: Recursive proof composition without a trusted setup. Cryptology ePrint Archive, Report 2019/1021, 2019. https://eprint.iacr.org/2019/1021.

[BLL+21] Fabrice Benhamouda, Tancrède Lepoint, Julian Loss, Michele Orrù, and Mariana Raykova. On the (in)security of ROS. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 33–53. Springer, Heidelberg, October 2021.

[BLS03] Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. Constructing elliptic curves with prescribed embedding degrees. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02*, volume 2576 of *LNCS*, pages 257–267. Springer, Heidelberg, September 2003.

[BLS04] Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. On the selection of pairing-friendly groups. In Mitsuru Matsui and Robert J. Zuccherato, editors, *SAC 2003*, volume 3006 of *LNCS*, pages 17–25. Springer, Heidelberg, August 2004.

[BMM+21] Benedikt Bünz, Mary Maller, Pratyush Mishra, Nirvan Tyagi, and Psi Vesely. Proofs for inner pairing products and applications. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part III*, volume 13092 of *LNCS*, pages 65–97. Springer, Heidelberg, December 2021.

[Bow17] Sean Bowe. BLS12-381: New zk-SNARK elliptic curve construction, 2017. https://electriccoin.co/blog/new-snark-curve/.

[Bra94] Stefan Brands. Untraceable off-line cash in wallets with observers (extended abstract). In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 302–318. Springer, Heidelberg, August 1994.

[BS20] Dan Boneh and Victor Shoup. A graduate course in applied cryptography, 2020. Available online https://toc.cryptobook.us/book.pdf.

[But14] Vitalik Buterin. Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform, 2014. Available at https://ethereum.org/en/whitepaper/.

[CCC+23] Ming-Shing Chen, Yu-Shian Chen, Chen-Mou Cheng, Shiuan Fu, Wei-Chih Hong, Jen-Hsuan Hsiang, Sheng-Te Hu, Po-Chun Kuo, Wei-Bin Lee, Feng-Hao Liu, and Justin Thaler. Preon: zk-snark based signature scheme, 2023. https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-1/spec-files/Preon-spec-web.pdf.

[CDDH19] Jan Camenisch, Manu Drijvers, Petr Dzurenda, and Jan Hajny. Fast keyed-verification anonymous credentials on standard smart cards. In *ICT Systems Security and Privacy Protection*, pages 286–298, 2019.

[CDG+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 Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1825–1842. ACM Press, October / November 2017.

[CFQ19] Matteo Campanelli, Dario Fiore, and Anaïs Querol. LegoSNARK: Modular design and composition of succinct zero-knowledge proofs. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2075–2092. ACM Press, November 2019.

[CGM16] Melissa Chase, Chaya Ganesh, and Payman Mohassel. Efficient zero-knowledge proof of algebraic and non-algebraic statements with applications to privacy preserving credentials. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 499–530. Springer, Heidelberg, August 2016.

[CHM+20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Psi Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768. Springer, Heidelberg, May 2020.

[CL02] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 61–76. Springer, Heidelberg, August 2002.

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72. Springer, Heidelberg, August 2004.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503. ACM Press, May 2002.

[CMZ14] Melissa Chase, Sarah Meiklejohn, and Greg Zaverucha. Algebraic MACs and keyed-verification anonymous credentials. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014*, pages 1205–1216. ACM Press, November 2014.

[CP93] David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 89–105. Springer, Heidelberg, August 1993.

[CPZ20] Melissa Chase, Trevor Perrin, and Greg Zaverucha. The signal private group system and anonymous credentials supporting efficient verifiable encryption. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1445–1459. ACM Press, November 2020.

[Cra97] Ronald Cramer. *Modular Design of Secure yet Practical Cryptographic Protocols*. PhD thesis, CWI Amsterdam, The Netherlands, 1997.

[CT10] Alessandro Chiesa and Eran Tromer. Proof-carrying data and hearsay arguments from signature cards. In *ICS*, volume 10, pages 310–331, 2010.

[DBB+15] Gaby G. Dagher, Benedikt Bünz, Joseph Bonneau, Jeremy Clark, and Dan Boneh. Provisions: Privacy-preserving proofs of solvency for bitcoin exchanges. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015*, pages 720–731. ACM Press, October 2015.

[DKR+22] Christoph Dobraunig, Daniel Kales, Christian Rechberger, Markus Schofnegger, and Greg Zaverucha. Shorter signatures based on tailor-made minimalist symmetric-key crypto. pages 843–857. ACM Press, 2022.

[DKS12] Orr Dunkelman, Nathan Keller, and Adi Shamir. Minimalism in cryptography: The Even-Mansour scheme revisited. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 336–354. Springer, Heidelberg, April 2012.

[DLO+18] Ivan Damgård, Ji Luo, Sabine Oechsner, Peter Scholl, and Mark Simkin. Compact zero-knowledge proofs of small Hamming weight. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 530–560. Springer, Heidelberg, March 2018.

[DR91] Joan Daemen and Vincent Rijmen. The design of {Rijndael}:{AES}. *Journal of Cryptology*, 4(1):3–72, 1991.

[DRZ20] Vanesa Daza, Carla Ràfols, and Alexandros Zacharakis. Updateable inner product argument with logarithmic verifier and applications. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 527–557. Springer, Heidelberg, May 2020.

[dVYA] Henry de Valence, Cathie Yun, , and Oleg Andreev. Rust bulletproofs crate. Accessed October 2022 (HEAD at 6fb4135).

[Eag22] Liam Eagen. Bulletproofs++. Cryptology ePrint Archive, Report 2022/510, 2022. https://eprint.iacr.org/2022/510.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987.

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *22nd ACM STOC*, pages 416–426. ACM Press, May 1990.

[FW22] Georg Fuchsbauer and Mathias Wolf. Concurrently secure blind schnorr signatures. Cryptology ePrint Archive, Paper 2022/1676, 2022. https://eprint.iacr.org/2022/1676.

[GHL22] Craig Gentry, Shai Halevi, and Vadim Lyubashevsky. Practical non-interactive publicly verifiable secret sharing with thousands of parties. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part I*, volume 13275 of *LNCS*, pages 458–487. Springer, Heidelberg, May / June 2022.

[GKL+22] Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, Markus Schofnegger, and Roman Walch. Reinforced concrete: A fast hash function for verifiable computation. pages 1323–1335. ACM Press, 2022.

[GKR+21] Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. Poseidon: A new hash function for zero-knowledge proof systems. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021*, pages 519–535. USENIX Association, August 2021.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 339–358. Springer, Heidelberg, May / June 2006.

[GP20] Ariel Gabizon and Zachary J. Williamson (Aztec Protocol). Proposal: The turbo-plonk program syntax for specifying snark programs. ZKProof3, 2020.

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

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Heidelberg, April 2008.

[GW20] Ariel Gabizon and Zachary J. Williamson. plookup: A simplified polynomial protocol for lookup tables. Cryptology ePrint Archive, Paper 2020/315, 2020. https://eprint.iacr.org/2020/315.

[GWC19] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019. https://eprint.iacr.org/2019/953.

[Hab22] Ulrich Haböck. Multivariate lookups based on logarithmic derivatives. Cryptology ePrint Archive, Paper 2022/1530, 2022. https://eprint.iacr.org/2022/1530.

[HdVLA22] Mike Hamburg, Henry de Valence, Isis Lovecruft, and Tony Arcieri. The Ristretto group, 2022. https://ristretto.group/.

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

[KHSS22] Daniel Kang, Tatsunori Hashimoto, Ion Stoica, and Yi Sun. Zk-img: Attested images via zero-knowledge proofs to fight disinformation, 2022.

[Kil90] Joe Kilian. *Uses of Randomness in Algorithms and Protocols*. MIT Press, Cambridge, MA, USA, 1990.

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography: principles and protocols*. Chapman and hall/CRC, 2007.

[KLS18] Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 552–586. Springer, Heidelberg, April / May 2018.

[KS23] Abhiram Kothapalli and Srinath Setty. Cyclefold: Folding-scheme-based recursive arguments over a cycle of elliptic curves. Cryptology ePrint Archive, Paper 2023/1192, 2023. https://eprint.iacr.org/2023/1192.

[lam] Lambdaclass aes encryption circuit. https://github.com/lambdaclass/AES_zero_knowledge_proof_circuit.

[Lee21] Jonathan Lee. Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part II*, volume 13043 of *LNCS*, pages 1–34. Springer, Heidelberg, November 2021.

[LFKN90] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. In *31st FOCS*, pages 2–10. IEEE Computer Society Press, October 1990.

[Lin03] Yehuda Lindell. Parallel coin-tossing and constant-round secure two-party computation. *Journal of Cryptology*, 16(3):143–184, June 2003.

[LKWL22] T. Looker, V. Kalos, A. Whitehead, and M. Lodder. The BBS signature scheme, October 2022. IRTF CFRG working group draft.

[Lyu08] Vadim Lyubashevsky. Lattice-based identification schemes secure under active attacks. In Ronald Cramer, editor, *PKC 2008*, volume 4939 of *LNCS*, pages 162–179. Springer, Heidelberg, March 2008.

[Lyu09] Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 598–616. Springer, Heidelberg, December 2009.

[Mau09] Ueli Maurer. Unifying zero-knowledge proofs of knowledge. In B. Preneel, editor, *Advances in Cryptology - AfricaCrypt 2009*, Lecture Notes in Computer Science. Springer-Verlag, 6 2009.

[MGGR13] Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. Zerocoin: Anonymous distributed E-cash from Bitcoin. In *2013 IEEE Symposium on Security and Privacy*, pages 397–411. IEEE Computer Society Press, May 2013.

[MRV16] Paz Morillo, Carla Ràfols, and Jorge Luis Villar. The kernel matrix Diffie-Hellman assumption. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 729–758. Springer, Heidelberg, December 2016.

[Nak08] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System, 2008. Available at http://bitcoin.org/bitcoin.pdf.

[NBS23] Wilson Nguyen, Dan Boneh, and Srinath Setty. Revisiting the nova proof system on a cycle of curves. Cryptology ePrint Archive, Paper 2023/969, 2023. https://eprint.iacr.org/2023/969.

[Oka93] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 31–53. Springer, Heidelberg, August 1993.

[Ped92] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Heidelberg, August 1992.

[Pol78] John M. Pollard. Monte carlo methods for index computation (mod p). *Mathematics of computation*, 32(143):918–924, 1978.

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000.

[PS16] David Pointcheval and Olivier Sanders. Short randomizable signatures. In *CT-RSA 2016*, pages 111–126, 2016.

[PZ13] C. Paquin and G. Zaverucha. U-prove cryptographic specification v1.1 (revision 2), 2013. Available online: www.microsoft.com/uprove.

[SAB+19] Alberto Sonnino, Mustafa Al-Bassam, Shehar Bano, Sarah Meiklejohn, and George Danezis. Coconut: Threshold issuance selective disclosure credentials with applications to distributed ledgers. In *NDSS 2019*. The Internet Society, February 2019.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, January 1991.

[Set20] Srinath Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 704–737. Springer, Heidelberg, August 2020.

[Sma99] Nigel P. Smart. The discrete logarithm problem on elliptic curves of trace one. *Journal of Cryptology*, 12(3):193–196, June 1999.

[SSS+22] Huachuang Sun, Haifeng Sun, Kevin Singh, Akhil Sai Peddireddy, Harshad Patil, Jianwei Liu, and Weikeng Chen. The inspection model for zero-knowledge proofs and efficient zerocash with secp256k1 keys. Cryptology ePrint Archive, Report 2022/1079, 2022. https://eprint.iacr.org/2022/1079.

[STW23] Srinath Setty, Justin Thaler, and Riad Wahby. Unlocking the lookup singularity with lasso. *Cryptology ePrint Archive*, 2023.

[Tha13] Justin Thaler. Time-optimal interactive proofs for circuit evaluation. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 71–89. Springer, Heidelberg, August 2013.

[Val08] Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 1–18. Springer, Heidelberg, March 2008.

[XZZ+19] Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. Libra: Succinct zero-knowledge proofs with optimal prover computation. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 733–764. Springer, Heidelberg, August 2019.

[Yas12] Masaya Yasuda. A generalization of the anomalous attack for the ecdlp over qp. *International Journal of Pure and Applied Mathematics*, 77(1):1–9, 2012.

[zca] Halo2 sha256. https://zcash.github.io/halo2/design/gadgets/sha256.html.

[zkb] ZK Bug Tracker. https://github.com/0xPARC/zk-bug-tracker/.

A A simple lookup for small tables

Our lookup argument, illustrated in in Figure 4, proves in zero-knowledge statements for the relation

$$\mathsf{R}_{lup} \coloneqq \left\{ ((\vec{f}, \phi), F, \vec{t}) : \ F = \sum_{i=1}^{n} f_i G_i + \phi H \ \wedge \ \forall i \in [1, n], f_i \in \{t_j\}_{j=1}^{|\vec{t}|} \right\}$$

Note that the table to be queried, $\vec{t}$, is part of the instance and known to the verifier.

