MANY-OUT-OF-MANY PROOFS

with applications to *Anonymous Zether*

Benjamin E. Diamond
J.P. Morgan

Abstract

We introduce a family of extensions to the *one-out-of-many proofs* of Groth and Kohlweiss [GK15], which efficiently prove statements about *many* messages among a list of commitments. We allow a prover to demonstrate knowledge of a *secret* orbit under a fixed permutation, as well as openings to zero of the images under *arbitrary* linear functionals of the commitments represented by this orbit. This powerful technique strictly generalizes [GK15]. Our communication remains logarithmic; our computation increases only by a logarithmic multiplicative factor. Our work introduces a new "bitwise rotation" technique, and a novel instantiation of the number-theoretic transform. We also introduce a new ring signature scheme, which is more flexible in some circumstances than that of [GK15].

Applying these techniques, we construct a protocol for the *Anonymous Zether* payment system, resolving a question raised by Bünz, Agrawal, Zamani, and Boneh [BAZB, §D]. We further correct shortcomings in the "Σ-Bullets" of [BAZB]. Finally, we describe a concrete implementation of our protocol. Anonymous Zether represents a leading option for private payment in enterprise settings.

1 Introduction

One-out-of-many proofs, introduced by Groth and Kohlweiss [GK15], allow a prover to demonstrate knowledge of a secret *element* among a public list of commitments, together with an opening of this commitment to 0. This important primitive has been used to construct ring signatures, zerocoin, and proofs of set membership [GK15], along with "accountable ring signatures" [BCC$^+$15]; it has also been re-instantiated in the setting of lattices [ESS$^+$19].

By definition, these proofs bear upon only one (secret) element of a list, and establish nothing about the others; indeed, in general the prover knows nothing about these other elements. Certain applications, however, require more flexible assertions (which, in particular, pertain to more than one element of the list). For example, given some list $c_0, \dots, c_{N-1}$ of commitments, and having agreed upon some pre-specified linear *map* $M \colon \mathbb{F}_q^N \to \mathbb{F}_q^s$, a prover might wish to prove knowledge of a *secret* permutation $K \in \mathbf{S}_N$, as well as of openings to zero of the image points of $c_{K(0)}, \dots, c_{K(N-1)}$ under M. We show how this can be done, provided that the prover and verifier agree in advance to restrict K to one among a certain class of order-N subsets (often subgroups) of $\mathbf{S}_N$.

This technique is powerful, with an interesting combinatorial flavor. In fact, we situate the above-described protocol within a natural family of extensions to [GK15], themselves parameterized by permutations $\kappa \in \mathbf{S}_N$ of a certain form (namely, those whose action partitions $\{0, 1, \dots, N-1\}$ into equal-sized orbits). In this family, $\kappa = \mathrm{id} \in \mathbf{S}_N$ exactly recovers [GK15], whereas the above example corresponds to κ an N-cycle. Finally, $\kappa = (0, 2, \dots, N-2)(1, 3, \dots, N-1)$ (and N even) serves as the crucial step in Anonymous Zether. In each case, the prover proves knowledge of exactly one "ordered orbit" of κ, as well as that the commitments represented by this orbit satisfy prescribed linear equations.

Remarkably, our communication *remains* logarithmic (like that of [GK15]). Moreover, under mild conditions on the linear map M (which hold in all of our applications), we add at most a logarithmic multiplicative factor to both the prover's and verifier's computational complexity. These thus remain "quasilinear".

1.1 Zether and Anonymous Zether

Our primary application is a cryptographic protocol for *Anonymous Zether*, originally proposed in Bünz, Agrawal, Zamani and Boneh [BAZB]. *Zether* is a system for confidential payment, which features many novel properties. In contrast to Monero and Zcash, Zether demands only constant long-term space overhead per participant (though see also Quisquis [FMMO]); Zether also lacks a trusted setup, and is "account-based".

The article [BAZB] focuses on "basic Zether", in which account balances and transfer amounts are concealed, but participants' identities are not. In contrast, the appendix [BAZB, §D] outlines an approach to *anonymous* payment; this appendix suggests a relation, but does not provide a proof protocol.

While the authors of [BAZB] suggest using one-out-of-many proofs, it seems evident that no elementary adaptation of [GK15] (or of the extension [BCC$^+$15]) can alone suffice. Indeed, the Anonymous Zether relation entails facts not just about *two* among a list of ciphertexts (namely, the sender's and receiver's, which are required to encrypt opposite amounts) but also about all of *the rest* (which are required to encrypt zero). Absent new ideas, this statement—which, again, concerns *all* N among a list of ciphertexts—appears to entail, at the very least, higher asymptotic complexity (in both communication and computation).

Using our "many-out-of-many" proofs, as well as a number of additional innovations—all described in this paper—we offer a resolution to this problem, providing a protocol for Anonymous Zether which features only $O(\log N)$ communication and $O(N \log N)$ computation for both the prover and the verifier (on an "anonymity set" of size N). Our approach proves a basic Zether-like relation over *two* (secret) ciphertexts, while proving that the rest are encryptions of zero.

We also identify and address shortcomings in basic Zether. Its central cryptographic technique, called "Σ-Bullets", seeks to adapt Bünz, Bootle, Boneh, Poelstra, Wuille and Maxwell's *Bulletproofs* [BBB$^+$18] (originally designed for Pedersen commitments) to the setting of El Gamal ciphertexts. We argue below that the Σ-Bullets protocol of [BAZB] is insecure, and describe concrete attacks on it (targeting both soundness and zero-knowledge). We also propose a secure amendment.

We finally describe an Ethereum-based implementation of our protocol, with the aforementioned asymptotics and favorable constants. Anonymous Zether's proving time is competitive with the state-of-the-art protocols; its verification time is also competitive. This makes it particularly compelling in enterprise and consortium environments, in which the size of the *entire network* may be small enough to fit into a single anonymity set. Anonymous Zether is also feasible for use on the Ethereum mainnet, as of the Istanbul hard fork (with caveats concerning "gas linkability", discussed for example in [BAZB, §F]).

2 Overview of Techniques

The central technique of one-out-of-many proofs is the construction, by the prover, of certain polynomials $P_i(X)$, $i \in \{0, \ldots, N-1\}$, and the efficient transmission (i.e., using only $O(\log N)$ communication) to the verifier of these polynomials' *evaluations* $p_i := P_i(x)$ at a challenge x. Importantly, each $P_i(X)$ has "high degree" (i.e., m, where $m = \log N$) if and only if $i = l$, where l is a secret index chosen by the prover.

Our main idea is that, having reconstructed from the prover the vector $(p_i)_{i=0}^{N-1}$ of evaluations, the verifier may "homomorphically permute" this vector and re-use it in successive multi-exponentiations. In this way, the verifier will "pick out" secret elements among $c_0, \ldots, c_{N-1}$ in a highly controlled way (and without necessitating further communication).

In what follows, we fix a permutation $\kappa \in \mathbf{S}_N$. Given the vector $(p_i)_{i=0}^{N-1}$—and, again, *not* knowing that index l for which $P_i(X)$ has high degree—the verifier may, by iteratively permuting this vector, nonetheless construct the sequence of vectors

$$\left(p_{\kappa^{-j}(i)}\right)_{i=0}^{N-1},$$

for $j \in \{0, \ldots, o-1\}$, where each $\kappa^{-j} \in \mathbf{S}_N$ is an "inverse iterate" of κ and o denotes κ's order in $\mathbf{S}_N$.

Despite not knowing l, the verifier nonetheless knows that $P_{\kappa^{-j}(i)}(X)$ has high degree if and only if $i = \kappa^j(l)$. In this way, the verifier implicitly iterates through the orbit (under κ) of an *unknown* element $l \in \{0, \ldots, N-1\}$. Under the additional condition that $\langle \kappa \rangle \subset \mathbf{S}_N$ acts freely on $\{0, \ldots, N-1\}$, each implicit map $\{0, \ldots, o-1\} \to \{0, \ldots, N-1\}$ (sending $j \mapsto \kappa^j(l)$) is necessarily injective (i.e., regardless of

l), and these orbits never "double up". Permutations κ of this type thus represent a natural class for our purposes.

2.1 Correction terms and linear maps

An issue arises from the fact that $\prod_{i=0}^{N-1} c_i^{p_i}$ does not *directly* yield $c_l^{x^m}$ (where $m = \log N$), but rather the sum of this element with lower-order terms which must be "cancelled out". More generally, an analogous issue holds for each $e_j := \prod_{i=0}^{N-1} c_i^{p_{\kappa^{-j}(i)}}$ (for $j \in \{0, \ldots, o-1\}$). Furthermore, there may be up to linearly many such terms (if $o = N$, say), and to send correction terms for each would impose excessive communication costs.

Our compromise is to correct not each individual term e_j, but rather a "Vandermonde-style" combination of these terms. In fact, for additional flexibility, we interpose an arbitrary linear transformation $M \colon \mathbb{F}_q^o \to \mathbb{F}_q^s$. We then send correction terms *only* for the single element

$$\begin{bmatrix} 1 & y & \ldots & y^{s-1} \end{bmatrix} \cdot \begin{bmatrix} M \end{bmatrix} \cdot \begin{bmatrix} e_0 \\ e_1 \\ \vdots \\ e_{o-1} \end{bmatrix},$$

where y is a challenge (the left dot is a matrix product, whereas the right dot is a "module product"). By interleaving y with the many-out-of-many process with appropriate delicacy, we can ensure that the resulting protocol is still sound.

2.2 A canonical example

To illustrate these ideas, we describe an example which is essentially canonical: the case $\kappa = (0, 1, \ldots, N-1)$ (we describe reductions from general κ to this case below). Iterating this permutation corresponds exactly to "circularly rotating" the vector $(p_i)_{i=0}^{N-1}$; this process in turn "homomorphically increments" l modulo N. In this way, the prover implicitly sends the top row of an unknown permutation matrix to the verifier, who constructs the rest locally.

$$\begin{bmatrix} \underbrace{0, \ldots\ldots, 1, \ldots\ldots\ldots\ldots, 0}_{\text{1 only at index } l} \\ 0, \ldots\ldots\ldots\ldots, 1, \ldots\ldots\ldots\ldots, 0 \\ 0, \ldots\ldots\ldots\ldots\ldots, 1, \ldots\ldots\ldots, 0 \\ \vdots \\ 0, \ldots, 1, \ldots\ldots\ldots\ldots\ldots\ldots, 0 \\ 0, \ldots\ldots, 1, \ldots\ldots\ldots\ldots\ldots, 0 \end{bmatrix}$$

Figure 1: "Prover's view".

$$\begin{bmatrix} \text{———} & (p_i)_{i=0}^{N-1} & \text{———} \\ \text{——} & \text{———} & (p_i)_{i=0}^{N-1} & \text{———} \\ \text{———} & \text{———} & (p_i)_{i=0}^{N-1} & \text{——} \\ & & \vdots & \\ \text{——} & (p_i)_{i=0}^{N-1} & \text{———} & \text{———} \\ \text{———} & (p_i)_{i=0}^{N-1} & \text{———} & \text{——} \end{bmatrix}$$

Figure 2: "Verifier's view".

The evaluation of the matrix multiplication of Fig. 2 by the vector of curve points $(c_j)_{j=0}^{N-1}$ takes $O(N^2)$ time, naïvely. Yet Fig. 2 is exactly a "circulant" matrix, and this multiplication is a convolution; the number-theoretic transform can thus be applied (we discuss this further below).