Inner-product proofs. We make use of two sub-protocols that are fairly standard in the literature:

- Π_{tsp} for proving in zero-knowledge statements for the inner product relation

$$\mathsf{R}_{tsp} \coloneqq \left\{ ((\vec{f}, \phi, \vec{e}, \varepsilon, y, \psi), \vec{v}, F, E, Y) : \begin{array}{c} F = \sum_{i=1}^{n} f_i G_i + \phi H \ \wedge \ E = \sum_{i=1}^{n} e_i G_i + \varepsilon H \ \wedge \ Y = yG + \psi H \\ \langle \vec{f}, \vec{v} \circ \vec{e} \rangle = y \end{array} \right\} ;$$

- Π_{lin} for proving in zero-knowledge statements for the linear evaluation relation

$$\mathsf{R}_{lin} \coloneqq \left\{ ((\vec{f}, \phi, y, \psi), (F, \vec{e}, Y)) : \ F = \textstyle\sum_i f_i G_i + \phi H \ \wedge \ Y = yG + \psi H \ \wedge \ \langle \vec{f}, \vec{e} \rangle = y \right\} .$$

It is possible to instantiate Π_{lin} using Π_{tsp} setting $\vec{v} = \vec{1}$, $\varepsilon = 0$, and $E = \sum_i e_i G_i$. For inner-products of size $n = |\vec{v}|$, the best asymptotics for Π_{tsp} to this day are dominated by $O(n)$ group operations, with $O(n)$ group operations for the verifier as well (in the pairing-free setting).[15] We describe an inner-product argument based on Σ-protocols in Appendix B for completeness.

[15] In the pairing setting, Dory [Lee21] achieves sublinear time, but requires $O(\log n)$ pairings; in the trusted-setup setting, it is possible to achieve constant-time verification using Lagrange interpolation and univariate sumcheck [CHM+20], at the cost of a quasilinear prover.

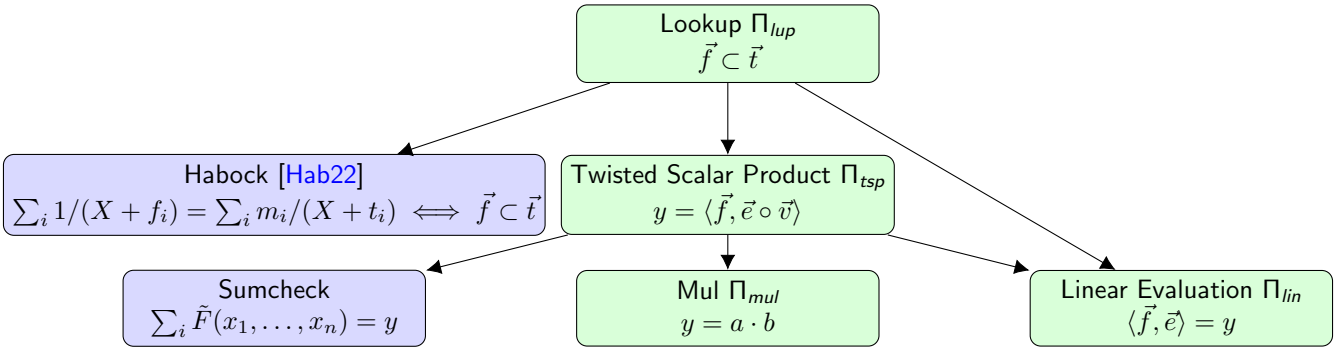

Figure 4: Structure of our zero-knowledge lookup argument using a result by [Hab22] and a linear evaluation argument and an inner product argument as sub-protocols. The inner product argument internally uses a sumcheck result by [BCHO22] (not zero-knowledge), combined with Π_{lin} and Π_{mul}.

Protocol. A formal description of the lookup argument is given in Figure 5. Given input $((\vec{f}, \phi), F, \vec{t})$ the prover computes and sends a commitment to the frequency vector

$$\vec{m} \coloneqq \left[\sum_{j=1}^{n} (t_i = f_j)\right]_i^{|t|} \quad ,$$

that is, $\vec{m}$ the vector of Lemma 14 that at the i-th position holds the number of times t_i appears in $\vec{f}$. The verifier samples and sends $c \leftarrow\!\!\$\ \mathbb{Z}_p$. The value c will be a random evaluation point for the rational polynomial given in Lemma 14, used to check that the equation holds for the claimed f_i with high probability. Note that once $X = c$, Equation (16) simplifies to $q = \sum m_i h_i$ where q and m_i are values known to the prover, and h_i is public. The prover and verifier both compute

$$\vec{h} \coloneqq \left[(t_i + c)^{-1}\right]_i^{|t|} \quad .$$

The prover also computes and sends a commitment to

$$\vec{q} \coloneqq \left[(f_i + c)^{-1}\right]_{i=1}^{|f|} \quad .$$

We are left with showing that

$$\langle \vec{m}, \vec{h} \rangle = \langle \vec{q}, \vec{1} \rangle \quad , \text{ and} \tag{13}$$

$$\vec{q} \circ (\vec{f} + c \cdot \vec{1}) = \vec{1} \quad , \tag{14}$$

to check that Lemma 14 holds and, respectively, that the prover computed $\vec{q}$ correctly ("$\circ$" denotes component-wise – Hadamard – product). Left- and right-hand side of Equation (13) are linear evaluation claims that can be shown with Π_{lin}. To show Equation (14), we use a common approach for reducing Hadamard products to inner-products [BCHO22]: the verifier samples a random vector $\vec{v}$ and prover and verifier check that

$$\langle \vec{q}, \vec{v} \circ (\vec{f} + c \cdot \vec{1}) \rangle = \sum_{i=1}^{n} v_i \quad . \tag{15}$$

If above relation, called *twisted inner-product*, holds, then Equation (14) holds except with negligible probability $|\vec{f}|/|\mathbb{Z}_p|$. The latter is easier than the former as the component-wise product is with a public vector. It is possible to instantiate twisted scalar product from inner-product arguments.[16] When $\vec{v}$ is set to be consecutive powers $(1, v, v^2, \ldots)$, the right-hand side can be computed in $O(\log |\vec{v}|)$ field operations.

[16] Split-and-fold arguments with independent generators can generally use the "generator-swapping" trick [BBB+18]: the commiter key of the right-hand side is considered to be $\vec{v} \circ \vec{G}$ instead of $\vec{G}$. For sumcheck-based techniques, it is possible to embed the "twist" within the sumcheck messages.

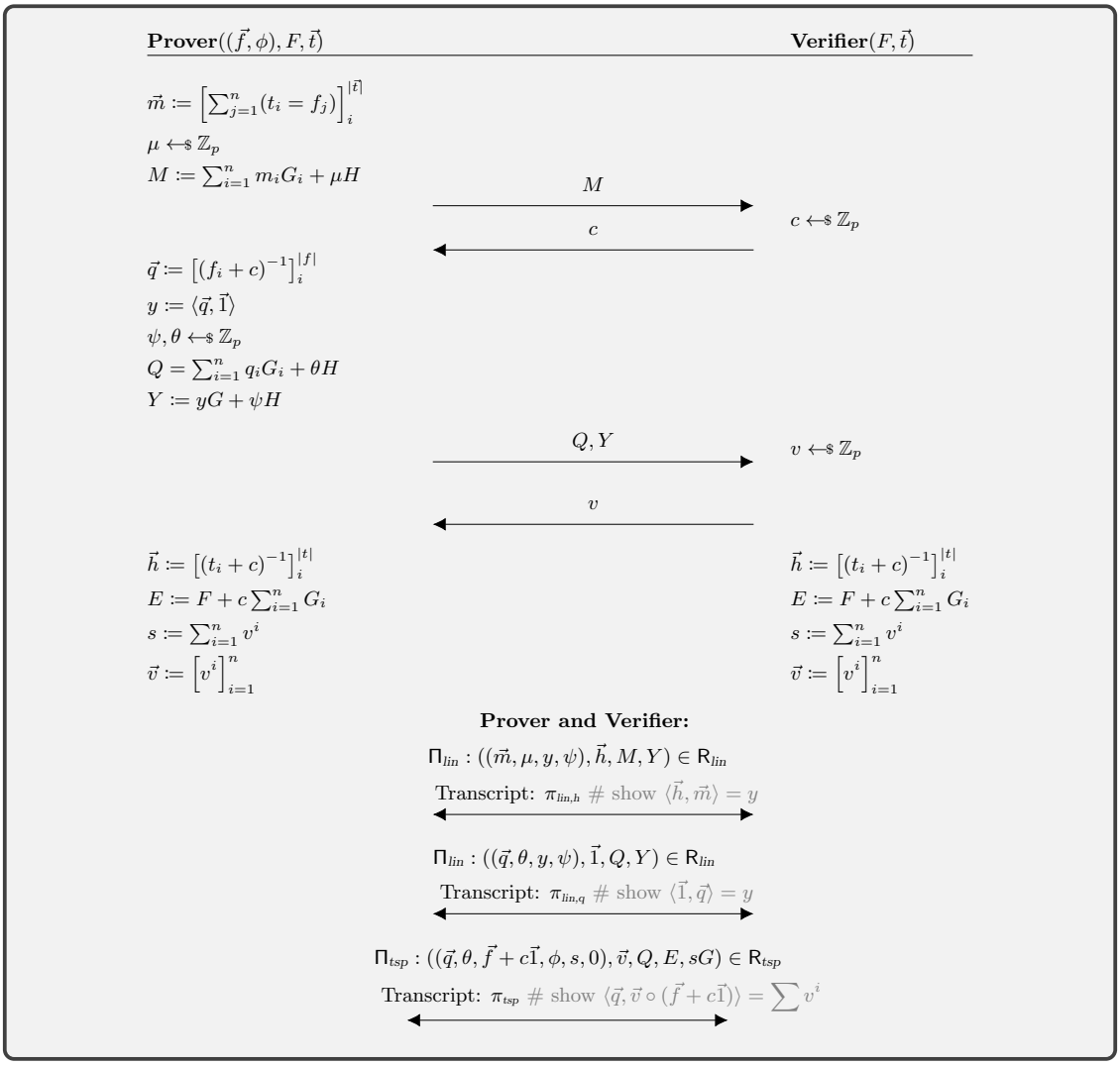

Figure 5: The lookup protocol Π_{lup} for checking that $\vec{f} \subset \vec{t}$ and $F = \sum_i f_i G_i + \phi H$.

A.1 Analysis

In this section, we prove the security of our lookup argument by showing that Π_{lup} satisfies completeness, knowledge soundness, and honest-verifier zero-knowledge. At the core of our protocol is the following observation from Haböck [Hab22].

Lemma 14 ([Hab22, Lemma 5])**.** *Let $\vec{f}, \vec{t}$ be vectors over $\mathbb{Z}_p$ of dimension less than p. Then, $\{f_i\}_i \subset \{t_i\}_i$ if and only if there exists a sequence of integers $\vec{m}$ (all less than p) such that*

$$\sum_{i=1}^{|\vec{f}|} \frac{1}{X + f_i} = \sum_{i=1}^{|\vec{t}|} \frac{m_i}{X + t_i} \tag{16}$$

holds in the function field $\mathbb{Z}_p[X]$.

Informally, the vector $\vec{m}$ accounts for the multiplicity of each element f_i in $\vec{t}$. For example, if $\vec{f} \coloneqq (1, 1, 1, 2, 3)$ and $\vec{t} \coloneqq (1, 2, 3, 4)$, then $\vec{m} = (3, 1, 1, 0)$. Looking ahead, the equality will be checked on a random challenge point sent by the verifier, and soundness will be guaranteed by Schwartz-Zippel Lemma over the above identity. First, assuming that the underlying Π_{tsp} and Π_{lin} are complete, we note that the protocol Π_{lup} is complete.

Theorem 15. *The protocol Π_{lup} for the relation R_{lup} is complete with error $|\vec{t}|/|\mathbb{Z}_p| + 2\delta_{lin} + \delta_{tsp}$.*

Proof. The protocol aborts if the sub-protocols Π_{tsp} and Π_{lin} do not complete successfully, or if the field inversion fails for generating the elements $\vec{q}$ and $\vec{h}$. As $\vec{f} \subset \vec{t}$ and $c \in \mathbb{Z}_p$ is sampled uniformly at random, it follows that the protocol aborts with probability at most $|\vec{t}|/|\mathbb{Z}_p| + 2\delta_{lin} + \delta_{tsp}$. □

Theorem 16. *Suppose Π_{lup} from Figure 5 is instantiated with a knowledge-sound Π_{lin} protocol and a knowledge-sound Π_{tsp} for the relations R_{lin} and R_{tsp} respectively. Then Π_{lup} is knowledge-sound if the discrete logarithm problem is hard.*

Proof. We consider the extractor that internally runs the extractor of Π_{tsp} on the proof π_{tsp} for $(\vec{v}, Q, E, sG)$ to extract some $(\vec{q}, \theta, \vec{e}, \epsilon, y, \psi)$ such that $Q = \sum q_i G_i + \theta H$, $E = \sum e_i G_i + \epsilon H$ and $sG = yG + \psi H$ with $y = \langle \vec{q}, \vec{v} \circ \vec{e} \rangle$. The extractor returns $(\vec{f} := \vec{e} - c\vec{1},\ \phi := \epsilon)$. We must show that it is a valid witness for the relation R_{tsp}, and we do sia via a hybrid argument.

- $\mathbf{H}_1$ This game behaves identically to the original knowledge soundness game for Π_{lup} except that if $\Pi_{tsp}.\mathsf{Ext}$ on the proof π_{tsp} for $(\vec{v}, Q, E, sG)$ fails extract some $(\vec{q}, \theta, \vec{e}, \varepsilon, y, \psi)$ such that $((\vec{q}, \theta, \vec{e}, \varepsilon, y, \psi), \vec{v}, Q, E, sG) \in \mathsf{R}_{tsp}$ then the game aborts. This game is indistinguishable by the knowledge soundness of Π_{tsp}.

- $\mathbf{H}_2$ Run the extractor of Π_{lin} on the proof $\pi_{lin,\ h}$ for $(\vec{h}, M, Y)$ fails extract some $(\vec{m}, \mu, y', \psi')$ such that $((\vec{m}, \mu, y', \psi'), \vec{h}, M, Y) \in \mathsf{R}_{lin}$ then the game aborts. This game is indistinguishable by the knowledge soundness of Π_{lin}.

- $\mathbf{H}_3$ Run the extractor of Π_{lin} on the proof $\pi_{lin,\ q}$ for $(\vec{1}, Q, Y)$ fails extract some $(\vec{q}'', \theta'', y'', \psi'')$ such that $((\vec{q}'', \theta'', y'', \psi''), \vec{1}, Q, Y) \in \mathsf{R}_{lin}$ then the game aborts. This game is indistinguishable by the knowledge soundness of Π_{lin}.

- $\mathbf{H}_4$ If $\vec{q} \neq \vec{q}''$ and $\theta \neq \theta''$ then the game aborts. Indeed if this game aborts when the previous one doesn't then

$$Q = \sum_{i=1}^{n} q_i G_i + \theta H = \sum_{i=1}^{n} q_i'' G_i + \theta'' H$$

  and we can build a reduction against the discrete logarithm problem.

- $\mathbf{H}_5$ If $s \neq y$ or $s \neq y'$ or $s \neq y''$ then the game aborts. Indeed if this game aborts when the previous one doesn't then

$$sG = yG + \psi H = y'G + \psi' H = y''G + \psi'' H$$

  and we can build a reduction against the discrete logarithm problem.

- $\mathbf{H}_6$ If there exists $i \in [1, n]$ such that $q_i \neq f_i + c$ then the game aborts. If this game aborts when the previous one doesn't then

$$\vec{q} \circ (\vec{f} + c\vec{1}) \neq \vec{1}$$

  We design a reduction against the discrete logarithm problem. The reduction runs the adversary on the same random coins initially, but in the third move sends a different random $v^* \neq v$. The reduction continues trying v^* values until the adversary terminates with a valid proof.

  The reduction runs the $\Pi_{tsp}.\mathsf{Ext}$ with respect to the proof π_{tsp} for $(\vec{v}^*, Q, E, s^*G)$ to extract some $(\vec{q}^*, \theta^*, \vec{e}^*, \varepsilon^*, y^*, \psi^*)$ such that $((\vec{q}^*, \theta^*, \vec{e}^*, \varepsilon^*, y^*, \psi^*), \vec{v}^*, Q, E, s^*G) \in \mathsf{R}_{tsp}$. By the knowledge soundness of Π_{tsp} this succeeds with overwhelming probability.

  If $\vec{q} \neq \vec{q}^*$ and $\theta \neq \theta^*$ then

$$Q = \sum_{i=1}^{n} q_i G_i + \theta H = \sum_{i=1}^{n} q_i^* G_i + \theta^* H$$

  and the reduction can solve the discrete logarithm problem. Similarly if $\vec{e} \neq \vec{e}^*$ and $\varepsilon \neq \varepsilon^*$ then the reduction can solve the discrete logarithm problem. If $y^* \neq s^*$ then the reduction can solve the discrete logarithm problem. Else

$$\begin{aligned} \langle \vec{q}^*, \vec{v}^* \circ \vec{e}^* \rangle = \sum v_i^* &\Leftarrow \langle \vec{q}, \vec{v}^* \circ \vec{e} \rangle = \sum v_i^* \\ &\Leftarrow \langle \vec{q}, \vec{v}^* \circ (\vec{f} + c\vec{1}) \rangle = \sum v_i^* \\ \sum (v^*)^i q_i (f_i + c) = \sum (v^*)^i & \end{aligned}$$

  The probability of the latter occurring is $\frac{1}{p}$ because $\vec{q}$ and $\vec{f}$ are independent from v^* and $\vec{q} \circ (\vec{f} + c\vec{1}) \neq \vec{1}$. □

Theorem 17. *Suppose Π_{lup} from Figure 5 is instantiated with honest-verifier zero-knowledge Π_{lin} and Π_{tsp} protocols for the relations R_{lin} and R_{tsp} respectively. Then Π_{lup} is honest-verifier zero-knowledge.*

Proof. Consider the simulator that, given as input $F, \vec{t}$, chooses $c, v \leftarrow\$ \mathbb{Z}_p$, then $M, Q, Y \leftarrow\$ \mathbb{G}$, and then, for the final round messages:

- compute $\vec{h} = [(t_i + c)^{-1}]_i^{|t|}$, then $E = F + c\sum_{i=1}^{n} G_i$;
- compute $\pi_{lin,h} \leftarrow \Pi_{lin}.\mathsf{Sim}(\vec{h}, M, Y)$, $\pi_{lin,q} \leftarrow \Pi_{lin}.\mathsf{Sim}(\vec{1}, Q, Y)$, $\pi_{tsp} \leftarrow \Pi_{tsp}.\mathsf{Sim}(\vec{v}, Q, E, sG)$.

Finally, return the transcript

$$(F, \vec{t}), M, c, (Q, Y), v, (\pi_{lin,h}, \pi_{lin,q}, \pi_{tsp}) \ .$$

$\mathbf{H}_1$ This game behaves identically to the original zero-knowledge game for Π_{lup} except that when the honest prover is run, the proof $\pi_{lin,h}$ is computed as $\pi_{lin,h} \leftarrow \Pi_{lin}.\mathsf{Sim}(\vec{h}, M, Y)$. Since the honest prover knows a witness for $\vec{h}, M, Y$, we can build a reduction against the zero-knowledge of Π_{lin} should this change be distinguishable.

$\mathbf{H}_2$ The proof $\pi_{lin,q}$ is computed as $\pi_{lin,q} \leftarrow \Pi_{lin}.\mathsf{Sim}(\vec{1}, Q, Y)$. Since the honest prover knows a witness for $(\vec{1}, Q, Y)$, we can build a reduction against the zero-knowledge of Π_{lin} should this change be distinguishable.

$\mathbf{H}_3$ The proof π_{tsp} is computed as $\pi_{tsp} \leftarrow \Pi_{tsp}.\mathsf{Sim}(\vec{v}, Q, E, sG)$. Since the honest prover knows a witness for $(\vec{v}, Q, E, sG)$, we can build a reduction against the zero-knowledge of Π_{tsp} should this change be distinguishable.

We now see that the final hybrid is statistically impossible. Indeed, the only proof elements that are non-identical are M, Q and Y. However these elements are computed uniformly at random both by the honest prover (due to the blinders μ, θ, ψ) and the simulator. □

A.2 Efficiency

A discussion about asymptotic and concrete efficiency is already present in Section 5.2. Informally, the prover and verifier complexity is linear in the size of $|\vec{t}|$, and requires performing two linear-size group operation: one multi-scalar multiplication of $\mathbb{Z}_p$-elements of size $|\vec{f}|$ (to commit to $\vec{q}$) and one of small integers of size $|\vec{t}|$ (recall that $\sum_i m_i = |\vec{f}|$). Finally, prover and verifier must account for the inner-product and linear evaluation protocols.

Concrete efficiency. When instantiated with Π_{tsp}, some optimizations are possible to lower the concrete running-time of the prover's algorithm. instead of running all linear evaluation claims (concerning: $\langle \vec{q}, \vec{1} \rangle$, $\langle \vec{m}, \vec{h} \rangle$, $\langle \vec{q}, \otimes_j (1, c_j) \rangle$, $\langle \vec{f} + c\vec{1} \rangle$) separately, the verifier sends a single challenge c_{batch} and the prover and verifier engage into a sumcheck protocol reducing the above into a single claim about a linear evaluation of $\vec{q} + c_{batch}\vec{m} + c_{batch}^2(\vec{f} + c\vec{1})$, which is proven with a single Π_{lin} invocation.[17] This is a standard technique in the literature for batching Schnorr proofs and polynomial commitment evaluations, and in this case it concretely it allows to trade off the prover's time from 4 Π_{lin}'invocations to 1. In practice, also the proof size of Π_{aes} decreases from roughly 122K to 80K when using Π_{lin} of Figure 7.

If Π_{lin} is instantiated with standard Σ-protocols, the overall lookup protocol's complexity is dominated by two MSM of $\mathbb{Z}_p$-elements of size $|\vec{f}|$ and $\max(|\vec{f}|, |\vec{t}|)$. If the protocol Π_{lin} is instantiated with compressed Σ-protocols, the overall lookup protocol's complexity is dominated by $\max(|\vec{f}|, |\vec{t}|)$ scalar multiplication and additions in $\mathbb{G}$ (the generator's split and fold sub-procedure). If the protocol Π_{lin} is instantiated with log-size arguments, the overall lookup protocol's proof transcipt becomes logarithmic while slightly increasing the overall (concrete) cost (maintaining the same asymptotic): the proving time is dominated by the split-and-fold procedure of the basis, which requires $\max(|\vec{f}|, |\vec{t}|)$ scalar multiplication and additions in $\mathbb{G}$. The overall proof size for Π_{aes} goes down to less than 3K as seen in Table 4.

Function evaluation via lookup. Let ϕ be a function with a short (polynomial in λ) table representation. Let $\vec{f}_{\text{in}}, \vec{f}_{\text{out}} \in \mathbb{Z}_p^n$ be vectors. A standard technique for proving correct function evaluation $\phi(\vec{f}_{\text{in}}) = \vec{f}_{\text{out}}$ is to use a lookup protocol.[18] To do so, the verifier sends a random challenge c and the prover, verifier engage in a lookup protocol for $\vec{f} = \vec{f}_{\text{in}} + c\vec{f}_{\text{out}}$ inside $\vec{t} = [\vec{a} + c \cdot \phi(\vec{a})]_{a \in \text{Dom}(\varphi)}$. This technique, which (to our knowledge) first appears in plookup [GW20], is used in Section 5 to check the correct evaluation of the boolean functions composing AES.

[17] By "batch sumcheck" here we mean running the sumcheck protocol over the random linear combination of the sumcheck polynomials associated to the inner-product claims as described in Appendix B.

[18] The notation above is imprecise. By $\phi(\vec{f})$ we actually mean the element-wise application of the map ϕ.