The resulting matrix product $(e_j)_{j=0}^{N-1}$ yields, modulo lower-order terms, the *permuted* input vector $(c_{\kappa^l(j)})_{j=0}^{N-1}$, upon which any linear transformation M (as well as the "Vandermonde trick") can be homomorphically applied. (We use the identity $\kappa^j(l) = \kappa^l(j)$, true in particular for $\kappa = (0, 1, \ldots, N-1)$.) Suppose now, in addition, that the prover and verifier have agreed in advance upon a linear functional $M \colon \mathbb{F}_q^N \to \mathbb{F}_q$. Our general protocol, in this particular case, thus yields a proof of knowledge of a secret permutation $K \in \langle (0, 1, \ldots, N-1) \rangle$, as well as an opening to zero of the image under M of the *permuted*

vector $c_{K(0)}, \ldots, c_{K(N-1)}$. (Crucially, the permutation K remains secret.) Heuristically, we prove that the "messages" of $c_0, \ldots, c_{N-1}$, once appropriately rotated, reside in a specified hyperplane of $\mathbb{F}_q^N$.

Our communication complexity is *still* logarithmic; the computational complexity becomes $O(N \log^2 N)$ for the prover and $O(N \log N)$ for the verifier.

2.3 Circular convolutions and the number-theoretic transform

We remark briefly on our use of Fourier-theoretic techniques. That the fast Fourier transform can be applied in finite fields (which possess adequately many roots of unity) was first noted apparently by Pollard [Pol71], and is surveyed further in Nussbaumer [Nus82, §8]. In both settings, however, only the convolution of two *field-element* vectors is discussed.

We view the elliptic curve $E(\mathbb{F}_p)$ as a module (in fact a vector space) over $\mathbb{F}_q$ in what follows. Our setting, unusually, mandates that a vector of *module* elements (i.e., curve points) be convolved with a vector of field elements. Our observation in this capacity is that only the *module* structure, and not the ring structure, of a signal's domain figures in its role throughout the Fourier transform, and the techniques indeed carry through (i.e., even when "ring multiplication" can't be performed on the signal). Despite our having made a cursory search, this observation appears absent from the literature; in any case, it complements well-known techniques.

2.4 Additional innovations

We introduce various additional innovations throughout our construction of Anonymous Zether. These include an adaptation of many-out-of-many proofs to El Gamal ciphertexts under *heterogeneous* keys, and a technique to ensure that l is chosen *consistently* across multiple executions of the many-out-of-many procedure (both are applicable equally in the classical case).

We also introduce a new ring signature, which replaces the final "randomness revelation" step of [GK15] with a Schnorr knowledge-of-exponent protocol. The advantage of this technique is that the final Schnorr proof can be shared across concurrent executions of the protocol over *multiple* rings, ensuring in particular that the same secret key is used in each execution.

Finally, we introduce an "opposite parity proof", used to assert that two separate executions of the many-out-of-many protocol use secrets l featuring opposite parities (for N even). This technique finds important use in Anonymous Zether.

3 Security Definitions

We recall various security definitions.

3.1 Groups

Following Katz and Lindell [KL15, §8.3.2], we let $\mathcal{G}$ denote a *group-generation algorithm*, which on input 1^λ outputs a cyclic group $\mathbb{G}$, its prime order q (with bit-length λ) and a generator $g \in \mathbb{G}$. Moreover, we have:

Definition (Katz–Lindell [KL15, Def. 8.62])**.** The *discrete-logarithm experiment* $\mathsf{DLog}_{\mathcal{A},\mathcal{G}}(\lambda)$ is defined as:

1. Run $\mathcal{G}(1^\lambda)$ to obtain $(\mathbb{G}, q, g)$.
2. Choose a uniform $h \in \mathbb{G}$.
3. $\mathcal{A}$ is given $\mathbb{G}, q, g, h$, and outputs $x \in \mathbb{Z}_q$.
4. The output of the experiment is defined to be 1 if $g^x = h$, and 0 otherwise.

We say that the *discrete-logarithm problem is hard relative to* $\mathcal{G}$ if for each probabilistic polynomial-time algorithm $\mathcal{A}$ there exists a negligible function negl such that $\Pr[\mathsf{DLog}_{\mathcal{A},\mathcal{G}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

We also have the decisional Diffie–Hellman assumption, which we adapt from [KL15, Def. 8.63]:

Definition. The *DDH experiment* $\mathsf{DDH}_{\mathcal{A},\mathcal{G}}(\lambda)$ is defined as:

1. Run $\mathcal{G}(1^\lambda)$ to obtain $(\mathbb{G}, q, g)$.
2. Choose uniform $x, y, z \in \mathbb{Z}_p$ and a uniform bit $b \in \{0,1\}$.
3. Give $(\mathbb{G}, q, g, g^x, g^y)$ to $\mathcal{A}$, as well as g^z if $b = 0$ and g^{xy} if $b = 1$. $\mathcal{A}$ outputs a bit b'.
4. The output of the experiment is defined to be 1 if and only if $b' = b$.

We say that *the DDH problem is hard relative to* $\mathcal{G}$ if for each probabilistic polynomial-time algorithm $\mathcal{A}$ there exists a negligible function negl such that $\Pr[\mathsf{DDH}_{\mathcal{A},\mathcal{G}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

3.2 Commitment schemes

A *commitment scheme* is a pair of probabilistic algorithms $(\mathsf{Gen}, \mathsf{Com})$; given public parameters $\mathsf{param} \leftarrow \mathrm{Gen}(1^\lambda)$ and a message $m \in \{0,1\}^\lambda$, we have a commitment $\mathsf{com} := \mathsf{Com}(\mathsf{params}, m; r)$, as well as a decommitment procedure (effected by sending m and r). We now define:

Definition (Katz–Lindell [KL15, Def. 5.13])**.** The *commitment binding experiment* $\mathsf{Binding}_{\mathcal{A},\mathsf{Com}}(\lambda)$ is defined as:

1. Parameters $\mathsf{params} \leftarrow \mathsf{Gen}(1^\lambda)$ are generated.
2. $\mathcal{A}$ is given params and outputs (m_0, r_0, m_1, r_1).
3. The output of the experiment is defined to be 1 if and only if $m_0 \neq m_1$ and $\mathsf{Com}(\mathsf{params}, m_0; r_0) = \mathsf{Com}(\mathsf{params}, m_1; r_1)$.

A commitment scheme Com is *computationally binding* if for each PPT adversary there is a negligible function negl such that $\Pr\left[\mathsf{Binding}_{\mathcal{A},\mathsf{Com}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$. If $\mathsf{negl} = 0$, w say that Com is *perfectly binding*.

Definition (Katz–Lindell [KL15, Def. 5.13])**.** The *commitment hiding experiment* $\mathsf{Hiding}_{\mathcal{A},\mathsf{com}}(\lambda)$ is defined as:

1. Paramers $\mathsf{params} \leftarrow \mathsf{Gen}(1^\lambda)$ are generated.
2. The adversary $\mathcal{A}$ is given input params, and outputs a pair of messages $m_0, m_1 \in \{0,1\}^\lambda$.
3. A uniform bit $b \in \{0,1\}$ is chosen and $\mathsf{com} \leftarrow \mathsf{Com}(\mathsf{params}, m_b; r)$ is computed.
4. The adversary $\mathcal{A}$ is given com and outputs a bit b'.
5. The output of the experiment is 1 if and only if $b' = b$.

A commitment scheme Com is *computationally hiding* if for each PPT adversary there exists a negligible function negl such that $\Pr\left[\mathsf{Hiding}_{\mathcal{A},\mathsf{Com}}(\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$. If $\mathsf{negl} = 0$, we say that Com is *perfectly hiding*.

For example, we have Pedersen commitments (to both scalars and vectors), as described in [BCC+16, §2.2]. Assuming a group generation algorithm $\mathcal{G}$, we define the *Pedersen commitment* scheme by setting $\mathsf{Gen}(1^\lambda) = \mathsf{params} := (g, h)$, where $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^\lambda)$ and $h \leftarrow \mathbb{G}$ is random, and defining $\mathsf{Com}(\mathsf{params}, m; r) := g^m h^r$. The Pedersen commitment scheme is perfectly hiding; if the discrete-logarithm problem is hard relative to $\mathcal{G}$, it's also computationally binding. Pedersen vector commitments are defined similarly, and are secure under a generalization of the discrete-logarithm assumption (see e.g. [BBB+18, Def. 6]).

A commitment scheme is *homomorphic* if, for each params, its message, randomness, and commitment spaces are abelian groups, and the commitment function is a group homomorphism. For notational convenience, we often omit params. For notational convenience, we often omit params.

3.3 Zero-knowledge arguments of knowledge

We define zero-knowledge arguments of knowledge, closely following [GK15] and [BCC+16]. Where possible, we formulate security definitions in the "experiment-based" style of Katz and Lindell.

We posit a triple of interactive, probabilistic polynomial time algorithms $\Pi = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$. Given some polynomial-time-decidable ternary relation $\mathcal{R} \subset (\{0,1\}^*)^3$, each common reference string $\sigma \leftarrow \mathsf{Setup}(1^\lambda)$ yields an NP language $L_\sigma = \{x \mid (\sigma, x, w) \in \mathcal{R}\}$. We denote by $\mathrm{tr} \leftarrow \langle \mathcal{P}(s), \mathcal{V}(t) \rangle$ the (random) transcript of an interaction between $\mathcal{P}$ and $\mathcal{V}$ on auxiliary inputs s and t (respectively), and write the verifier's output b as $\langle \mathcal{P}(s), \mathcal{V}(t) \rangle = b$.

We now have:

Definition. The *completeness experiment* $\mathsf{Complete}_{\mathcal{A},\Pi,\mathcal{R}}(\lambda)$ is defined as:

1. A common reference string $\sigma \leftarrow \mathsf{Setup}(1^\lambda)$ is generated.
2. $\mathcal{A}$ is given σ and outputs (u, w).
3. An interaction $\langle \mathcal{P}(\sigma, u, w), \mathcal{V}(\sigma, u) \rangle = b$ is carried out.
4. The output of the experiment is defined to be 1 if and only if $(\sigma, u, w) \in \mathcal{R} \rightarrow b = 1$.

We say that $\Pi = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ is *perfectly complete* if for each PPT adversary $\mathcal{A}$, $\Pr[\mathsf{Complete}_{\mathcal{A},\Pi,\mathcal{R}}(\lambda)] = 1$.

Supposing that $\Pi = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ is a $2\mu + 1$-move, public-coin interactive protocol, we have:

Definition. The $(n_1, \ldots, n_\mu)$*-special soundness experiment* $\mathsf{Sound}^{(n_1,\ldots,n_\mu)}_{\mathcal{A},\mathcal{X},\Pi,\mathcal{R}}(\lambda)$ is defined as:

1. A common reference string $\sigma \leftarrow \mathsf{Setup}(1^\lambda)$ is generated.
2. $\mathcal{A}$ is given σ and outputs u, as well as an $(n_1, \ldots, n_\mu)$-tree (say tree) of accepting transcripts.
3. $\mathcal{X}$ is given σ, u, and tree and outputs w.
4. The output of the experiment is designed to be 1 if and only if $(\sigma, u, w) \in \mathcal{R}$.