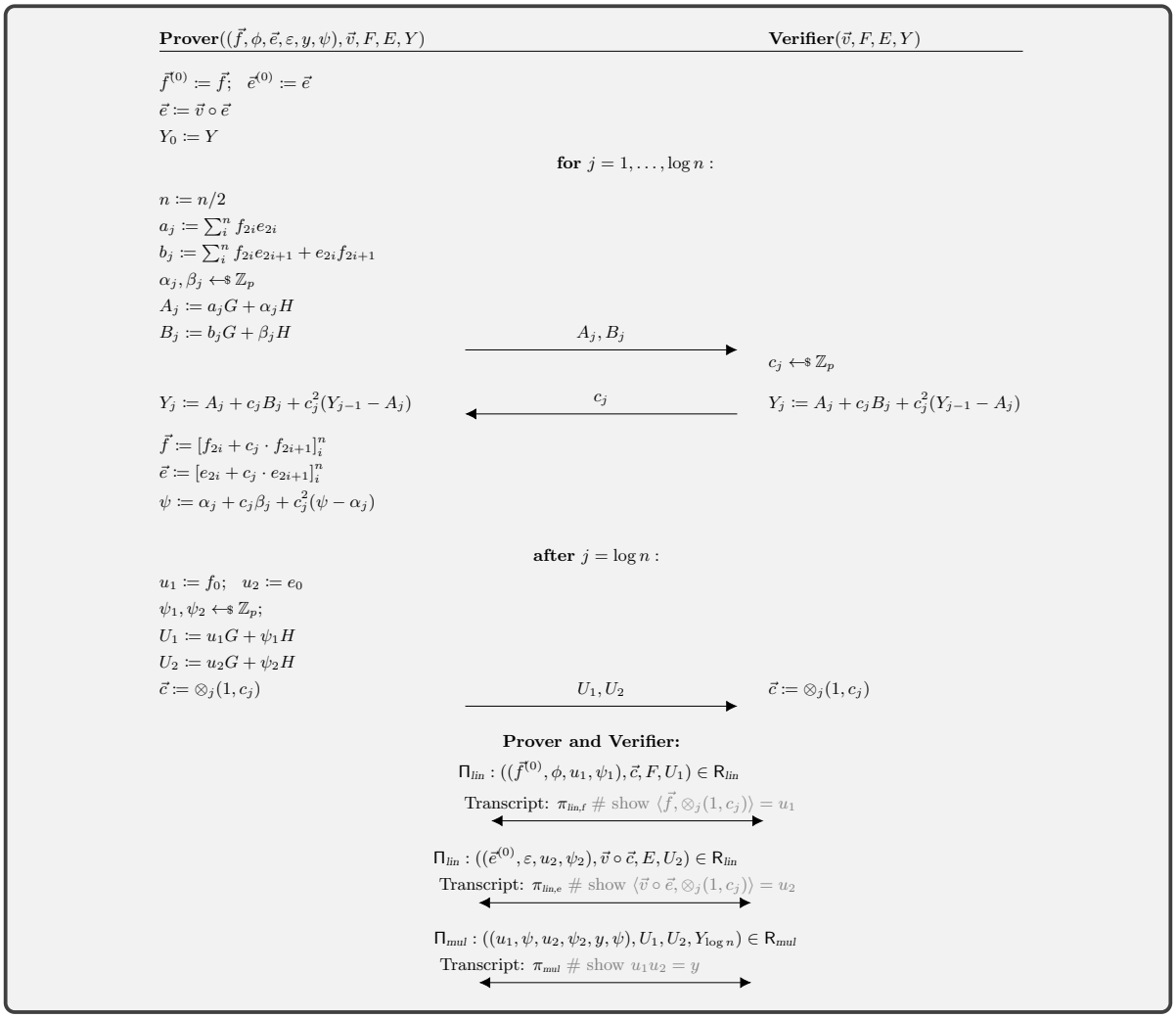

Figure 6: The inner-product protocol Π_{tsp} for showing that F, E, Y contain $\vec{f}, \vec{e}, y$ such that $\langle \vec{f}, \vec{v} \circ \vec{e}\rangle = y$. The protocol uses a sumcheck argument with zero-knowledge obtained by committing to each round message. The protocol is used during Π_{lup}

Batching function evaluation. When proving correct function evaluation of $\phi_1, \phi_2, \ldots, \phi_2$, the verifier can send challenges c_i for $i = 1,\ldots,m$, so that the prover can construct $\vec{f_i}$ and $\vec{t_i}$ as above using c_i. Then, prover and verifier engage in a single lookup for $(\vec{f_1}\|\vec{f_2}\|\cdots\|\vec{f_m}) \subset (\vec{t_1},\|\vec{t_2}\|\cdots\|\vec{t_m})$.

B An inner-product argument from Σ-protocols

We describe an inner-product proof Π_{tsp} in Figure 6. The relation proven is actually a slight generalization of it, that allows for a public "twist"in the inner-product relation. Informally, the protocol performs a multivariate sumcheck protocol (in zero-knowledge) that reduces the inner-product claim to two tensorcheck claims [BCHO22], which are then proven using the linear evaluation protocol Π_{lin}. The literature presents a vast amount of inner-product protocols [BCC+16, BMM+21, BBB+18, Eag22, DRZ20, BGH19], which we believe are all valid and can be adapted to accomodated for the public "twist". The particularity the one we describe is that it is tailored to rely solely on Σ-protocols and that, in terms of concrete optimizations, uses the least number of group operations possible (cf. Appendix A.2). More formally, we provide a protocol for the relation

$$\mathsf{R}_{tsp} \coloneqq \left\{ ((\vec{f},\psi,\vec{e},\varepsilon,y,\psi),\vec{v},F,E,Y) : \begin{array}{c} F = \sum_{i=1}^n f_iG_i + \phi H \ \wedge\ E = \sum_{i=1}^n e_iG_i + \varepsilon H \ \wedge\ Y = yG + \psi H \\ y = \langle \vec{f}, \vec{v}\circ\vec{e}\rangle \end{array} \right\}.$$

Sub-protocols. The protocol internally relies on two sub-protocols:

- Π_{lin}, illustrated in Figure 7, for proving in zero-knowledge statements for the linear evaluation relation

$$\mathsf{R}_{lin} := \left\{ ((\vec{f}, \phi, y, \psi), (F, \vec{e}, Y)) : F = \sum_i f_i G_i + \phi H \ \wedge \ Y = yG + \psi H \ \wedge \ \langle \vec{f}, \vec{e} \rangle = y \right\} .$$

  In the language of Maurer proofs [Mau09], the protocol proves knowledge of the opening of two Pedersen commitments F, Y satisfying the linear relation

$$\begin{bmatrix} F \\ Y \end{bmatrix} = \begin{bmatrix} G_1 & \cdots & G_n & H & 0 \\ e_1 G & \cdots & e_n G & 0 & H \end{bmatrix} \cdot \begin{bmatrix} f_1 & \cdots & f_n & \phi & \psi \end{bmatrix}^t$$

  Π_{lin} be found in schoolbook literature Boneh–Shoup [BS20], but more recent works instantiate it with log-size proof size, see e.g. the work of Gentry, Halevi, and Lyubashevsky [GHL22, Section E] and the work of Cramer and Attema [AC20, Protocol 2].[19]

- Π_{mul}, illustrated in Figure 8, for proving in zero-knowledge statements for the multiplication relation

$$\mathsf{R}_{mul} := \left\{ ((a, \alpha, b, \beta, y, \psi), A, B, Y) : \begin{array}{c} A = aG + \alpha H \ \wedge \ B = bG + \beta H \ \wedge \ Y = yG + \psi H \\ y = ab \end{array} \right\} .$$

  Π_{mul} is found in the work of Maurer [Mau09, Sec. 6.7], and reformulated also afterwards e.g. by Damgård et al. [DLO+18], with a minor change to allow for knowledge soundness.The extractor of Maurer's protocol, in fact, only allows to extract one of the two multipliands (which is sufficient for proving the relation is sound). To prove that Y commits to the product of the values committed to as A, B, the prover shows knowledge of a, α s.t. $A = aG + \alpha H$ and that $Y - aB$ is a commitment to zero. To extract an opening for Y and B, in Figure 8 we perform paralle AND composition with representation proof for B.

The sumcheck protocol. Lund et al. [LFKN90] provide an interactive sumcheck protocol for proving R_{tsp} with soundness error is $O(\log n/|\mathbb{F}|)$; round complexity is $O(\log n)$; prover time $O(n)$; and verifier time $O(\log n)$. It can be made non-interactive with some loss in tightness by applying the Fiat-Shamir transform. After $\log n$ rounds of interaction the verifier is convinced provided that

$$\begin{aligned} \langle \vec{f}, \otimes_j (1, c_j) \rangle &= u_1 \\ \langle \vec{e}, \otimes_j (1, c_j) \rangle &= u_2 \end{aligned} \tag{17}$$

which can be proven using Π_{lin} described in Figure 7.[20] In this section we describe some background behind the knowledge-sound protocol, before providing a zero-knowledge variant in Appendix B.

Consider two vectors $\vec{f}, \vec{e} \in \mathbb{F}^n$ with $\langle \vec{f}, \vec{e} \rangle = y$. Denote with $\mathsf{p}f$ the (monomial) multilinear encoding of $\vec{f}$

$$\mathsf{p}f(X_1, \ldots, X_{\log n}) := \sum_{\vec{b} \in \{0,1\}^{\log n}} f_{\sum_{j=0}^{\log n} b_j 2^j} X_1^{b_1} X_2^{b_2} \cdots X_{\log n}^{b_{\log n}} \tag{18}$$

In other words, the element in the $(\sum_j 2^j b_j)$-th position in $\vec{f}$ is the coefficient of $X_1^{b_1} \cdots X_{\log n}^{b_{\log n}}$ in $\mathsf{p}f$. Note that $\mathsf{p}f$ is of degree at most one in each of X_i for $i = 1, \ldots, \log n$. Prior work remarked that:

Lemma 18 ([Tha13, XZZ+19, BCG20a]). *Let $\vec{f}, \vec{e} \in \mathbb{F}^n$ be vectors of length n. Denote with $\mathsf{p}f, \mathsf{p}e$ the polynomials encoding $\vec{f}, \vec{e}$ respectively. Then*

$$\sum_{w \in \{-1,1\}^{\log n}} \mathsf{p}f(w) \cdot \mathsf{p}e(w) \cdot w_1 \cdots w_{\log n} = 2^{\log n} \cdot \langle \vec{f}, \vec{e} \rangle \tag{19}$$

One can use the sumcheck protocol for Equation (19). The sumcheck protocol is inductive and proceeds over $\log n$ rounds. In the first round the prover sends the coefficients of degree-2 polynomials

$$P_1(X) := \sum_{\vec{w} \in \{-1,1\}^{\log n - 1}} p_{1,\vec{w}}(w_1, \ldots, w_{\log n - 1}, X) \cdot w_1 \cdots w_{\log n - 1} X$$

[19] To use Protocol 2 from Cramer and Attema [AC20], one would prove that $F + Y$ commits to a vector such whose inner-product with $(\vec{e}, -1)$ is zero. In order to properly extract the opening information of F, Y one would also have to send a representation proof of Y.

[20] As an example: $\otimes_{j=0}^{2}(1, c_j) = (1, c_0, c_1, c_1 c_0, c_2, c_2 c_0, c_2 c_1, c_2 c_1 c_0)$

In the first round $p_{1,\vec{w}}$ are the coefficients of the polynomial defined in Equation (18). The verifier checks that $P_1(1) + P_1(-1) = 2^{\log n} \cdot \langle \vec{f}, \vec{e} \rangle$ is equal to the claimed sum. It then suffices to show that $P_1(X)$ is computed correctly. To do the verifier sends a random challenge $c_1 \leftarrow\!\!\$\ \mathbb{F}$ and is satisfied if

$$P_1(c_1) = \sum_{\vec{w} \in \{-1,1\}^{\log n - 1}} p_{1,\vec{w}}(w_1, \ldots, w_{\log n - 1}, c_1) \cdot w_1 \cdots w_{\log n - 1} c_1$$

Rather than check this statement directly, the verifier proceeds with further rounds of interaction. More generally, denote

$$P_i(X) = \sum_{\vec{w} \in \{-1,1\}^{\log n - i}} p_{i,\vec{w}}(w_1, \ldots, w_{\log n - i}, X, c_{i-1}, \ldots, c_1) \cdot w_1 \cdots w_{\log n - i} X c_{i-1} \cdots c_1$$

where $p_{i,\vec{w}}$ are the coefficients corresponding to the partial evaluation of $\mathsf{p}f$ in $\vec{c}$, i.e.: $\mathsf{p}p_{i,\vec{w}} = \mathsf{p}f(\vec{w}, \vec{c}) \cdot \mathsf{p}e(\vec{w}, \vec{c})$. At this point, the verifier sends a random challenge $c_i \leftarrow\!\!\$\ \mathbb{F}$ and the prover considers (recursively) the sumcheck claim

$$\sum_{\vec{w} \in \{-1,1\}^{i-1}} p_{i+1,\vec{w}}(w_1, \ldots, w_{i-1}, c_i, \ldots, c_{\log n}) = P_i(c_i)$$

After $\log n$ rounds, the prover is left with two constant terms $u_1 \coloneqq \mathsf{p}f(c_1, \ldots, c_{\log n})$, $u_2 \coloneqq \mathsf{p}e(c_1, \ldots, c_{\log n})$ that are sent to the verifier, which checks that $u_1 \cdot u_2 = P_{\log n}(c_{\log n})$. We are left with proving that u_1, u_2 are indeed correct, i.e. with the so-called *tensorcheck claim* [BCHO22]

$$\begin{aligned} \langle \vec{f}, \otimes_j (1, c_j) \rangle &= u_1 \\ \langle \vec{e}, \otimes_j (1, c_j) \rangle &= u_2 \end{aligned} \tag{20}$$

where

$$\langle \vec{f}, \otimes_j (1, c_j) \rangle = \sum_{\vec{b} \in \{0,1\}^{\log n}} f_{\sum_{j=0}^{\log n} b_j 2^j} c_1^{b_1} c_2^{b_2} \cdots c_{\log n}^{b_{\log n}} \tag{21}$$

A zero-knowledge sumcheck. The protocol of Lund et al. [LFKN90] described in is an interactive protocol that by itself does not achieve zero-knowledge. The prover leaks at a logarithmic number of linear evaluations of the witness. We describe a zero-knowledge sumcheck argument for R_{tsp} in Figure 6 and prove security in Theorem 19 and Theorem 20.

To achieve zero-knowledge we employ a commitment scheme to commit each sumcheck message, leveraging the hiding property. In order to preserve soundness, the prover must also prove knowledge of a valid opening for which sumcheck verification holds in the final step of the protocol. Additionally, we extend the inner-product relation with a *twist*, to prove that

$$y = \langle \vec{f}, \vec{v} \circ \vec{e} \rangle$$

with respect to public $\vec{v}$ as opposed to just checking that $y = \langle \vec{f}, \vec{e} \rangle$. This is done by incorporating $\vec{v}$ into the final check

$$\langle \vec{e}, \vec{v} \circ (\otimes_j (1, c_j)) \rangle = u_2$$

B.1 Analysis

Theorem 19. *Suppose that Π_{tsp} is instantiated with knowledge-sound protocols Π_{lin} and Π_{mul}. Then Π_{tsp} is knowledge-sound if the discrete logarithm assumption is hard.*

Proof. Consider an extractor Ext that behaves as follows.

- Run the extractor of Π_{lin} on the proof for $(\otimes_j (1, c_j), F, U_1)$ to extract some $(\vec{f}^{(0)}, \phi, u_1, \psi_1)$ such that $F = \sum f_i G_i + \phi H$ and $U_1 = u_1 G + \psi_1 H$ and $u_1 = \langle \otimes_j (1, c_j), \vec{f} \rangle$.
- Run the extractor of Π_{lin} on the proof for $(\vec{v} \circ (\otimes_j (1, c_j)), E, U_2)$ to extract some $(\vec{e}^{(0)}, \epsilon, u_2, \psi_2)$ such that $E = \sum e_i G_i + \epsilon H$ and $U_2 = u_2 G + \psi_2 H$ and $u_2 = \langle \vec{v} \circ \otimes_j (1, c_j), \vec{e} \rangle$.
- Run the extractor of Π_{mul} on the proof for (U_1, U_2, Y) to extract some $(u_1', \psi_1', u_2', \psi_2', y, \psi)$ such that $U_1 = u_1' G + \psi_1' H$, $U_2 = u_2' G + \psi_2' H$ and $Y = yG + \psi H$.

- Runs the following extractor Ext^* to get representations $\{(a_j, b_j, \alpha_j, \beta_j, y_j, \gamma_j)\}_{j=1}^{\log n}$ such that $A_j = a_j G + \alpha_j H$ and $B_j = b_j G + \beta_j H$ and $Y_j = y_j G + \gamma_j H$.

  Initially let $\mathsf{Ext}_{\log n}$ be the extractor that runs the extractor of Π_{mul} to get (y, ψ) such that $Y_{\log n} = yG + \psi H$ and returns $(\emptyset, (y, \psi))$.

  Let Ext_i be an extractor that returns representations $\left(\{(a_j, b_j, \alpha_j, \beta_j)\}_{j=i}^{\log n}, (y, \psi)\right)$. Then by running the adversary 3 times Ext_{i-1} learns

$$\begin{aligned} Y_i &= A_{i-1} + c_{i-1} B_{i-1} + c_{i-1}^2 (Y_{i-1} - A_{i-1}) \\ Y_i' &= A_{i-1} + c_{i-1}' B_{i-1} + (c_{i-1}')^2 (Y_{i-1} - A_{i-1}) \\ Y_i'' &= A_{i-1} + c_{i-1}'' B_{i-1} + (c_{i-1}'')^2 (Y_{i-1} - A_{i-1}) \end{aligned}$$

  Then Ext_{i-1} runs Ext_i with respect to Y_i, Y_i', Y_i'' and backwards solves to compute $(y_i, \psi_i, y_i', \psi_i', y_i'', \psi_i'')$ such that

$$\begin{aligned} y_i G + \psi_i H &= A_{i-1} + c_{i-1} B_{i-1} + c_{i-1}^2 (Y_{i-1} - A_{i-1}) \\ y_i' G + \psi_i' H &= A_{i-1} + c_{i-1}' B_{i-1} + (c_{i-1}')^2 (Y_{i-1} - A_{i-1}) \\ y_i'' G + \psi_i'' H &= A_{i-1} + c_{i-1}'' B_{i-1} + (c_{i-1}'')^2 (Y_{i-1} - A_{i-1}) \end{aligned}$$

  Indeed, starting with $y_{\log n} = y$ we have that

$$\begin{aligned} &\text{for } \log n \geq k \geq i : \\ &\qquad y_{k-1} = \frac{1}{c_{k-1}} \left(y_k + c_{k-1} a_{k-1} - a_{k-1} - c_{k-1} b_{k-1}\right) \\ &\qquad \psi_{k-1} = \frac{1}{c_{k-1}} \left(\psi_k + c_{k-1} \alpha_{k-1} - \alpha_{k-1} - c_{k-1} \beta_{k-1}\right) \end{aligned}$$

  Then using Gaussian elimination Ext_{i-1} computes $(a_{i-1}, b_{i-1}, \alpha_{i-1}, \beta_{i-1})$. Thus Ext_1 is considered the full extractor that returns the full set of representations.

- Return $(\vec{f}, \phi, \vec{e}, \varepsilon, y_1, \psi_1)$

We must show that Ext returns a valid witness for the relation R_{tsp}.

$\mathbf{H}_1$ This game behaves identically to the original knowledge soundness game for Π_{tsp} except that if the extractor of Π_{lin} for $(\otimes_j(1, c_j), F, U_1)$ fails extract some $(\vec{f}^{(0)}, \phi, u_1, \psi_1)$ with $((\vec{f}^{(0)}, \phi, u_1, \psi_1), \otimes_j(1, c_j), F, U_1) \in \Pi_{lin}$ then the game aborts. This game is indistinguishable by the knowledge soundness of Π_{lin}.

$\mathbf{H}_2$ This game behaves identically to the previous except that if the extractor of Π_{lin} for $(\vec{v} \circ \otimes_j(1, c_j), E, U_2)$ fails extract some $(\vec{e}^{(0)}, \epsilon, u_2, \psi_2)$ with $((\vec{e}^{(0)}, \epsilon, u_2, \psi_2), \vec{v} \circ (\otimes_j(1, c_j)), E, U_2) \in \Pi_{lin}$ then the game aborts. This game is indistinguishable by the knowledge soundness of Π_{lin}.

$\mathbf{H}_3$ If the extractor of Π_{mul} for (U_1, U_2, Y) fails extract some $(u_1', \psi_1', u_2', \psi_2', y, \psi)$ with $((u_1', \psi_1', u_2', \psi_2', y, \psi), U_1, U_2, Y) \in \Pi_{mul}$ then the game aborts. This game is indistinguishable by the knowledge soundness of Π_{mul}.

$\mathbf{H}_4$ If $(u_1', \psi_1') \neq (u_1, \psi_1)$ or $(u_2', \psi_2') \neq (u_2, \psi_2)$ then the game aborts. This game is indistinguishable if the discrete logarithm problem is hard. Indeed if this game aborts when the previous one doesn't then

$$U_1 = u_1 G + \psi_1 H = u_1' G + \psi_1' H \text{ and } U_2 = u_2 G + \psi_2 H = u_2' G + \psi_2' H$$

Hence a reduction that gets given then challenge discrete logarithm (G, H) can run the adversary $\mathcal{A}$ and then the extractor of Π_{tsp} to get $(u_1', \psi_1', u_1, \psi_1, u_2', \psi_2', u_2, \psi_2)$. The reduction then returns either

$$x = \frac{u_1 - u_1'}{\psi_1' - \psi_1} \text{ or } x = \frac{u_2 - u_2'}{\psi_2' - \psi_2}$$

such that $H = xG$.

$\mathbf{H}_5$ This game aborts if Ext^* fails. The extractor succeeds at Gaussian elimination provided that

$$\begin{pmatrix} 1-c_{i-1}^2 & c_{i-1} & c_{i-1}^2 \\ 1-(c'_{i-1})^2 & c_{i-1} & (c'_{i-1})^2 \\ 1-(c''_{i-1})^2 & c_{i-1} & (c''_{i-1})^2 \end{pmatrix}$$

is invertible for $1 \leq i \leq \log n$. By the Schwartz-Zippel Lemma this is true in each round except with probability $\frac{8}{|\mathbb{F}|}$. By the Forking Lemma [BCC+16] this extractor runs in polynomial time.

Finally one must argue that it is statistically unlikely that

$$\langle \otimes_j (1, c_j), \vec{f} \rangle \langle \vec{v} \circ (\otimes_j (1, c_j)), \vec{e} \rangle = y_{\log n}$$

unless $y_0 = \langle \vec{f}, \vec{v} \circ \vec{e} \rangle$. We proceed via induction.

First observe that for any vector $\vec{d}$, $\vec{e}$, $\vec{v}$

$$\langle \vec{v} \circ \vec{d}, \vec{e} \rangle = \sum_j v_j d_j e_j = \langle \vec{d}, \vec{v} \circ \vec{e} \rangle$$

Thus it suffices to show that

$$\langle \otimes_j (1, c_j), \vec{f} \rangle \langle (\otimes_j (1, c_j)), \vec{s} \rangle = y_{\log n} \;\Rightarrow \langle \vec{f}, \vec{s} \rangle = y$$

for $\vec{s} = \vec{v} \circ \vec{e}$.

Base case. The $a_{\log n}, b_{\log n}, y_{\log n}$ satisfy

$$\begin{aligned} & a_{\log n} + b_{\log n} X + (y_{\log n} - a_{\log n}) X^2 \\ &= \langle \otimes_{j=1}^{\log n - 1} (1, c_j), (f_{0,\gamma})_{\gamma \in \{0,1\}^{\log n - 1}} \rangle \langle \otimes_{j=1}^{\log n - 1} (1, c_j), (s_{0,\gamma})_{\gamma \in \{0,1\}^{\log n - 1}} \rangle + b_{\log n} X \\ &+ X^2 \sum_{\beta \in \{0,1\}^{\log(n)-1}} \langle \otimes_{j=1}^{i-1} (1, c_j), (f_{1,\gamma})_{\gamma \in \{0,1\}^{\log n - 1}} \rangle \langle \otimes_{j=1}^{\log n - 1} (1, c_j), (s_{1,\gamma})_{\gamma \in \{0,1\}^{\log n - 1}} \rangle \end{aligned}$$

except with negligible probability.

Base case proof. Where the verifier is satisfied we have that for $i = \log n$

$$a_{\log n} + b_{\log n} c_{\log n} + (y_{\log n} - a_{\log n}) c_{\log n}^2 = \langle \otimes_{j=1}^{\log n} (1, c_j), (f_\gamma)_{\gamma \in \{0,1\}^{\log n}} \rangle \langle \otimes_{j=1}^{\log n} (1, c_j), (s_\gamma)_{\gamma \in \{0,1\}^{\log n}} \rangle$$

which we can rearrange to equal

$$\begin{aligned} =& \langle \otimes_{j=1}^{\log n - 1} (1, c_j), (f_{0,\gamma})_{\gamma \in \{0,1\}^{\log n - 1}} \rangle \langle \otimes_{j=1}^{\log n - 1} (1, c_j), (s_{0,\gamma})_{\gamma \in \{0,1\}^{\log n - 1}} \rangle + d_{\log n} c_{\log n} \\ &+ c_{\log n}^2 \langle \otimes_{j=1}^{\log n - 1} (1, c_j), (f_{1,\gamma})_{\gamma \in \{0,1\}^{\log n - 1}} \rangle \langle \otimes_{j=1}^{\log n - 1} (1, c_j), (s_{1,\gamma})_{\gamma \in \{0,1\}^{\log n - 1}} \rangle \end{aligned}$$

Here $d_{\log n}$ is independent from $c_{\log n}$. Where $a_{\log n}, b_{\log n}, y_{\log n}$ are independent from $c_{\log n}$, the probability of this occurring is $\frac{2}{p}$ unless the base claim holds.