We say that $\Pi = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ is *computationally* $(n_1, \ldots, n_\mu)$*-special sound* if there exists a PPT extractor $\mathcal{X}$ for which, for each PPT adversary $\mathcal{A}$, there exists a negligible function negl for which $\Pr[\mathsf{Sound}^{(n_1,\ldots,n_\mu)}_{\mathcal{A},\mathcal{X},\Pi,\mathcal{R}}(\lambda) = 1] \geq 1 - \mathsf{negl}(\lambda)$. If $\mathsf{negl} = 0$, we say that Π is *perfectly* $(n_1, \ldots, n_\mu)$*-special sound*.

We turn to zero knowledge.

Definition. The *special honest verifier zero knowledge experiment* $\mathsf{SHVZK}_{\mathcal{A},\mathcal{S},\Pi,\mathcal{R}}(\lambda)$ is defined as:

1. A common reference string $\sigma \leftarrow \mathsf{Setup}(1^\lambda)$ is generated.
2. $\mathcal{A}$ is given σ and outputs (u, w, ρ).
3. A uniform bit $b \in \{0, 1\}$ is chosen.
   - If $b = 0$, $\mathrm{tr} \leftarrow \langle \mathcal{P}^*(\sigma, u, w), \mathcal{V}(\sigma, u; \rho) \rangle$ is assigned.
   - If $b = 1$, $\mathrm{tr} \leftarrow \mathcal{S}(\sigma, u, \rho)$ is assigned.
4. The adversary $\mathcal{A}$ is given tr and outputs a bit b'.
5. The output of the experiment is defined to be 1 if and only if $b' = b$ and $(\sigma, u, w) \in \mathcal{R}$.

We say that $\Pi = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ is *computationally special honest verifier zero knowledge* if there exists a PPT simulator $\mathcal{S}$ for which, for each PPT adversary A, there exists a negligible function negl for which $\Pr[\mathsf{SHVZK}_{\mathcal{A},\mathcal{S},\Pi,\mathcal{R}}(\lambda) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$. If $\mathsf{negl} = 0$, we say that Π is *perfect special honest verifier zero knowledge*.

3.4 Ring signatures

We define ring signature schemes, closely following the article of Bender, Katz, and Morselli [BKM09]. We begin with algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$. $\mathsf{Gen}(1^\lambda)$ outputs a keypair (pk, sk), whereas $\sigma \leftarrow \mathsf{Sign}_{s,sk}(m, R)$ signs the message m on behalf of the ring $R = (pk_0, \ldots, pk_{N-1})$, where (pk_s, sk) is a valid keypair; finally, $\mathsf{Vrfy}_R(m, \sigma)$ verifies the purported signature σ of m on behalf of R. We fix a polynomial $N(\cdot)$ in what follows.

Definition (Bender–Katz–Morselli [BKM09, Def. 3])**.** The *unforgeability with respect to insider corruption experiment* $\mathsf{UnforgeIC}^{N(\cdot)}_{\mathcal{A},\Pi}(\lambda)$ is defined as:

1. Keypairs $(pk_i, sk_i)_{i=0}^{N(\lambda)-1}$ are generated using $\mathsf{Gen}(1^\lambda)$, and the list of public keys $S := (pk_i)_{i=0}^{N(\lambda)-1}$ is given to $\mathcal{A}$.
2. $\mathcal{A}$ is given access to a *signing oracle* $\mathsf{Osign}(\cdot,\cdot,\cdot)$ such that $\mathsf{Osign}(s, m, R)$ returns $\mathsf{Sign}_{sk_s}(m, R)$, where we require $pk_s \in R$.
3. $\mathcal{A}$ is also given access to a *corrupt oracle* $\mathsf{Corrupt}(\cdot)$, where $\mathsf{Corrupt}(i)$ outputs sk_i.
4. $\mathcal{A}$ outputs (R^*, m^*, σ^*), and succeeds if $\mathsf{Vrfy}_{R^*}(m^*, \sigma^*) = 1$, $\mathcal{A}$ never queried $(\star, m^*, R^*)$, and $R^* \subset S \backslash C$, where C is the set of corrupted users.

We say that $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ is *unforgeable with respect to insider corruption* if, for each PPT adversary and polynomial $N(\cdot)$, there exists a negligible function negl for which $\Pr[\mathsf{UnforgeIC}^{N(\cdot)}_{\mathcal{A},\Pi}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Definition (Bender–Katz–Morselli [BKM09, Def. 3])**.** The *anonymity with respect to adversarially chosen keys experiment* $\mathsf{AnonACK}^{N(\cdot)}_{\mathcal{A},\Pi}(\lambda)$ is defined as:

1. Keypairs $(pk_i, sk_i)_{i=0}^{N(\lambda)-1}$ are generated using $\mathsf{Gen}(1^\lambda)$, and the list of public keys $S := (pk_i)_{i=0}^{N(\lambda)-1}$ is given to $\mathcal{A}$.
2. $\mathcal{A}$ is given access to a *signing oracle* $\mathsf{Osign}(\cdot,\cdot,\cdot)$ such that $\mathsf{Osign}(s, m, R)$ returns $\mathsf{Sign}_{sk_s}(m, R)$, where we require $pk_s \in R$.
3. $\mathcal{A}$ outputs a message m, distinct indices i_0 and i_1, and a ring R for which $pk_{i_0}, pk_{i_1} \in R$.
4. A random bit b is chosen, and $\mathcal{A}$ is given the signature $\sigma \leftarrow \mathsf{Sign}_{sk_{i_b}}(m, R)$. The adversary outputs a bit b'.
5. The output of the experiment is defined to be 1 if and only if $b' = b$.

We say that $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ achieves *anonymity with respect to adversarially chosen keys* if for each PPT adversary $\mathcal{A}$ and polynomial $N(\cdot)$, there exists a negligible function negl for which $\Pr[\mathsf{AnonACK}^{N(\cdot)}_{\mathcal{A},\Pi}(\lambda) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$.

We note that this definition is *not* the strongest formulation of anonymity given in [BKM09], and in particular does not ensure security in the face of attribution attacks or full key exposure [BKM09, Def. 4]. We will argue below that this slightly weaker definition suffices for our purposes.

4 Results and Protocols

We turn to our main results. We begin with preliminaries on permutations, referring to Cohn [Coh74] for further background.

Definition. We say that a permutation $\kappa \in \mathbf{S}_N$ is *free* if it satisfies any, and hence all, of the following equivalent conditions:

- The natural action of $\langle\kappa\rangle \subset \mathbf{S}_N$ on $\{0,\ldots,N-1\}$ is free.
- The natural action of $\langle\kappa\rangle$ partitions the set $\{0,\ldots,N-1\}$ into orbits of equal size.
- κ is a product of equal-length cycles, with no fixed points.
- For each $l \in \{0,\ldots,N-1\}$, the stabilizer $\langle\kappa\rangle_l \subset \langle\kappa\rangle$ is trivial.
- For each $l \in \{0,\ldots,N-1\}$, the natural map $\{0,\ldots,o-1\} \to \{0,\ldots,N-1\}$ sending $j \mapsto \kappa^j(l)$ is injective (we write o for the order of κ).

We leave the equivalence of these definitions to the reader.

For any free $\kappa \in \mathbf{S}_N$, we also have a notion of an "ordered orbit": namely, the ordered sequence $\kappa^j(l) \in \{0,\ldots,N-1\}$, for $j \in \{0,\ldots,o-1\}$. By hypothesis on κ, this sequence contains no repetitions.

We now present our protocols. In all protocols, $\mathsf{Setup}(1^\lambda)$ runs the group-generation procedure $\mathcal{G}(1^\lambda)$ and the commitment scheme setup $\mathsf{Gen}(1^\lambda)$, and then stores $\sigma \leftarrow \mathsf{Setup}(1^\lambda) = (\mathbb{G}, q, g, \mathsf{params})$.

4.1 Commitments to bits

We replicate in its entirety, for convenience, the "bit commitment" protocol of Bootle, Cerulli, Chaidos, Ghadafi, Groth, and Petit [BCC+15, Fig. 4], which we further specialize to the binary case (i.e., $n = 2$). This protocol improves the single-bit commitment procedure of [GK15, Fig. 1], and requires slightly less communication.

Following [BCC+15], we have the relation:

$$\mathcal{R}_1 = \{(B; (b_0,\ldots,b_{m-1}), r_B) : \forall i, b_i \in \{0,1\} \wedge B = \mathsf{Com}\,(b_0,\ldots,b_{m-1}; r_B)\}\,,$$

and the protocol:

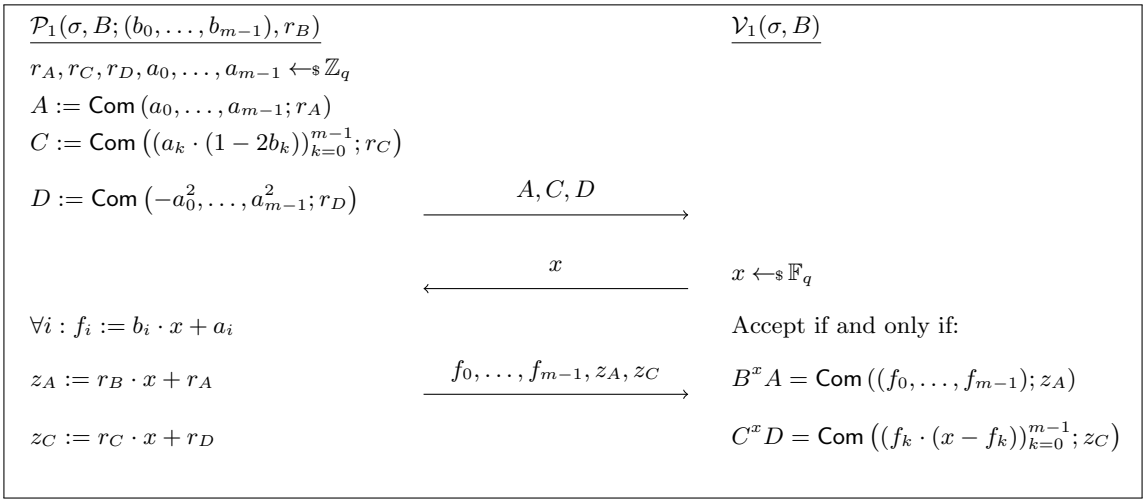

Figure 3: Protocol for the relation $\mathcal{R}_1$.

Finally, we have:

Lemma (Bootle, et al. [BCC+15])**.** *The protocol of Fig. 3 is perfectly complete. If* Com *is (perfectly) binding, then it is (perfectly) (3)-special sound. If* Com *is (perfectly) hiding, then it is (perfectly) special honest verifier zero knowledge*

Proof. We refer to [BCC+15, Lem. 1]. We note that [BCC+15, Fig. 4]'s perfect SHVZK relies on its use of (perfectly hiding) Pedersen commitments; in our slightly more general setting, $\mathcal{S}$ must simulate $C \leftarrow \mathsf{Com}(0,\ldots,0)$ as a random commitment to zero. As in [BCC+15, Lem. 1], we observe that the

remaining elements of the simulated transcript are either identically distributed to those of real ones or are uniquely determined given C. The indistinguishability of the simulation therefore reduces directly to the hiding property of the commitment scheme. □

4.2 Main protocol

Our main result is a proof of knowledge of an index l, as well as of openings $r_0, \ldots, r_{s-1}$ to 0 of the image points (under a fixed linear map $M\colon \mathbb{F}_q^o \to \mathbb{F}_q^s$) of the commitments $c_{\kappa^j(l)}$ represented by l's *ordered orbit*. We represent M as an $s \times o$ matrix over $\mathbb{F}_q$ in what follows. For commitments $c_0, \ldots, c_{N-1}$, we thus have the relation:

$$\mathcal{R}_2 = \left\{ (\sigma, (c_0, \ldots, c_{N-1}), \kappa, M; l, (r_0, \ldots, r_{s-1})) : \left[\ \mathsf{Com}(0; r_i)\ \right]_{i=0}^{s-1} = \left[\ M\ \right] \cdot \left[\ c_{\kappa^j(l)}\ \right]_{j=0}^{o-1} \right\}.$$

We understand both arrays as column vectors, and the "dot" as a module product of a column vector (of curve points) by the field matrix M.