Induction claim. Suppose for $i \in [1, \log n]$ we have that $a_i + b_i c_i + (y_{i-1} - a_i) c_i^2 = y_i$ for c_i sampled only after a_i, b_i are determined. The probability that a_i, b_i, y_{i-1} satisfy

$$\begin{aligned} a_i + b_i X + (y_{i-1} - a_i) X^2 = & \sum_{\beta \in \{0,1\}^{\log(n)-i}} \langle \otimes_{j=1}^{i-1} (1, c_j), (f_{\beta,0,\gamma})_{\gamma \in \{0,1\}^{i-1}} \rangle \langle \otimes_{j=1}^{i-1} (1, c_j), (s_{\beta,0,\gamma})_{\gamma \in \{0,1\}^{i-1}} \rangle + b_i X \\ & + X^2 \sum_{\beta \in \{0,1\}^{\log(n)-i}} \langle \otimes_{j=1}^{i-1} (1, c_j), (f_{\beta,1,\gamma})_{\gamma \in \{0,1\}^{i-1}} \rangle \langle \otimes_{j=1}^{i-1} (1, c_j), (s_{\beta,1,\gamma})_{\gamma \in \{0,1\}^{i-1}} \rangle \end{aligned} \tag{22}$$

is negligible unless $a_{i-1}, b_{i-1}, y_{i-2}$ also satisfy.

$$\begin{aligned} & a_{i-1} + b_{i-1} X + (y_{i-2} - a_{i-1}) X^2 \\ &= \sum_{\beta \in \{0,1\}^{\log(n)-(i-1)}} \langle \otimes_{j=1}^{i-2} (1, c_j), (f_{\beta,0,\gamma})_{\gamma \in \{0,1\}^{i-2}} \rangle \langle \otimes_{j=1}^{i-2} (1, c_j), (s_{\beta,0,\gamma})_{\gamma \in \{0,1\}^{i-2}} \rangle + b_{i-1} X \\ &+ X^2 \sum_{\beta \in \{0,1\}^{\log(n)-(i-1)}} \langle \otimes_{j=1}^{i-2} (1, c_j), (f_{\beta,1,\gamma})_{\gamma \in \{0,1\}^{i-2}} \rangle \langle \otimes_{j=1}^{i-2} (1, c_j), (s_{\beta,1,\gamma})_{\gamma \in \{0,1\}^{i-2}} \rangle \end{aligned}$$

Induction claim proof. If Equation (22) holds then

$$a_i = \sum_{\beta \in \{0,1\}^{\log(n)-i}} \langle \otimes_{j=1}^{i}(1, c_j), (f_{\beta,0,\gamma})_{\gamma \in \{0,1\}^{i-1}} \rangle \langle \otimes_{j=1}^{i}(1, c_j), (s_{\beta,0,\gamma})_{\gamma \in \{0,1\}^{i-1}} \rangle$$

and hence

$$\begin{aligned} y_{i-1} &= \sum_{\beta \in \{0,1\}^{\log(n)-i}} \langle \otimes_{j=1}^{i}(1, c_j), (f_{\beta,0,\gamma})_{\gamma \in \{0,1\}^{i-1}} \rangle \langle \otimes_{j=1}^{i}(1, c_j), (s_{\beta,0,\gamma})_{\gamma \in \{0,1\}^{i-1}} \rangle \\ &+ \sum_{\beta \in \{0,1\}^{\log(n)-i}} \langle \otimes_{j=1}^{i}(1, c_j), (f_{\beta,1,\gamma})_{\gamma \in \{0,1\}^{i-1}} \rangle \langle \otimes_{j=1}^{i}(1, c_j), (s_{\beta,1,\gamma})_{\gamma \in \{0,1\}^{i-1}} \rangle \\ &= \sum_{\beta \in \{0,1\}^{\log(n)-(i-1)}} \langle \otimes_{j=1}^{i}(1, c_j), (f_{\beta,\gamma})_{\gamma \in \{0,1\}^{i-1}} \rangle \langle \otimes_{j=1}^{i}(1, c_j), (s_{\beta,\gamma})_{\gamma \in \{0,1\}^{i-1}} \rangle \end{aligned}$$

Now we can rewrite this expression for y_{i-1} as

$$\begin{aligned} y_{i-1} = &\sum_{\beta \in \{0,1\}^{\log(n)-(i-1)}} \langle \otimes_{j=1}^{i-1}(1, c_j), (f_{\beta,0,\gamma})_{\gamma \in \{0,1\}^{i-2}} \rangle \langle \otimes_{j=1}^{i-1}(1, c_j), (s_{\beta,0,\gamma})_{\gamma \in \{0,1\}^{i-2}} \rangle \\ &+ c_i d_{i-1} + c_i^2 \sum_{\beta \in \{0,1\}^{\log(n)-(i-1)}} \langle \otimes_{j=1}^{i-1}(1, c_j), (f_{\beta,1,\gamma})_{\gamma \in \{0,1\}^{i-2}} \rangle \langle \otimes_{j=1}^{i-1}(1, c_j), (s_{\beta,1,\gamma})_{\gamma \in \{0,1\}^{i-2}} \rangle \end{aligned}$$

where d_{i-1} is some value not depending on c_i. Hence

$$\begin{aligned} &a_{i-1} + b_{i-1} c_i + (y_{i-2} - a_{i-1})(c_i)^2 \\ &= \sum_{\beta \in \{0,1\}^{\log(n)-(i-1)}} \langle \otimes_{j=1}^{i-1}(1, c_j), (f_{\beta,0,\gamma})_{\gamma \in \{0,1\}^{i-2}} \rangle \langle \otimes_{j=1}^{i-1}(1, c_j), (s_{\beta,0,\gamma})_{\gamma \in \{0,1\}^{i-2}} \rangle \\ &+ c_i d_{i-1} + c_i^2 \sum_{\beta \in \{0,1\}^{\log(n)-(i-1)}} \langle \otimes_{j=1}^{i-1}(1, c_j), (f_{\beta,1,\gamma})_{\gamma \in \{0,1\}^{i-2}} \rangle \langle \otimes_{j=1}^{i-1}(1, c_j), (s_{\beta,1,\gamma})_{\gamma \in \{0,1\}^{i-2}} \rangle \end{aligned}$$

The probability of this occurring for c_i sampled at random is $\frac{2}{p}$ unless the induction claim holds.

Summary Putting everything together, we see that except with negligible probability

$$y = y_0 = \sum_{\beta \in \{0,1\}^{\log(n)-1}} f_\beta s_\beta = \sum_{\beta \in \{0,1\}^{\log(n)-1}} v_\beta f_\beta e_\beta$$

as required. □

Theorem 20. *Suppose Π_{tsp} from Figure 5 is instantiated with honest-verifier zero-knowledge Π_{lin} and Π_{mul} protocols for the relations R_{lin} and R_{mul} respectively. Then Π_{tsp} is honest-verifier zero-knowledge.*

Proof. Consider the simulator that behaves as follows

1. Take as input $(\vec{v}, F, E, Y)$.
2. For $j \in [1, \log n]$ sample $A_j, B_j \leftarrow\!\!\$\ \mathbb{G}$ and $c_j \leftarrow\!\!\$\ \mathbb{Z}_p$.
3. Set $Y_0 = Y$. For $j \in [1, \log n]$ set $Y_{j+1} = A_j + c_j B_j + c_j^2 (Y_{j-1} - A_j)$
4. For the final round messages
   (a) Set $\vec{c} = \otimes_j (1, c_j)$
   (b) Choose $U_1, U_2 \leftarrow\!\!\$\ \mathbb{G}$ randomly
   (c) Compute $\pi_{lin,hf} \leftarrow \Pi_{lin}.\mathsf{Sim}(\vec{c}, F, U_1)$
   (d) Compute $\pi_{lin,e} \leftarrow \Pi_{lin}.\mathsf{Sim}(\vec{c}, E, U_2)$
   (e) Compute $\pi_{mul} \leftarrow \Pi_{mul}.\mathsf{Sim}(U_1, U_2, Y_{\log n})$

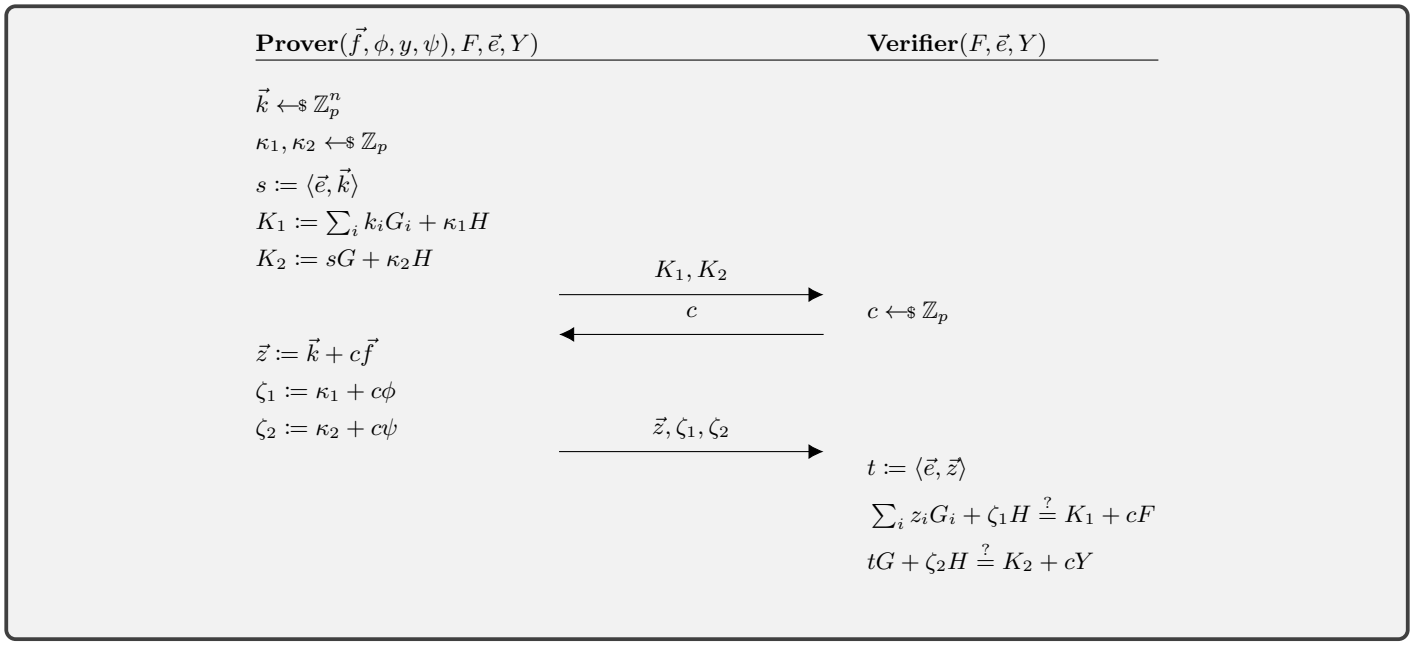

Figure 7: Protocol Π_{lin} for proving linear eval.: knowledge of $\vec{f}, y$ (committed in F, Y) such that $\langle \vec{f}, \vec{e} \rangle = y$. The protocol is used both during Π_{lup} and during Π_{tsp}.

5. Return the transcript
$$(\vec{v}, F, E, Y), \{(A_j, B_j), c_j\}_{j=1}^{\log n}, (U_1, U_2, \pi_{lin,f}, \pi_{lin,e}, \pi_{mul})$$

We argue that this simulated transcript is indistinguishable a transcript produced by an honest prover and verifier in the zero-knowledge game via a series of Hybrids.

- $\mathbf{H}_1$ This game behaves identically to the original zero-knowledge game for Π_{lup} except that when the honest prover is run, the proof $\pi_{lin,f}$ is computed as $\pi_{lin,f} \leftarrow \Pi_{lin}.\mathsf{Sim}(\vec{c}, F, U_1)$. Since the honest prover knows a witness for $\vec{c}, F, U_1$, we can build a reduction against the zero-knowledge of Π_{lin} should this change be distinguishable.
- $\mathbf{H}_2$ The proof $\pi_{lin,e}$ is computed as $\pi_{lin,q} \leftarrow \Pi_{lin}.\mathsf{Sim}(\vec{c}, E, U_2)$. Since the honest prover knows a witness for $(\vec{c}, E, U_2)$, we can build a reduction against the zero-knowledge of Π_{lin} should this change be distinguishable.
- $\mathbf{H}_3$ The proof π_{mul} is computed as $\pi_{mul} \leftarrow \Pi_{mul}.\mathsf{Sim}(U_1, U_2, Y_{\log n})$. Since the honest prover knows a witness for $(U_1, U_2, Y_{\log n})$, we can build a reduction against the zero-knowledge of Π_{tsp} should this change be distinguishable.

We now see that the final hybrid is statistically impossible. Indeed, the only proof elements that are non-identical are $(A_j, B_j)_{j=1}^{\log n}$, U_1 and U_2. However these elements are computed uniformly at random both by the honest prover (due to the blinders α_j, β_j, ψ_1, ψ_2) and the simulator. □

Informally the protocol behaves as follows. First the prover chooses random $R = rG + \rho H$ used to mask A, and $S = sG + \sigma H$ used to mask B, and $T = -rbG + \tau H$ where r is the same value as in R and where b is the value committed in B. It sends these values to the verifier receiving back random c as a response.

Then the prover sends $u = r + ca$ and $\mu = \rho + c\alpha$. The verifier checks correctness i.e. that $R + cA = (rG + \rho H) + c(aG + \alpha H) = uG + \mu H$, thus convincing itself that $u = r + ca$. The prover proves knowledge of the discrete logarithm of B with $S = sG + \sigma H$, $v = s + cb$, and $\nu = \sigma + c\beta$.

The prover also sends $\omega = u\beta + \tau - c\psi$ and the verifier checks that

$$uB + T - cY = u(bG + \beta H) + (-rbG + \tau H) - c(yG + \psi H) = (ub - rb - cy)G + (u\beta + \tau - c\psi) \stackrel{?}{=} \omega H$$

thus getting convinced that $(ub - rb - cy) = 0$. Substituting $u = r + ca$ we see

$$(r + ca)b - rb - cy = c(ab - y) = 0$$

as required. Note that if the prover chooses $T \neq -rbG + \tau H$ then with high probability verification will not pass for random c.

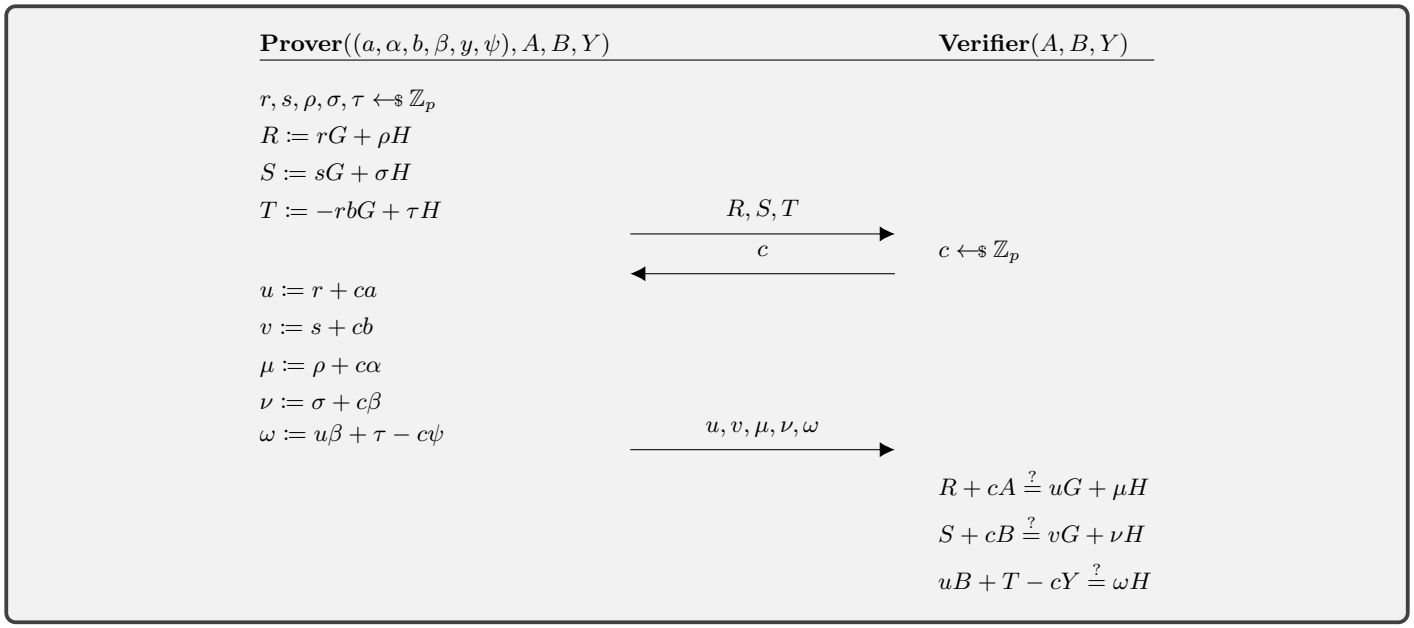

Figure 8: The zero-knowledge protocol Π_{mul} for proving that $A = aG + \alpha H$, $B = bG + \beta H$, $Y = abG + \psi H$. This protocol is used during Π_{tsp}.

C AES witness generation

C.1 Key schedule

Rijndael is a block cipher that operates on a fixed block that we denote st and a key size of ksz over a number of rounds $rnds$. Key expansion is given as input the key $k \in \{0,1\}^{ksz}$ and outputs the rounds keys $rk_0, \ldots, rk_{rnds} \in (\mathbb{F}_2^8)^{16}$.

In AES we have that key sizes have either 128, 192 or 256 bits: $ksz \in \{128, 192, 256\}$ and $k \in \{0,1\}^{ksz}$. For simplicity, in this description we shall assume a key size of $ksz = 128$ bits. The number of rounds is either 10, 12, or 14: $rnds \in \{10, 12, 14\}$. We split the trace into bytes. In our implementation these bytes are further reduced into two 4 bit segments. These segments always appear together thus we here ignore this detail.

Format of the key expansion trace. The key scheduling execution trace contains 2 intermediary states over all rounds $rnds$.

$$\vec{keytr} = \left(1, \vec{ksch}_{0,0}, \vec{ksch}_{0,1}, \ldots, \vec{ksch}_{rnds,0}, \vec{ksch}_{rnds,1}\right)$$

The 16 byte rounds keys are determined by $rk_i = \vec{ksch}_{i,0}$. The states $\vec{ksch}_{i,1}$ contain auxiliary information useful for checking the correctness of the expansion algorithm. Each intermediary state $\vec{ksch}_{i,j}$ contains 16 bytes arranged in a 4x4 grid of bytes

$$\vec{ksch}_{i,j} = \begin{pmatrix} \vec{ksch}_{i,j,0,0} & \vec{ksch}_{i,j,0,1} & \vec{ksch}_{i,j,0,2} & \vec{ksch}_{i,j,0,3} \\ \vec{ksch}_{i,j,1,0} & \vec{ksch}_{i,j,1,1} & \vec{ksch}_{i,j,1,2} & \vec{ksch}_{i,j,1,3} \\ \vec{ksch}_{i,j,2,0} & \vec{ksch}_{i,j,2,1} & \vec{ksch}_{i,j,2,2} & \vec{ksch}_{i,j,2,3} \\ \vec{ksch}_{i,j,3,0} & \vec{ksch}_{i,j,3,1} & \vec{ksch}_{i,j,3,2} & \vec{ksch}_{i,j,3,3} \end{pmatrix}$$

where $\vec{ksch}_{i,j,k,\ell} \in \{0,1\}^8$. We can interpret $\vec{ksch}_{i,j,k,\ell} \in \mathbb{F}_2^8$ because $\{0,1\}^8$ and $\mathbb{F}_2^8$ are bijective.

The key expansion protocol. The key schedule is generated over 32 bit words with

$$kword_{i,j,\ell} = (\vec{ksch}_{i,j,0,\ell} \parallel \vec{ksch}_{i,j,1,\ell} \parallel \vec{ksch}_{i,j,2,\ell} \parallel \vec{ksch}_{i,j,3,\ell})$$

Then we have that

- $\vec{ksch}_{0,0,k,\ell} = k_{4k+\ell}$. Namely the first ksz bits of the keys are copied in the key schedule state. We do not have to prove this equality, we simply define the k to be the concatenation of $\vec{ksch}_{0,0,k,\ell}$.

- $kword_{i,0,0} = kword_{i-1,0,0} \oplus \mathsf{sbox}(\mathsf{RotWord}(kword_{i-1,0,3})) \oplus rcon_i$ if $i \geq 1$
- $kword_{i,0,\ell} = kword_{i-1,0,\ell} \oplus kword_{i,0,\ell-1}$ if $i \geq 1$ and $\ell \neq 0$.

Here $\mathsf{RotWord}([b_0, b_1, b_2, b_3]) = [b_1, b_2, b_3, b_0]$ is a one byte left circular shift. The constant words $rcon_i$ are known to the verifier.

We can directly lookup whether $kword_{i,0,\ell} = kword_{i-1,0,\ell} \oplus kword_{i,0,\ell-1}$. However, checking whether $kword_{i,0,0} = kword_{i-1,0,0} \oplus \mathsf{sbox}(\mathsf{RotWord}(kword_{i-1,0,3})) \oplus rcon_i$ requires multiple steps. The prover thus sets

- $kword_{i,1,0} = \mathsf{sbox}(\mathsf{RotWord}(kword_{i-1,0,3}))$
- $kword_{i,1,1} = kword_{i,1,0} \oplus rcon_i$
- $kword_{i,0,0} = kword_{i-1,0,0} \oplus kword_{i,1,1}$ if $i > 0$.

The $\vec{ksch}_{i,j,k,\ell}$ are chosen such that the key words $kword_{i,\ell}$ satisfy the above constraints. We are now ready to define our selection matrices for the key expansion algorithm.

Selection matrices for key expansion. To generate the $S_{xor,O}$, $S_{xor,L}$, and $S_{xor,R}$ matrices do the following.

- Initialise $n = 0$
- For $1 \leq i \leq rnds$, $0 \leq k < 4$, $1 \leq \ell < 4$ then:
  - $S_{xor,O}[n, 1 + 32i + 4k + \ell] = 1$; $S_{xor,L}[n, 1 + 32(i-1) + 4k + \ell] = 1$; $S_{xor,R}[n, 1 + 32i + 4k + (\ell - 1)] = 1$;
  - Increment $n = n + 1$. # To show $kword_{i,0,\ell} = kword_{i-1,0,\ell} \oplus kword_{i,0,\ell-1}$ for $i \geq 1$ and $\ell \neq 0$
- For $1 \leq i \leq rnds$, $0 \leq k < 4$ then:
  - $S_{xor,O}[n, 1 + 32i + 16 + 4k + 1] = 1$; $S_{xor,L}[n, 1 + 32i + 16 + 4k] = 1$; $S_{xor,R}[n, 1] = rcon_i$;
  - Increment $n = n + 1$. # To show $kword_{i,1,1} = kword_{i,1,0} \oplus rcon_i$
- For $1 \leq i \leq rnds$, $0 \leq k < 4$ then:
  - $S_{xor,O}[n, 1 + 32i + 4k] = 1$; $S_{xor,L}[n, 1 + 32(i-1) + 4k] = 1$; $S_{xor,R}[n, 1 + 32i + 16 + 4k + 1] = rcon_i$;
  - Increment $n = n + 1$. # To show $kword_{i,0,0} = kword_{i-1,0,0} \oplus kword_{i,1,1}$ if $i > 0$
- Set all other entries to 0

To generate the $S_{sbox,O}$, $S_{sbox,I}$ matrices do the following.

- Initialise $n = 0$
- For $0 \leq i \leq rnds$, $0 \leq k < 4$, $1 \leq \ell < 4$ then:
  - $S_{sbox,O}[n, 1 + 32i + 16 + 4k'] = 1$ for $k' = (k+1) \mod 4$;
  - $S_{sbox,I}[n, 1 + 32(i-1) + 4k' + \ell] = 1$ for $k' = (k+1) \mod 4$;
  - Increment $n = n + 1$. # To show $kword_{i,1,0} = \mathsf{sbox}(\mathsf{RotWord}(kword_{i-1,0,3}))$
- Set all other entries to 0

C.2 Cipher

The AES cipher is given as input the 128 bit rounds keys $rk_0, \ldots, rk_{rnds} \in (\mathbb{F}_2^8)^{16}$ and a 128 bit message $\vec{m}$. The cipher outputs a ciphertext ctx. Similar to the key expansion, blocks have 128 bits: $st \in \{0,1\}^{128}$ and we split the trace into bytes. Additionally $rnds \in \{10, 12, 14\}$. In our implementation these bytes are further reduced into two 4 bit segments. These segments always appear together thus we here ignore this detail.

Format of the cipher trace. The AES execution trace contains 8 intermediary states over all rounds *rnds*. This has some redundancy and unused states are set to 0.

$$\vec{tr} = (1, st_{0,0}, \ldots, st_{0,7}, \ldots, st_{rnds,0}, \ldots, st_{rnds,7})$$

Each intermediary state $st_{i,j}$ of $\vec{tr}$ contains 16 bytes arranged in a 4x4 grid of bytes

$$st_{i,j} = \begin{pmatrix} st_{i,j,0,0} & st_{i,j,0,1} & st_{i,j,0,2} & st_{i,j,0,3} \\ st_{i,j,1,0} & st_{i,j,1,1} & st_{i,j,1,2} & st_{i,j,1,3} \\ st_{i,j,2,0} & st_{i,j,2,1} & st_{i,j,2,2} & st_{i,j,2,3} \\ st_{i,j,3,0} & st_{i,j,3,1} & st_{i,j,3,2} & st_{i,j,3,3} \end{pmatrix}$$

where $st_{i,j,k,\ell} \in \{0,1\}^8$. We can interpret $st_{i,j,k,\ell} \in \mathbb{F}_2^8$ because $\{0,1\}^8$ and $\mathbb{F}_2^8$ are bijective.

We set that $st_{i,7} = rk_i$.