As in [GK15], we write i_k for the k^{th} bit of an integer $i \in \{0, \ldots, N\}$, where $k \in \{0, \ldots, m-1\}$. We also have the polynomials:

$$F_{k,1}(X) := l_k \cdot X + a_k, F_{k,0}(X) := (1 - l_k) \cdot X - a_k,$$

for $k \in \{0, \ldots, m-1\}$, as well as the products:

$$P_i(X) := \prod_{k=0}^{m-1} F_{k,i_k}(X) = \delta_{i,l} \cdot X^n + \sum_{k=0}^{m-1} P_{i,k} \cdot X^k,$$

for $i \in \{0, \ldots, N-1\}$. The coefficients $P_{i,k}$ can be calculated in advance by the prover.

We now have:

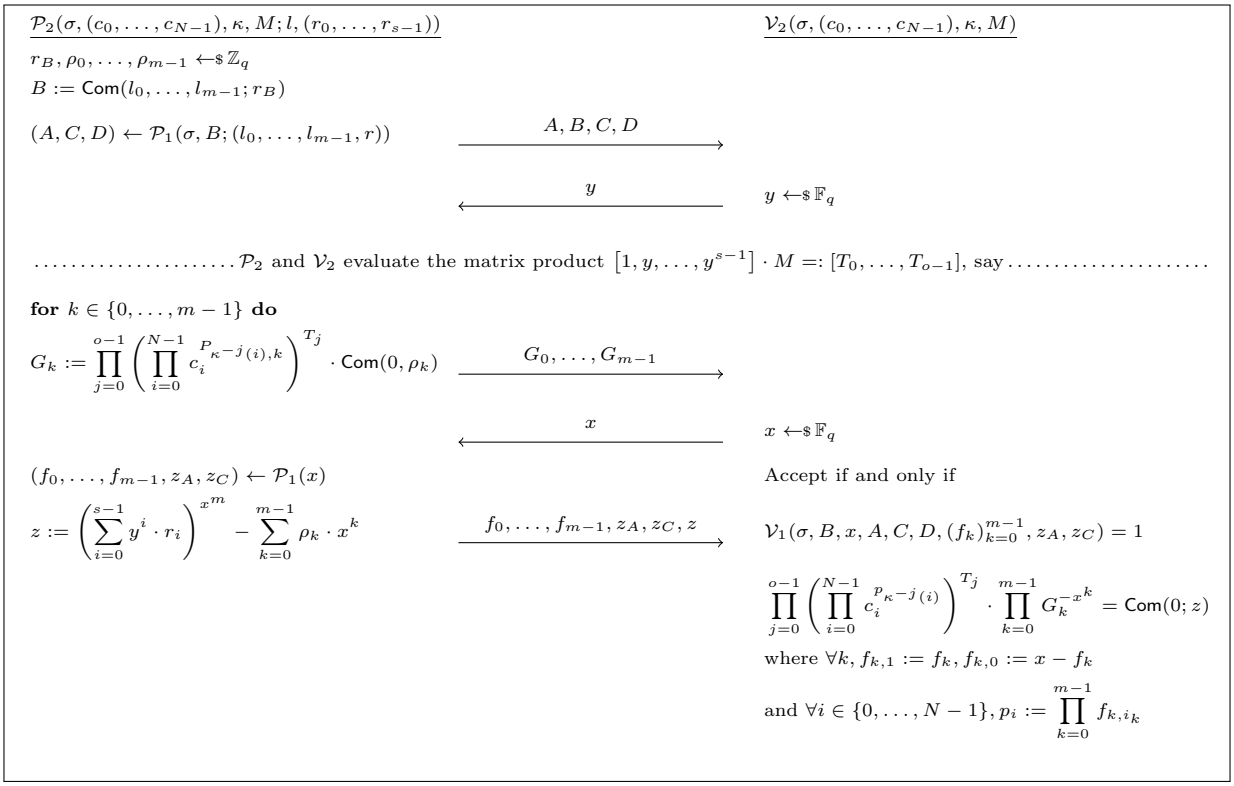

Figure 4: Protocol for the relation $\mathcal{R}_2$.

Example. Taking $\kappa = \mathrm{id} \in \mathbf{S}_N$ the identity permutation, and $M \colon \mathbb{F}_q \to \mathbb{F}_q$ the identity *map*, exactly recovers the original protocol of Groth and Kohlweiss [GK15].

The protocol $\Pi = (\mathsf{Setup}, \mathcal{P}_2, \mathcal{V}_2)$ of Fig. 4 is perfectly complete. This follows essentially by inspection; we note in particular that $(0; \sum_{i=0}^{s-1} y^i \cdot r_i)$ opens the matrix product

$$\left[\begin{array}{cccc} 1 & y & \dots & y^{s-1} \end{array}\right] \cdot \left[\begin{array}{c} M \end{array}\right] \cdot \begin{bmatrix} c_l \\ c_{\kappa(l)} \\ \vdots \\ c_{\kappa^{o-1}(l)} \end{bmatrix},$$

by hypothesis on the $r_0, \dots, r_{s-1}$.

Moreover, we have:

Theorem. *If* Com *is (perfectly) binding, then* Π *is (perfectly)* $(s, m+1)$*-special sound.*

Proof. We describe an extractor $\mathcal{X}$ which, given an $(s, m+1)$-tree of accepting transcripts, either returns a witness $(l, (r_0, \dots, r_{s-1}))$ or breaks the binding property of the commitment scheme Com. We suppose that $\sigma \leftarrow \mathsf{Setup}(1^\lambda)$ has been generated; we let u and tree be arbitrary. We essentially follow [GK15, Thm. 3], while introducing an additional (i.e., a *second*) Vandermonde inversion step. Details follow.

We first consider, for *fixed* y, accepting responses $(f_0, \dots, f_{m-1}, z_A, z_C, z)$ to $m+1$ distinct challenges x. With recourse to the extractor of Fig. 3 (see [BCC+15, §B.1]) and responses to 3 distinct challenges x, $\mathcal{X}$ obtains openings $(b_0, \dots, b_{m-1}; r_B)$ and $(a_0, \dots, a_{m-1}; r_A)$ of B and A (respectively) for which each $b_k \in \{0, 1\}$. The bits $l_k := b_k$ define the witness l. Moreover, *each* response $(f_k)_{k=0}^{m-1}$ either takes the form $(b_k \cdot x + a_k)_{k=0}^{m-1}$, or yields a violation of Com's binding property. Barring this latter contingency, $\mathcal{X}$ may construct using b_k and a_k polynomials $P_i(X)$, for $i \in \{0, \dots, N-1\}$—of degree m if and only if $i = l$—for which $p_i = P_i(x)$ for each x (where p_i are as computed by the verifier).

Using these polynomials, $\mathcal{X}$ may, for each x, re-write the final verification equation as:

$$\left(\prod_{j=0}^{o-1} (c_{\kappa^j(l)})^{T_j}\right)^{x^m} \cdot \prod_{k=0}^{m-1} (\widetilde{G}_k)^{x^k} = \mathsf{Com}(0; z),$$

for elements $\widetilde{G}_k$ which depend only on the polynomials $P_i(X)$ and the elements G_k (in particular, they don't depend on x). Exactly as in [GK15, Thm. 3], by inverting an $(m+1) \times (m+1)$ Vandermonde matrix containing the challenges x (and reading off its bottom row), $\mathcal{X}$ obtains a linear combination of the responses z, say z_y, for which:

$$\prod_{j=0}^{o-1} (c_{\kappa^j(l)})^{T_j} = \mathsf{Com}\,(0, z_y)\,.$$

In fact, an expression of this form can be obtained for *each* challenge y. Furthermore, now using the definition of $[T_0, \dots, T_{o-1}]$, we rewrite this expression's left-hand side as the matrix product:

$$\left[\begin{array}{cccc} 1 & y & \dots & y^{s-1} \end{array}\right] \cdot \left[\begin{array}{c} M \end{array}\right] \cdot \begin{bmatrix} c_l \\ c_{\kappa(l)} \\ \vdots \\ c_{\kappa^{o-1}(l)} \end{bmatrix} = \mathsf{Com}\,(0, z_y)\,.$$

Using expressions of this form for s distinct challenges y, and inverting a second Vandermonde matrix, $\mathcal{X}$ obtains combinations of the values z_y, say $r_0, \dots, r_{s-1}$, for which:

$$\left[\begin{array}{c} M \end{array}\right] \cdot \begin{bmatrix} c_l \\ c_{\kappa(l)} \\ \vdots \\ c_{\kappa^{o-1}(l)} \end{bmatrix} = \begin{bmatrix} \mathsf{Com}\,(0, r_0) \\ \mathsf{Com}(0, r_1) \\ \vdots \\ \mathsf{Com}(0; r_{s-1}) \end{bmatrix}.$$

This completes the extraction process. Finally, any adversary $\mathcal{A}$ who causes $\mathcal{X}$ to lose $\mathsf{Sound}^{(n_1,\ldots,n_\mu)}_{\mathcal{A},\mathcal{X},\Pi,\mathcal{R}}(\lambda)$ can be converted into an adversary $\mathcal{A}'$ who wins $\mathsf{Binding}_{\mathcal{A}',\mathsf{Com}}(\lambda)$ with the same probability. Indeed, on input params, $\mathcal{A}'$ can simulate a view for $\mathcal{A}$ by including params in a common reference string σ and giving it to $\mathcal{A}$. The outcome $\mathsf{Sound}^{(n_1,\ldots,n_\mu)}_{\mathcal{A},\mathcal{X},\Pi,\mathcal{R}}(\lambda) = 0$ if and only if $\mathcal{A}$'s tree causes $\mathcal{X}$ to extract a violation of the binding property; if this happens, $\mathcal{A}'$ simply returns the offending values m, r, m', r' directly. □

Finally:

Theorem. *If* Com *is (perfectly) hiding, then* Π *is (perfectly) special honest verifier zero knowledge.*

Proof. We describe a PPT simulator $\mathcal{S}$ which outputs accepting transcripts. Given input σ and u (as well as the verifier's randomness ρ, which explicitly determines the challenges y and x), $\mathcal{S}$ first randomly generates $B \leftarrow \mathsf{Com}(0,\ldots,0)$, and invokes the simulator of [BCC$^+$15, §B.1] on B and x to obtain values $A, C, D, z_A, z_C, f_0, \ldots, f_{m-1}$. $\mathcal{S}$ then randomly selects z, and, for each $k \in \{1,\ldots,m-1\}$, assigns to $G_k \leftarrow \mathsf{Com}(0)$ a random commitment to 0. Finally, $\mathcal{S}$ sets

$$G_0 := \left(\prod_{i=0}^{N-1} c_i^{p_{\kappa^{-j}(i)}}\right)^{T_j} \cdot \prod_{k=1}^{m-1} G_k^{-x^k} \cdot \mathsf{Com}(0;-z),$$

where $[T_0,\ldots,T_{o-1}]$ and $(p_i)_{i=0}^{m-1}$ are computed exactly as is prescribed for the verifier.

As in the proof of the SHVZK of Fig. 3, the elements $f_0,\ldots,f_{m-1},z_A,z_C,z$ are distributed identically to those in real transcripts; conditioned on B, as well as on C (which Fig. 3's simulator must construct), A and D are also distributed identically. Furthermore, conditioned on these values as well as those of $G_1,\ldots,G_{m-1}$, G_0 is uniquely determined by the main verification equation. This leaves the commitments $C, B, G_1, \ldots, G_{m-1}$. Any adversary who distinguishes simulated proofs from real ones therefore does so on the basis of these commitments.

We define a succession of "hybrid SHVZK experiments", each of which uses the random bit $b \in \{0,1\}$ to determine the construction strategy of one further among the commitments $C, B, G_1, \ldots, G_{m-1}$. An adversary $\mathcal{A}$ for any particular such hybrid immediately yields an adversary $\mathcal{A}'$ for $\mathsf{Hiding}_{\mathcal{A}',\mathsf{com}}(\lambda)$, via a direct reduction ($\mathcal{A}'$ simulates an execution of the hybrid experiment on $\mathcal{A}$, using the commitment oracle's challenge in place of the random element). The probability that $\mathcal{A}$ wins $\mathsf{SHVZK}_{\mathcal{A},\mathcal{S},\Pi,\mathcal{R}}(\lambda)$, when expressed in terms of those of the various hybrids, yields an expression which directly depends on that of $\mathsf{Hiding}_{\mathcal{A}',\mathsf{com}}(\lambda)$. □

4.3 Efficiency

We discuss the efficiency of our protocol, and argue that it can be computed in quasilinear time for both the prover and the verifier. In order to facilitate fair comparison, we assume throughout that only "elementary" field, group, and polynomial operations are used (in contrast with [GK15], who rely on multi-exponentiation algorithms and unspecified "fast polynomial multiplication techniques").

4.3.1 Analysis of [GK15]

We begin with an analysis of [GK15]. The prover and verifier may *naïvely* compute the polynomials $P_i(X)$ and the evaluations p_i in $O(N \log^2 N)$ and $O(N \log N)$ time, respectively. We claim that the prover and verifier can compute $(P_i(X))_{i=0}^{N-1}$ and $(p_i)_{i=0}^{N-1}$ (respectively) in $O(N \log N)$ and $O(N)$ time. (These are clearly optimal, in light of the output sizes.) To this end, we informally describe an efficient recursive algorithm, which closely evokes those used in bit reversal (see e.g., Jeong and Williams [JW90]).

Having constructed the linear polynomials $F_{k,1}(X)$ and $F_{k,0}(X)$ for $k \in \{0,\ldots,m-1\}$, the $P_i(X)$ arise from a procedure which, essentially, arranges the "upward paths" through the array $F_{k,j}(X)$ ($k \in \{0,\ldots,m-1\}, j \in \{0,1\}$) into a binary tree of depth m. Each leaf i gives the product $\prod_{k=0}^{m-1} F_{k,i_k}(X) = P_i(X)$, which can be written into the i^{th} index of a global array (the index i can be kept track of throughout the recursion, using bitwise operations). Each *edge* of this tree, on the other hand, represents the multiplication of an $O(\log N)$-degree "partial product" by a linear polynomial; we

conclude that the entire procedure takes $O(N \log N)$ time. (The m multi-exponentiations of $c_0, \ldots, c_{N-1}$ by $P_{i,k}$—conducted during the construction of the G_i—also take $O(N \log N)$ time.)

The verifier of [GK15] can be implemented $O(N)$ time. Indeed, the same binary recursive procedure—applied now to the *evaluations* $f_{k,j}$—takes $O(N)$ time, as in this setting the products don't grow as the depth increases, and each partial product can be extended in $O(1)$ time.

4.3.2 Efficiency analysis of main protocol

We turn to the protocol of Fig. 4. Its communication complexity is clearly $O(\log N)$, and in fact is identical to that of [BCC+15] (in its radix $n = 2$ variant).

Its runtime, however, is a bit delicate, and depends in particular on how the map M grows with N. Indeed—even assuming that the image dimension $s \leq o$, which doesn't impact generality—M could take as much as $\Theta(N^2)$ space to represent; the evaluation of $[1, y, \ldots y^{s-1}] \cdot M$ could also take $\Theta(N^2)$ time in the worst case. To eliminate these cases (which are perhaps of theoretical interest only), we insist that M has only $O(N)$ nonzero entries as N grows. This ensures that the expressions $[1, y, \ldots y^{s-1}] \cdot M$ can be evaluated in linear time. (We note that the *unevaluated* matrix product—represented as a matrix in the indeterminate Y—can be computed in advance of the protocol execution, and stored, or even "hard-coded" into the implementation; under our assumption, it will occupy $O(N)$ space, and require $O(N)$ time to evaluate during each protocol execution.)

This condition holds in particular if the number of rows $s = O(1)$. Importantly, it also holds in significant applications (like in Anonymous Zether) for which $s = \Theta(N)$; this latter fact makes the Vandermonde trick non-vacuous.

Even assuming this condition on M, a naïve implementation of the protocol of Fig. 4 uses $\Theta(N^2 \log N)$ time for the prover and $\Theta(N^2)$ time for the verifier (in the worst case $o = \Theta(N)$). It is therefore surprising that, imposing *only* the aforementioned assumption on M, we nonetheless attain:

Theorem. *Suppose that the number of nonzero entries of M grows as $O(N)$. Then the protocol of Fig. 4 can be implemented in $O(N \log^2 N)$ time for the prover and $O(N \log N)$ time for the verifier.*

Proof. We first argue that it suffices to consider only the "canonical" case $\kappa = (0, 1, \ldots, N-1)$. To this end, we fix a $\kappa' \in \mathbf{S}_N$, not necessarily equal to κ; we assume first that κ' is an N-cycle, say with cycle structure $(\kappa'_0, \kappa'_1, \ldots, \kappa'_{N-1})$. Given desired common inputs $(\sigma, (c_0, c_1, \ldots, c_{N-1}), \kappa', M)$, and private inputs $(l', (r_0, \ldots, r_{s-1}))$, we observe that the prover and verifier's purposes are equally served by running Fig. 4 instead on the common inputs $(\sigma, (c_{\kappa'_0}, c_{\kappa'_1}, \ldots, c_{\kappa'_{N-1}}), \kappa, M)$ and private inputs $(l, (r_0, \ldots, r_{s-1}))$, where l is such that $\kappa'_l = l'$.

Any arbitrary *free* permutation $\kappa'' \in \mathbf{S}_N$ (with order o, say), now, is easily seen to be an iterate (with exponent N/o) of some N-cycle κ'; in fact, one such κ' can easily be constructed in linear time by "collating" through the cycles of κ''. On desired inputs $(\sigma, (c_0, c_1, \ldots, c_{N-1}), \kappa'', M; l', (r_0, \ldots, r_{s-1}))$, then, the prover and verifier may use the above reduction to execute $(\sigma, (c_0, c_1, \ldots, c_{N-1}), \kappa', M; l', (r_0, \ldots, r_{s-1}))$; they may then discard all "rows" except those corresponding to indices $j \in \{0, \ldots, N-1\}$ for which $N/o \mid j$.

We therefore turn now to the case $\kappa = (0, 1, \ldots, N-1)$, whose analysis, by the above, suffices for arbitrary κ. The verifier's bottleneck is the evaluation of the matrix action

$$\left[\ e_j\ \right]_{j=0}^{N-1} := \left[\ p_{\kappa^{-j}(i)}\ \right]_{j,i=0}^{N-1} \cdot \left[\ c_i\ \right]_{i=0}^{N-1}.$$

Yet by hypothesis on κ, the matrix $\left[\ p_{\kappa^{-j}(i)}\ \right]_{j,i=0}^{N-1}$ is a circulant matrix, and the above equation's right-hand side is a convolution in the sense of, e.g., [Nus82, (8.1)] (we assume here that N is a power of 2 and that $N \mid (q-1)$; see [Nus82, Thm. 8.2]). Though the curve $E(\mathbb{F}_p)$ in which the points c_i reside is isomorphic (as an $\mathbb{F}_q$-module) to $\mathbb{F}_q$, this is isomorphism is not canonical, and isn't efficiently computable. Nonetheless, its module structure alone (and not its ring structure) suffices; following the notation of [Nus82, (8.5)], we observe that the elements h_i appear never with other h_i, but only with the ring elements g and x_i. Put differently, both the number-theoretic transform [Nus82, (8.2)] and the "multiplication" (i.e., multi-exponentiation) of $\overline{H}_k$ by $\overline{X}_k$ can be conducted with recourse to h_n's module structure alone. The verifier may thus evaluate this product in $O(N \log N)$ time using standard algorithms for the fast Fourier transform.

We turn to the prover, who must compute the m matrix evaluations:

$$\left[\ P_{\kappa^{-j}(i),k}\ \right]_{j,i=0}^{N-1} \cdot \left[\ c_i\ \right]_{i=0}^{N-1},$$

for each $k \in \{0, \ldots, m-1\}$ (in the process of computing the G_k). Using identical reasoning, we see that these can be computed with the aid of m parallel NTT-aided convolutions; the prover's complexity is therefore $O(N \log^2 N)$.

The remaining work, for both the prover and verifier, amounts to evaluating $[T_0, \ldots, T_{o-1}] := \left[1, y, \ldots, y^{s-1}\right] \cdot M$. By hypothesis on M, this can be done in linear time. □

5 An Alternative Ring Signature

We describe an alternative procedure for ring signatures, which adapts that of [GK15, §4].

In our treatment, we consider anonymity *only* with respect to adversarially chosen keys, and in fact our protocol is not secure in the stronger setting of full key exposure. Nonetheless, this limitation is acceptable in—and in fact is inherent to—our main application (namely Anonymous Zether), as we shall argue below. Moreover, our protocol admits important flexibility not offered by that of [GK15, §4]; informally, it can be run concurrently over *multiple* rings, while ensuring in each case that the same secret key is used. We further discuss both of these matters below, in connection with Anonymous Zether.