The cipher protocol. The cipher works as follows for *rnds* is the number of rounds:

- Initial round ($i = 0$),
  - AddRoundKey: add the first round key to the state, i.e. we set $st_{1,0} = st_{0,0} \oplus rk_0 = st_{0,0} \oplus st_{0,7}$
- Middle rounds ($i \in \{1, \ldots, rnds-1\}$)
  - SubBytes: applies the S-Box operation to each byte of the state, i.e. we set $st_{i,1} \coloneqq \mathsf{sbox}(st_{i,0})$
  - ShiftRows: permutes the state, by performing a cyclic rotation of the state $st_{i,1}$. Define by $\sigma : \mathbb{F}_4 \times \mathbb{F}_4 \mapsto \mathbb{F}_4 \times \mathbb{F}_4$ the permutation

$$\begin{pmatrix} \sigma(0,0) & \sigma(0,1) & \sigma(0,2) & \sigma(0,3) \\ \sigma(1,0) & \sigma(1,1) & \sigma(1,2) & \sigma(1,3) \\ \sigma(2,0) & \sigma(2,1) & \sigma(2,2) & \sigma(2,3) \\ \sigma(3,0) & \sigma(3,1) & \sigma(3,2) & \sigma(3,3) \end{pmatrix} = \begin{pmatrix} (0,0) & (0,1) & (0,2) & (0,3) \\ (1,1) & (1,2) & (1,3) & (1,0) \\ (2,2) & (2,3) & (2,0) & (2,1) \\ (3,3) & (3,0) & (3,1) & (3,2) \end{pmatrix}$$

    Then $\mathsf{ShiftRows}(st_{i,j})$ maps each $st_{i,j,k,\ell}$ to $st_{i,j,\sigma(k,\ell)}$. Rather than append a state to the transcript at this stage in the computation we instead an application of ShiftRows inside the next operations. This is because we are only permute the state and do not perform any operations.
  - MixColumns: performs a linear transformation on the state. Specifically MixColumns applies the matrix

$$\begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix}$$

    to each column of the state over the Galois field. We can represent this transform using rj2 and $\oplus$ operations (that are compatible with lookup tables) as follows.
    - $st_{i,2} = \mathsf{rj2}(\mathsf{ShiftRows}(st_{i,1}))$
    - for $k \in [0,3)$
      - $st_{i,3,k,0} = st_{i,1,\sigma(k,1)} \oplus st_{i,1,\sigma(k,2)} \oplus st_{i,1,\sigma(k,3)} \oplus st_{i,2,k,0} \oplus st_{i,2,k,1}$
      - $st_{i,3,k,1} = st_{i,1,\sigma(k,0)} \oplus st_{i,1,\sigma(k,2)} \oplus st_{i,1,\sigma(k,3)} \oplus st_{i,2,k,1} \oplus st_{i,2,k,2}$
      - $st_{i,3,k,2} = st_{i,1,\sigma(k,0)} \oplus st_{i,1,\sigma(k,1)} \oplus st_{i,1,\sigma(k,3)} \oplus st_{i,2,k,2} \oplus st_{i,2,k,3}$
      - $st_{i,3,k,3} = st_{i,1,\sigma(k,0)} \oplus st_{i,1,\sigma(k,1)} \oplus st_{i,1,\sigma(k,2)} \oplus st_{i,2,k,0} \oplus st_{i,2,k,3}$
  - AddRoundKey: adds the current round key to the state, i.e. $st_{i+1,0} \coloneqq st_{i,3} \oplus rk_i = st_{i,3} \oplus st_{i,7}$
- Final round ($i = rnds$);
  - SubBytes: applies the S-Box operation to each byte of the state, i.e. we set $st_{rnds,1} = \mathsf{sbox}(st_{rnds,0})$
  - $ctx = st_{rnds,2} = \mathsf{ShiftRows}(st_{rnds,1}) \oplus st_{rnds,7}$

We can directly many of these operations with our lookup tables. However, checking whether $st_{i,3,k,\ell}$ has been computed in MixColumns requires multiple steps. The prover thus sets for $k \in [0,3)$

$$\begin{aligned}
st_{i,4,k,0} &= st_{i,1,\sigma(k,0)} \oplus st_{i,1,\sigma(k,1)} & st_{i,5,k,0} &= st_{i,1,\sigma(k,3)} \oplus st_{i,4,k,0} & st_{i,6,k,0} &= st_{i,5,k,0} \oplus st_{i,2,k,2} \\
st_{i,4,k,1} &= st_{i,1,\sigma(k,2)} \oplus st_{i,1,\sigma(k,3)} & st_{i,5,k,1} &= st_{i,1,\sigma(k,2)} \oplus st_{i,4,k,0} & st_{i,6,k,1} &= st_{i,5,k,1} \oplus st_{i,2,k,3} \\
st_{i,4,k,2} &= st_{i,1,\sigma(k,1)} \oplus st_{i,4,k,0} & st_{i,5,k,2} &= st_{i,4,k,2} \oplus st_{i,2,k,0} & & \\
st_{i,4,k,3} &= st_{i,1,\sigma(k,0)} \oplus st_{i,4,k,1} & st_{i,5,k,3} &= st_{i,4,k,3} \oplus st_{i,2,k,1} & &
\end{aligned}$$

such that

$$\begin{aligned}
st_{i,3,k,0} &= st_{i,5,k,2} \oplus st_{i,2,k,1} & st_{i,3,k,1} &= st_{i,5,k,3} \oplus st_{i,2,k,2} \\
st_{i,3,k,2} &= st_{i,6,k,0} \oplus st_{i,2,k,3} & st_{i,3,k,3} &= st_{i,6,k,1} \oplus st_{i,2,k,3}
\end{aligned}$$

The $st_{i,j,k,\ell}$ are chosen such that the key words $st_{i,\ell}$ satisfy the above constraints. We are now ready to define our selection matrices for the cipher algorithm.

Selection matrices for the cipher. The concrete details of our selection matrices $S_{xor,O}$, $S_{xor,L}$, $S_{xor,R}$, $S_{sbox,I}$, $S_{sbox,O}$, $S_{rj2,O}$, $S_{rj2,I}$ are mechanical deferred to Appendix D. The general idea is that whenever, e.g.,

$$st_{i_O,j_O,k_O,\ell_O} = st_{i_L,j_L,k_L,\ell_L} \oplus st_{i_R,j_R,k_R,\ell_R}$$

then a new row is added to $S_{xor,O}$, $S_{xor,L}$, $S_{xor,R}$. In this new row, indexed n, all entries are 0 apart from

$$\begin{aligned}
S_{xor,O}[n, 1 + 128i_O + 16j_O + 4k_O + \ell_O] &= 1 \\
S_{xor,L}[n, 1 + 128i_L + 16j_L + 4k_L + \ell_L] &= 1 \\
S_{xor,R}[n, 1 + 128i_R + 16j_R + 4k_R + \ell_R] &= 1
\end{aligned}$$

D Selection Matrices for AES key expansion and cipher

To generate the $S_{xor,O}$, $S_{xor,L}$, and $S_{xor,R}$ matrices do the following.

- Initialise $n = 0$
- For $i = 0$, $k, \ell \in [0,4)$ then:
  - $S_{xor,O}[n, 1 + 128 + 4k + \ell] = 1$; $S_{xor,L}[n, 1 + 4k + \ell] = 1$; $S_{xor,R}[n, 1 + 16 \cdot 7 + 4k + \ell] = 1$
  - Increment $n = n + 1$. # To show $st_{1,0} = st_{0,0} \oplus st_{0,7}$
- For $i \in [1, rnds)$, $k, \ell \in [0,4)$ then:
  - $S_{xor,O}[n, 1 + 128(i+1) + 4k + \ell] = 1$; $S_{xor,L}[n, 1 + 128i + 16 \cdot 3 + 4k + \ell] = 1$; $S_{xor,R}[n, 1 + 128i + 16 \cdot 7 + 4k + \ell] = 1$
  - Increment $n = n + 1$. # To show $st_{i+1,0} := st_{i,3} \oplus rk_i = st_{i,3} \oplus st_{i,7}$
- For $i \in [1, rnds)$, $k \in [0,4)$ the following entries are set to 1. Let $\eta(k,\ell) : \mathbb{F}_4 \times \mathbb{F}_4 \mapsto \mathbb{F}_4 \times \mathbb{F}_4$ be the operation that applies $(k', \ell') = \sigma(k,\ell)$ and returns $4k' + \ell'$. Here σ is the permutation defined by ShiftRows. After each assignment

of $S_{xor,O}$, $S_{xor,L}$ and $S_{xor,R}$ we increment $n = n + 1$:

$S_{xor,O}[n, 1 + 128i + 16 \cdot 4 + 4k + 0]$	$S_{xor,L}[n, 1 + 128i + 16 \cdot 1 + \eta(k, 0)]$	$S_{xor,R}[n, 1 + 128i + 16 \cdot 1 + \eta(k, 1)]$
$S_{xor,O}[n, 1 + 128i + 16 \cdot 4 + 4k + 1]$	$S_{xor,L}[n, 1 + 128i + 16 \cdot 1 + \eta(k, 2)]$	$S_{xor,R}[n, 1 + 128i + 16 \cdot 1 + \eta(k, 3)]$
$S_{xor,O}[n, 1 + 128i + 16 \cdot 4 + 4k + 2]$	$S_{xor,L}[n, 1 + 128i + 16 \cdot 1 + \eta(k, 1)]$	$S_{xor,R}[n, 1 + 128i + 16 \cdot 4 + 4k + 0]$
$S_{xor,O}[n, 1 + 128i + 16 \cdot 4 + 4k + 3]$	$S_{xor,L}[n, 1 + 128i + 16 \cdot 1 + \eta(k, 0)]$	$S_{xor,R}[n, 1 + 128i + 16 \cdot 4 + 4k + 1]$
$S_{xor,O}[n, 1 + 128i + 16 \cdot 5 + 4k + 0]$	$S_{xor,L}[n, 1 + 128i + 16 \cdot 1 + \eta(k, 3)]$	$S_{xor,R}[n, 1 + 128i + 16 \cdot 4 + 4k + 0]$
$S_{xor,O}[n, 1 + 128i + 16 \cdot 5 + 4k + 1]$	$S_{xor,L}[n, 1 + 128i + 16 \cdot 1 + \eta(k, 2)]$	$S_{xor,R}[n, 1 + 128i + 16 \cdot 4 + 4k + 0]$
$S_{xor,O}[n, 1 + 128i + 16 \cdot 5 + 4k + 2]$	$S_{xor,L}[n, 1 + 128i + 16 \cdot 4 + 4k + 2]$	$S_{xor,R}[n, 1 + 128i + 16 \cdot 2 + 4k + 0]$
$S_{xor,O}[n, 1 + 128i + 16 \cdot 5 + 4k + 3]$	$S_{xor,L}[n, 1 + 128i + 16 \cdot 4 + 4k + 3]$	$S_{xor,R}[n, 1 + 128i + 16 \cdot 2 + 4k + 1]$
$S_{xor,O}[n, 1 + 128i + 16 \cdot 6 + 4k + 0]$	$S_{xor,L}[n, 1 + 128i + 16 \cdot 5 + 4k + 0]$	$S_{xor,R}[n, 1 + 128i + 16 \cdot 2 + 4k + 2]$
$S_{xor,O}[n, 1 + 128i + 16 \cdot 6 + 4k + 1]$	$S_{xor,L}[n, 1 + 128i + 16 \cdot 5 + 4k + 1]$	$S_{xor,R}[n, 1 + 128i + 16 \cdot 2 + 4k + 3]$
$S_{xor,O}[n, 1 + 128i + 16 \cdot 3 + 4k + 0]$	$S_{xor,L}[n, 1 + 128i + 16 \cdot 5 + 4k + 2]$	$S_{xor,R}[n, 1 + 128i + 16 \cdot 2 + 4k + 1]$
$S_{xor,O}[n, 1 + 128i + 16 \cdot 3 + 4k + 1]$	$S_{xor,L}[n, 1 + 128i + 16 \cdot 5 + 4k + 3]$	$S_{xor,R}[n, 1 + 128i + 16 \cdot 2 + 4k + 2]$
$S_{xor,O}[n, 1 + 128i + 16 \cdot 3 + 4k + 2]$	$S_{xor,L}[n, 1 + 128i + 16 \cdot 6 + 4k + 0]$	$S_{xor,R}[n, 1 + 128i + 16 \cdot 2 + 4k + 3]$
$S_{xor,O}[n, 1 + 128i + 16 \cdot 3 + 4k + 3]$	$S_{xor,L}[n, 1 + 128i + 16 \cdot 6 + 4k + 1]$	$S_{xor,R}[n, 1 + 128i + 16 \cdot 2 + 4k + 3]$

To show the Mix Columns $\oplus$ operations

- For $i = rnds$, $k, \ell \in [0, 4)$ then:
  - $S_{xor,O}[n, 1 + 128rnds + 16 \cdot 2 + 4k + \ell] = 1$; $S_{xor,L}[n, 1 + 128rnds + 16 \cdot 1 + 4k + \ell'] = 1$; $S_{xor,R}[n, 1 + 128rnds + 16 \cdot 7 + 4k + \ell] = 1$ where $\ell' = (\ell + k) \mod 4$.
  - Increment $n = n + 1$. # To show $st_{rnds,2} = \mathsf{ShiftRows}(st_{rnds,1}) \oplus st_{rnds,7}$
- Set all other entries to 0

To generate the $S_{sbox,O}$, $S_{sbox,I}$ matrices do the following.

- Initialise $n = 0$
- For $i \in [1, rnds]$, $k, \ell \in [0, 4)$ then:
  - $S_{sbox,O}[n, 1 + 128i + 16 \cdot 1 + 4k + \ell] = 1$; $S_{sbox,I}[n, 1 + 128i + 16 \cdot 0 + 4k + \ell] = 1$
  - Increment $n = n + 1$. # To show $st_{i,1} = \mathsf{sbox}(st_{i,0})$
- Set all other entries to 0

To generate the $S_{rj2,O}$, $S_{rj2,I}$ matrices do the following.

- Initialise $n = 0$
- For $i \in [1, rnds]$, $k, \ell \in [0, 4)$ then:
  - $S_{rj2,O}[n, 1 + 128i + 16 \cdot 2 + 4k + \ell] = 1$; $S_{rj2,I}[n, 1 + 128i + 16 \cdot 1 + 4k + \ell'] = 1$ where $\ell' = (\ell + k) \mod 4$
  - Increment $n = n + 1$. # To show $st_{i,1} = \mathsf{sbox}(st_{i,0})$
- Set all other entries to 0