For now, we introduce an interactive protocol for the relation:

$$\mathcal{R}_3 = \left\{(\sigma, (y_0, \ldots, y_{N-1}); l, \mathsf{sk}) : y_l = g^{\mathsf{sk}}\right\}.$$

We follow [GK15, Fig. 2], except that we replace the final revelation of z by a Schnorr knowledge-of-exponent identification protocol (see for example [KL15, Fig. 12.2]). Explicitly:

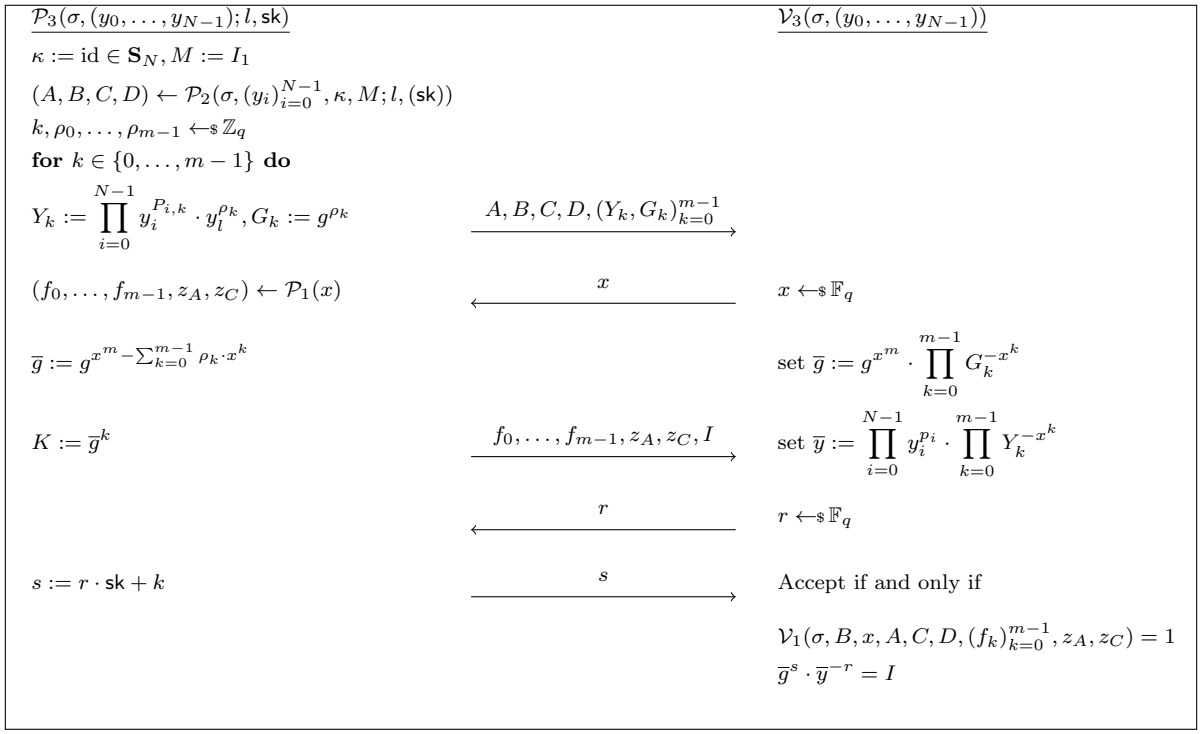

Figure 5: Protocol for the relation $\mathcal{R}_3$.

In effect, the prover sends correction terms for *both* y_l and g; the prover and verifier then conduct a Schnorr protocol on the "corrected" elements $\overline{y}$ and $\overline{g}$. We remark that the correction terms Y_k use the blinding scalars ρ_k in the exponent of y_l, and *not* of g.

We define a ring signature $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ by applying the Fiat–Shamir transform to Fig. 5 (see [KL15, Cons. 12.9]). $\mathsf{Gen}(1^\lambda)$ runs a group generation procedure $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^\lambda)$ and the commitment scheme setup, and chooses a function $H\colon \{0,1\}^* \to \mathbb{F}_q$. (In our security analyses below, we model H as a random oracle.) In addition, we define $x = H(A, B, C, D, (Y_k, G_k)_{k=0}^{m-1}, m)$ as well as $r = H(x, I)$; the verifier, given a transcript, checks also that these queries were computed correctly.

Π is complete, as can be seen from the completeness of the Schnorr signature, and from the discrete logarithm relation $\overline{y} = \overline{g}^{\mathsf{sk}}$. In fact, $\overline{g} = g^{x^m - \sum_{k=0}^{m-1} \rho_k \cdot x^k}$ and $\overline{y} = y_l^{x^m - \sum_{k=0}^{m-1} \rho_k \cdot x^k}$; the relation immediately follows. Moreover:

Theorem. *If* Com *is computationally binding and the discrete logarithm problem is hard with respect to* $\mathcal{G}$*, then* Π *is unforgeable with respect to insider corruption.*

Proof. We fix a polynomial $N(\cdot)$ and an adversary $\mathcal{A}$ targeting $\mathsf{UnforgeIC}^{N(\cdot)}_{\mathcal{A},\Pi}(\lambda)$; assuming that Com is binding, we define an adversary $\mathcal{A}'$ which wins $\mathsf{DLog}_{\mathcal{A}',\mathcal{G}}(\lambda)$ with polynomially related probability. In short, $\mathcal{A}'$ simulates an execution of $\mathsf{UnforgeIC}^{N(\cdot)}_{\mathcal{A},\Pi}(\lambda)$ on $\mathcal{A}$; if $\mathcal{A}'$ is able to obtain a $(2m+1, 2)$-tree of valid signatures on the right witness (and barring a violation of the binding property), $\mathcal{A}'$ returns a discrete logarithm with probability 1. The difficult part resides in this latter extraction. Indeed, after seeing x, the prover could in principle choose sk adaptively, and the extraction of $\log(y_l)$ demands the interpolation of a *rational* function (generalizing the Vandermonde-based polynomial interpolation of e.g. [GK15]).

$\mathcal{A}'$ works as follows. It is given $\mathbb{G}, q, g, h$ and Com as input.

1. Generate keys $(y_i, \mathsf{sk}_i)_{i=0}^{N(\lambda)-1}$ using $\mathsf{Gen}(1^\lambda)$. For a randomly chosen $l \in \{0, \ldots, N(\lambda) - 1\}$, re-assign to y_l the discrete logarithm challenge h. Finally, give the modified list $S := (y_i)_{i=0}^{N(\lambda)-1}$ to $\mathcal{A}$.
2. Respond to each of $\mathcal{A}$'s random oracle queries with a random element of $\mathbb{F}_q$.
3. For each oracle query $\mathsf{Osign}(s, m, R)$ for which $s \neq l$, simply compute a signature as specified by Fig. 5. If $s = l$, replace the final Schnorr protocol by a simulation (exactly as in [KL15, p. 456]).
4. For each query $\mathsf{Corrupt}(i)$ for which $i \neq l$, return sk_i; if $i = l$, abort.
5. When $\mathcal{A}$ outputs (R^*, m^*, σ^*), by rewinding it and freshly simulating its random oracle queries, obtain a $(2m+1, 2)$-tree of signatures (i.e., for $2m+1$ values of x and, for each one, 2 values of r). If any of the $(2m+1) \cdot 2$ resulting signatures fail to meet the winning condition of $\mathsf{UnforgeIC}^{N(\cdot)}_{\mathcal{A},\Pi}$, or if any collisions occur between the x or r challenges (respectively), then abort.
6. By running the extractor of Fig. 3, obtain openings $b_0, \ldots, b_{m-1}$, $a_0, \ldots, a_{m-1}$ of the initial commitments B and A for which $b_k \in \{0, 1\}$. If for *any* x it holds that $f_k \neq b_k \cdot x + a_k$ for some k, return the corresponding violation of the binding property of the commitment $B^x A$. Otherwise, determine whether the element $y^* \in R^*$ (say) whose index is given (in binary) by $b_0, \ldots, b_{m-1}$ equals y_l. If it doesn't, abort.
7. Given these conditions, use the following procedure to obtain, with probability 1, a discrete log $\mathsf{sk} \in \mathbb{F}_q$ for which $g^{\mathsf{sk}} = y_l = h$. By renaming variables, we may assume that $S = R^*$ and that $y^* = y_l$. Use the openings b_k and a_k to recover the polynomials $P_i(X)$, and hence representations, valid for *each* x, of the form:

$$(\overline{y}, \overline{g}) = \left(y_l^{x^m} \cdot \prod_{k=0}^{m-1} \widetilde{Y_k}^{-x^k}, g^{x^m} \cdot \prod_{k=0}^{m-1} G_k^{-x^k} \right),$$

for easily computable elements $\widetilde{Y_k}$. For each particular x, meanwhile, use the standard Schnorr extractor on the two equations $\overline{g}^s \cdot \overline{y}^{-r} = I$ to obtain some quantity $\widehat{\mathsf{sk}}$ (possibly depending on x) for which $\overline{g}^{\widehat{\mathsf{sk}}} = \overline{y}$. For each x, we thus obtain an $\widehat{\mathsf{sk}}$ for which:

$$y_l^{x^m} \cdot \prod_{k=0}^{m-1} \widetilde{Y_k}^{-x^k} = \left(g^{x^m} \cdot \prod_{k=0}^{m-1} G_k^{-x^k} \right)^{\widehat{\mathsf{sk}}}.$$

We apply a rational function interpolation procedure. Taking discrete logarithms of this equation with respect to g reveals an algebraic equation in two variables, with unknown coefficients, which $(x, \widehat{\mathsf{sk}})$ satisfies:

$$\log(y_l) \cdot x^m - \sum_{k=0}^{m-1} \log(\widetilde{Y}_i) \cdot x^k = \left(1 \cdot x^m - \sum_{k=0}^{m-1} \log(G_k) \cdot x^k\right) \cdot \widehat{\mathsf{sk}}. \tag{1}$$

We refer to the section *Cauchy interpolation* of von zur Gathen and Gerhard [vzGG13, §5.8]. We reside exactly in the closely related setting [vzGG13, §5.8, (21)], in which no division is performed. We require a relaxed version of the uniqueness result [vzGG13, Cor. 5.18, (ii)], in which (21) is considered instead of (20) (and the "canonical form" condition is dropped). For thoroughness, we outline the details.

View the $2m+1$ satisfying pairs $(x, \widehat{\mathsf{sk}})$ of (1) as an instance of [vzGG13, (21)] (using the parameters $n = 2m+1$, $k = m+1$). Denote by r and t the (unknown) polynomials, in the indeterminate X, which appear in (1)'s left- and right-hand sides. Use the Extended Euclidean Algorithm to construct polynomials $r_j, t_j \in \mathbb{F}_q[X]$, exactly as prescribed in the statement of [vzGG13, Cor. 5.18]. Adapting the proof of [vzGG13, Thm. 5.16, (ii)] (we note that its condition still holds), we see that there exists some nonzero $\alpha \in \mathbb{F}_q[X]$ for which

$$(r, t) = (\alpha \cdot r_j, \alpha \cdot t_j).$$

Though α's existence guarantee is not constructive, we note nonetheless that because t is monic, α's leading coefficient must equal τ^{-1} (as in [vzGG13, §5.18, (ii)], we denote by $\tau = \mathrm{lc}(t_j)$ the leading coefficient of t_j). Therefore, the desired discrete logarithm $\log(y_l) =: \mathsf{sk}$ is determined concretely as $\tau^{-1} \cdot \mathrm{lc}(r_j)$.

We first analyze a modified experiment $\mathsf{UnforgeIC}'^{N(\cdot)}_{\mathcal{A},\Pi}$, which differs from the standard $\mathsf{UnforgeIC}^{N(\cdot)}_{\mathcal{A},\Pi}$ *only* in that the experimenter, upon receiving the final message (R^*, m^*, σ^*), rewinds $\mathcal{A}$ so as to obtain a $(2m+1, 2)$-tree of signatures, and imposes the winning condition of $\mathsf{UnforgeIC}^{N(\cdot)}_{\mathcal{A},\Pi}$ on all $(2m+1) \cdot 2$ leaves. The experiment also returns 0 if any of the x or r values feature collisions. Using an argument exactly analogous to that of the proof of [KL15, Thm. 12.11] (and using Jensen's inequality twice), we see that $\Pr[\mathsf{UnforgeIC}'^{N(\cdot)}_{\mathcal{A},\Pi} = 1] \geq \Pr[\mathsf{UnforgeIC}^{N(\cdot)}_{\mathcal{A},\Pi}(\lambda) = 1]^{(2m+1)\cdot 2} - \mathsf{negl}(\lambda)$, where negl is a negligible function.

We now claim that if that Com is binding, the event in which both $\mathsf{UnforgeIC}'^{N(\cdot)}_{\mathcal{A},\Pi}(\lambda) = 1$ *and* the $(2m+1)\cdot 2$ signatures (R^*, m^*, σ^*) yield a violation of the binding property occurs with negligible probability (say, given by negl'). Indeed, an adversary $\mathcal{A}$ targeting $\mathsf{UnforgeIC}'^{N(\cdot)}_{\mathcal{A},\Pi}$ immediately yields an adversary $\mathcal{A}''$ targeting $\mathsf{Binding}_{\mathcal{A}'',\mathsf{Com}}$, which, on input Com, runs $\mathsf{UnforgeIC}'^{N(\cdot)}_{\mathcal{A},\Pi}(\lambda)$, and returns the violation (winning $\mathsf{Binding}_{\mathcal{A}'',\mathsf{Com}}$) exactly in this situation.

We finally point out that, among those executions of $\mathsf{UnforgeIC}'^{N(\cdot)}_{\mathcal{A},\Pi}$ which $\mathcal{A}$ wins and in which no binding violation occurs, that element $y^* \in R^*$ whose index in R^* is given by the binary representation $b_0, \ldots, b_{m-1}$ matches a uniform element $y_l \in S$ (chosen in advance) with probability exactly $\frac{1}{N(\lambda)}$; furthermore, in this latter setting, $\mathcal{A}$ necessarily never queries $\mathsf{Corrupt}(l)$ (this follows from the simple requirement $R^* \subset S \backslash C$).

We turn now to the simulation given above. We claim that $\mathcal{A}'$ answers $\mathcal{A}$'s random oracle queries inconsistently in at most negligibly many of its simulations. Indeed, inconsistency results only in the event that $\mathcal{A}'$, upon receiving a query $\mathsf{Osign}(l, m, R)$, simulates the Schnorr proof using random quantities s, r for which $\mathcal{A}$ has already queried (x, I) (where $I := \overline{g}^s \cdot \overline{y}^{-r}$) and for which $H(x, I) \neq r$. This happens with negligible probability (say $\mathsf{negl}''(\lambda)$).

Barring this event, and that in which $\mathcal{A}'$ aborts, $\mathcal{A}$'s view in the simulation exactly matches that in $\mathsf{UnforgeIC}'^{N(\cdot)}_{\mathcal{A},\Pi}$. Moreover, as long as no abort takes place and no violation of the binding property occurs,

$\mathcal{A}'$ necessarily wins $\mathsf{DLog}_{\mathcal{A}',\mathcal{G}}(\lambda)$. We thus see that:

$$\begin{aligned}
\Pr[\mathsf{DLog}_{\mathcal{A}',\mathcal{G}}(\lambda)=1] &\geq \Pr[\mathsf{UnforgeIC}'^{N(\cdot)}_{\mathcal{A},\Pi}(\lambda)=1 \wedge \text{No binding violation is obtained} \wedge y^* = y_l] - \mathsf{negl}''(\lambda) \\
&\geq \frac{1}{N(\lambda)} \cdot \Pr[\mathsf{UnforgeIC}'^{N(\cdot)}_{\mathcal{A},\Pi}(\lambda)=1 \wedge \text{No binding violation is obtained}] - \mathsf{negl}''(\lambda) \\
&\geq \frac{1}{N(\lambda)} \cdot \Big(\Pr[\mathsf{UnforgeIC}'^{N(\cdot)}_{\mathcal{A},\Pi}(\lambda)=1] - \mathsf{negl}'(\lambda)\Big) - \mathsf{negl}''(\lambda) \\
&\geq \frac{1}{N(\lambda)} \cdot \Big(\Pr[\mathsf{UnforgeIC}^{N(\cdot)}_{\mathcal{A},\Pi}(\lambda)=1]^{(2m+1)\cdot 2} - \mathsf{negl}(\lambda) - \mathsf{negl}'(\lambda)\Big) - \mathsf{negl}''(\lambda).
\end{aligned}$$

The result immediately follows from discrete logarithm assumption on $\mathcal{G}$. □

We turn to anonymity. We note that the interactive protocol of Fig. 5 is *not* zero-knowledge, or even witness-indistinguishable. Indeed, having chosen a statement and candidate witnesses $(l_0, \mathsf{sk}_0), (l_1, \mathsf{sk}_1)$, and given tr, $\mathcal{A}$ may simply return whichever $b' \in \{0,1\}$ satisfies:

$$y_{l_{b'}}^{x^m} \cdot \prod_{k=0}^{m-1} Y_k \cdot (G_k)^{-\mathsf{sk}_{b'}} \stackrel{?}{=} \prod_{i=0}^{N-1} y_i^{p_i}.$$

Heuristically, the failure stems from the pairs (Y_k, G_k), which are "El Gamal ciphertexts" under y_l and can be retrospectively "decrypted" if (and only if) sk is exposed. Analogously, Π is not anonymous against full key exposure (recall [BKM09, Def. 4]). Nonetheless, we have:

Theorem. *If* Com *is computationally hiding and the DDH problem is hard relative to* $\mathcal{G}$*, then* Π *is anonymous with respect to adversarially chosen keys.*

Proof. We convert an adversary $\mathcal{A}$ attacking $\mathsf{AnonACK}^{N(\cdot)}_{\mathcal{A},\Pi}$ into an adversary $\mathcal{A}'$ who, assuming that Com is hiding, wins $\mathsf{DDH}_{\mathcal{A}',\mathcal{G}}$ with polynomially related probability.

$\mathcal{A}'$ works as follows. It is given $\mathbb{G}, q, g, h_1, h_2, h'$ and Com as input.

1. Generate keys $(y_i, \mathsf{sk}_i)_{i=0}^{N(\lambda)-1}$ using $\mathsf{Gen}(1^\lambda)$. For a random index $l \in \{0, \ldots, N(\lambda)-1\}$, re-assign $y_l := h_1$. Finally, give the modified list $S := (y_i)_{i=0}^{N(\lambda)-1}$ to $\mathcal{A}$.
2. Respond to each of $\mathcal{A}$'s random oracle queries with a random element of $\mathbb{F}_q$.
3. For each oracle query $\mathsf{Osign}(s, m, R)$ for which $s \neq l$, simply compute a signature as specified by Fig. 5. If $s = l$, replace the final Schnorr protocol by a simulation (as in [KL15, p. 456]).
4. When $\mathcal{A}$ outputs m, i_0, i_1, R, choose a uniform bit $b \in \{0,1\}$. If $i_b \neq l$, abort and return a random bit.
5. If on the other hand $i_b = l$, construct a signature σ exactly as specified in Fig. 5, except set:

$$(Y_k, G_k) := \left(\prod_{i=0}^{N-1} y_i^{P_{i,k}} \cdot (h')^{\rho_k}, (h_2)^{\rho_k}\right)$$

   for each $k \in \{0, \ldots, m-1\}$. Moreover, simulate the final Schnorr signature, as in [KL15, p. 456].
6. When $\mathcal{A}$ outputs a bit b', return whether $b' \stackrel{?}{=} b$.

For notational ease, we first introduce a modified experiment $\mathsf{AnonACK}'^{N(\cdot)}_{\mathcal{A},\Pi}$, which differs from the standard $\mathsf{AnonACK}^{N(\cdot)}_{\mathcal{A},\Pi}$ *only* in the construction strategy of the signature σ. The experimenter proceeds as follows. After receiving i_0 and i_1 and generating the bit $b \in \{0,1\}$, it generates a *further* uniform bit b''. If $b'' = 1$, the experimenter proceeds exactly as in $\mathsf{AnonACK}^{N(\cdot)}_{\mathcal{A},\Pi}$. Otherwise, the experimenter uses independent

randomnesses χ_k in the elements Y_k (i.e., instead of re-using ρ_k), and furthermore simulates the final Schorr protocol.

We note that, barring inconsistencies in the random oracle queries implicit in $\mathcal{A}'$'s Schnorr simulations (which occur in negligibly many among $\mathcal{A}'$'s executions, say negl) and an abort by $\mathcal{A}'$ (which takes place in exactly $\frac{1}{N(\lambda)}$ of executions in which no inconsistencies occur), $\mathcal{A}$'s view in its simulation by $\mathcal{A}'$ exactly matches its view in the modified experiment $\mathsf{AnonACK}'^{N(\cdot)}_{\mathcal{A},\Pi}$. Indeed, the possibilities $b'' \in \{0,1\}$ in $\mathsf{AnonACK}'^{N(\cdot)}_{\mathcal{A},\Pi}$ correspond exactly to the two possibilities of the DDH experimenter's random bit in $\mathcal{A}'$'s simulation.

We now address the winning probability of $\mathcal{A}$ in $\mathsf{AnonACK}'^{N(\cdot)}_{\mathcal{A},\Pi}$. If $b'' = 1$, this probability exactly matches that of $\mathcal{A}$ in $\mathsf{AnonACK}^{N(\cdot)}_{\mathcal{A},\Pi}$ (by construction of the former experiment). If on the other hand $b'' = 0$, we claim—assuming now that Com is hiding—that $\Pr[\mathsf{AnonACK}'^{N(\cdot)}_{\mathcal{A},\Pi}(\lambda) = 1]$ differs at most negligibly from $\frac{1}{2}$ (say by negl'). Indeed, $\mathcal{A}$'s task in this setting (barring a random oracle inconsistency, which again is negligibly probable) is exactly that of distinguishing distributions which differ *only* in the commitments A, B, C, D (we note that when $b'' = 0$, both Y_k and G_k are uniform and independent). Any adversary $\mathcal{A}$ which targets this task thus immediately yields an adversary $\mathcal{A}''$ targeting $\mathsf{Hiding}_{\mathcal{A}'',\mathsf{com}}(\lambda)$; on input params, $\mathcal{A}''$ simply simulates an execution of $\mathsf{AnonACK}'^{N(\cdot)}_{\mathcal{A},\Pi}$ for which necessarily $b'' = 0$, using the oracle challenge com in place of one of the commitments B, C (and constructing A and D as in the simulator of Fig. 3).

Putting these facts together, and splitting $\Pr[\mathsf{AnonACK}'^{N(\cdot)}_{\mathcal{A},\Pi}(\lambda) = 1]$ along the possibilities $b'' \in \{0,1\}$, we see that:

$$\begin{aligned}\Pr[\mathsf{DDH}_{\mathcal{A}',\mathcal{G}}(\lambda) = 1] - \frac{1}{2} &\geq \frac{1}{N(\lambda)} \cdot \left(\Pr[\mathsf{AnonACK}'^{N(\cdot)}_{\mathcal{A},\Pi}(\lambda) = 1] - \frac{1}{2}\right) - \mathsf{negl}(\lambda) \\ &\geq \frac{1}{N(\lambda)} \cdot \left(\frac{1}{2} \cdot \left(-\mathsf{negl}'(\lambda)\right) + \frac{1}{2} \cdot \left(\Pr[\mathsf{AnonACK}^{N(\cdot)}_{\mathcal{A},\Pi}(\lambda) = 1] - \frac{1}{2}\right)\right) - \mathsf{negl}(\lambda).\end{aligned}$$

The result immediately follows from the DDH assumption on $\mathcal{G}$. □

6 Application: Anonymous Zether

6.1 Review of basic and anonymous Zether

We briefly summarize both basic and anonymous Zether; for further details we refer to [BAZB].

Zether's global state consists of a mapping acc from El Gamal public keys to El Gamal ciphertexts; y's table entry contains an encryption of y's balance b (in the exponent). Schematically, we can write:

$$\mathsf{acc}\colon \mathbb{G} \to \mathbb{G}^2,$$
$$y \mapsto \mathsf{acc}[y] = \mathsf{Enc}_y(b, r) = \left(g^b y^r, g^r\right),$$

for some randomness r which y in general does not know. (For details on the synchronization issues surrounding "epochs", we refer to [BAZB].)

6.1.1 Basic Zether

In "basic" (non-anonymous) Zether, a non-anonymous sender y may transfer funds to a non-anonymous recipient $\overline{y}$. To do this, y should publish the public keys y and $\overline{y}$, as well as a pair of ciphertexts (C, D) and $(\overline{C}, D)$ (the randomness may be shared). These should encrypt, under y and $\overline{y}$'s keys, the quantities g^{b^*} and g^{-b^*}, respectively, for some integer $b^* \in \{0, \ldots \mathsf{MAX}\}$ (MAX is a fixed constant of the form $2^n - 1$). To apply the transfer, the administering system (e.g., smart contract) should group-subtract (C, D) and $(\overline{C}, D)$ from y and $\overline{y}$'s account balances (respectively). We denote by (C_{Ln}, C_{Rn}) y's balance *after* the homomorphic deduction is performed.

Finally, the prover should prove knowledge of:

- sk for which $g^{\mathsf{sk}} = y$ (knowledge of secret key),
- r for which:
  - $g^r = D$ (knowledge of randomness),
  - $(y \cdot \overline{y})^r = (C \cdot \overline{C})$ (ciphertexts encrypt opposite balances),
- b^* and b^* in $\{0, \ldots, \mathsf{MAX}\}$ for which $C = g^{b^*} \cdot D$ and $C_{Ln} = g^{b'} \cdot C_{Rn}$ (overflow and overdraft protection).

Formally, we have the relation below, which essentially reproduces [BAZB, (2)]:

$$\mathsf{st}_{\mathsf{ConfTransfer}} : \begin{gathered} \Big\{ (y, \overline{y}, C_{Ln}, C_{Rn}, C, \overline{C}, D; \mathsf{sk}, b^*, b', r) : \\ g^{\mathsf{sk}} = y \wedge C = g^{b^*} \cdot D^{\mathsf{sk}} \wedge C_{Ln} = g^{b'} \cdot C_{Rn}^{\mathsf{sk}} \wedge \\ D = g^r \wedge (y \cdot \overline{y})^r = C \cdot \overline{C} \wedge \\ b^* \in \{0, \ldots, \mathsf{MAX}\} \wedge b' \in \{0, \ldots, \mathsf{MAX}\} \Big\}. \end{gathered}$$

6.1.2 Anonymous Zether

In anonymous Zether [BAZB, §D], a sender may hide herself and the recipient in a larger "ring" $(y_i)_{i=0}^{N-1}$. To an observer, it should be impossible to discern which among a ring's members sent or received funds. Specifically, a sender should choose a list $(y_i)_{i=0}^{N-1}$, as well as indices idx_0 and idx_1 for which y_{idx_0} and y_{idx_1} belong to the sender and recipient, respectively. The sender should then publish this list, as well as a list of ciphertexts $(C_i, D)_{i=0}^{N-1}$ for which (C_{idx_0}, D) encrypts g^{b^*} under y_{idx_0}, (C_{idx_1}, D) encrypts g^{-b^*} under y_{idx_1}, and (C_i, D) for each $i \notin \{\mathsf{idx}_0, \mathsf{idx}_1\}$ encrypts g^0 under y_i. To apply the transfer, the contract should deduct each (C_i, D) from y_i's balance; we denote the list of *new* balances by $(C_{Ln,i}, C_{Rn,i})_{i=0}^{N-1}$.

Finally, the prover should prove knowledge of:

- $\mathsf{idx}_0, \mathsf{idx}_1 \in \{0, \ldots, N-1\}$ (sender's and recipient's secret indices),
- sk for which $g^{\mathsf{sk}} = y_{\mathsf{idx}_0}$ (knowledge of secret key),
- r for which:
  - $g^r = D$ (knowledge of randomness),
  - $(y_{\mathsf{idx}_0} \cdot y_{\mathsf{idx}_1})^r = C_{\mathsf{idx}_0} \cdot C_{\mathsf{idx}_1}$ (sender's and receiver's ciphertexts encrypt opposite balances),
  - for each $i \notin \{\mathsf{idx}_0, \mathsf{idx}_1\}$, $y_i^r = C_i$ (all ciphertexts other than the sender's and recipient's encrypt 0),
- b^* and b^* in $\{0, \ldots, \mathsf{MAX}\}$ for which $C_{\mathsf{idx}_0} = g^{b^*} \cdot D$ and $C_{Ln,\mathsf{idx}_0} = g^{b'} \cdot C_{Rn,\mathsf{idx}_0}$ (overflow and overdraft protection).

We group these facts into a formal relation, adapting [BAZB, (8)]. For technical reasons (described below), we actually prove a slight variant of this relation, in which N is required to be *even* and idx_0 and idx_1 are required to have opposite parity. Formally:

$$\mathsf{st}_{\mathsf{AnonTransfer}} : \begin{gathered} \Big\{ \big((y_i, C_i, C_{Ln,i}, C_{Rn,i})_{i=0}^{N-1}, D, u, g_{\mathsf{epoch}}; \mathsf{sk}, b^*, b', r, \mathsf{idx}_0, \mathsf{idx}_1 \big) : \\ g^{\mathsf{sk}} = y_{\mathsf{idx}_0} \wedge C_{\mathsf{idx}_0} = g^{b^*} D^{\mathsf{sk}} \wedge C_{Ln,\mathsf{idx}_0} = g^{b'} C_{Rn,\mathsf{idx}_0}^{\mathsf{sk}} \wedge \\ D = g^r \wedge (y_{\mathsf{idx}_0} \cdot y_{\mathsf{idx}_1})^r = C_{\mathsf{idx}_0} \cdot C_{\mathsf{idx}_1} \wedge \bigwedge_{i \notin \{\mathsf{idx}_0, \mathsf{idx}_1\}} y_i^r = C_i \wedge \\ g_{\mathsf{epoch}}^{\mathsf{sk}} = u \wedge b^* \in \{0, \ldots, \mathsf{MAX}\} \wedge b' \in \{0, \ldots, \mathsf{MAX}\} \wedge \\ N \equiv 0 \mod 2 \wedge \mathsf{idx}_0 \not\equiv \mathsf{idx}_1 \mod 2 \Big\}. \end{gathered} \tag{2}$$

6.2 Revisiting "Σ-Bullets"

We now describe the "Σ-Bullets" protocol of [BAZB], and our improvements. In fact, the protocol as described on page 48 of [BAZB, §G] is not complete as written; to see this, note that $\left(\frac{C^c}{D^{s_{\mathsf{sk}}}}\right)^{z^2} \cdot \left(\frac{C_{Ln}^c}{C_{Rn}^{s_{\mathsf{sk}}}}\right)^{z^3} = \left(D^{z^2} \cdot C_{Rn}^{z^3}\right)^{-k_{\mathsf{sk}}} \cdot g^{c\cdot(b^*\cdot z^2 + b'\cdot z^3)}$, whereas $g^{s_b} = g^{k_b} \cdot g^{c\cdot(b^*\cdot z^2 + b'\cdot z^3)}$. We thus consider the version implemented in the repository `bbuenz / BulletProofLib`. In this version, the the prover sends $A_t = \left(D^{z^2} \cdot C_{Rn}^{z^3}\right)^{k_{\mathsf{sk}}}$; the verifier then checks

$$g^{c\cdot\hat{t}} \cdot h^{c\cdot\tau_x} \stackrel{?}{=} g^{c\cdot\delta(y,z)} \cdot A_t \cdot {c_{\mathsf{commit}}}^{-1} \cdot \left(T_1^x \cdot T_2^{x^2}\right)^c, \tag{3}$$

where $c_{\mathsf{commit}} = \left(D^{z^2} \cdot C_{Rn}^{z^3}\right)^{s_{\mathsf{sk}}} \cdot \left(C^{z^2} \cdot C_{Ln}^{z^3}\right)^{-c}$.

6.2.1 Attacks on [BAZB]

Attack (Soundness)**.** Informally, nothing prevents the message of (C, D) from having an "h part". Suppose that the prover were to construct (C, D) so as to encrypt the message $g^{b^*}h^{\gamma}$, for some nonzero γ (and b^* as usual). (The recipient's ciphertext, $(\overline{C}, D)$, would encrypt $g^{-b^*}h^{-\gamma}$.) By adding the constant term $z^2 \cdot \gamma$ to τ_x (as implemented in `BulletProofLib`, τ_x has no constant term) the prover can preserve the validity of equation (3); on the other hand, the message $g^{b^*}h^{\gamma}$ would completely invalidate the recipient's ciphertext (and the sender's).

Attack (Zero-knowledge)**.** Informally, the term $A_t = \left(D^{z^2} \cdot C_{Rn}^{z^3}\right)^{k_{\mathsf{sk}}}$ allows the verifier to decrypt the combined ciphertext $(C, D)^{z^2} \cdot (C_{Ln}, C_{Rn})^{z^3}$. Indeed, after honest behavior by the prover, the term $A_t \cdot {c_{\mathsf{commit}}}^{-1}$ equals $g^{c\cdot(b^*\cdot z^2 + b'\cdot z^3)}$; from this quantity, the verifier may brute-force the values b^* and b' using only 2^{64} work (and much less, if the verifier can "guess" these values sooner).

6.2.2 A refined "Σ-Bullets"

We propose a modified version of "Σ-Bullets" which addresses both of these issues. In fact, this is a *generic* approach for applying Bulletproofs to El Gamal-encrypted integers (in the exponent). For notational ease, we specialize to the case of basic Zether.

We informally describe our amended version; our detailed protocol (which also includes anonymity) can be found in below. Alongside the initial commitments $\mathbf{A}$ and $\mathbf{S}$, the prover should publish additional ciphertexts (C', D') and (C'_{Ln}, C'_{Rn}) which encrypt h^{γ^*} and $h^{\gamma'}$ respectively (for randomly chosen scalars γ^* and γ'). Upon receiving x, the prover should add to τ_x the constant term $z^2 \cdot \gamma^* + z^3 \cdot \gamma'$. During the sigma protocols, in addition to proving knowledge of sk for which $g^{\mathsf{sk}} = y$, the prover should *also* prove knowledge of b^*, b', γ^*, and γ' for which:

$$g^{b^*} = D^{-\mathsf{sk}} \cdot C, \quad g^{b'} = C_{Rn}^{-\mathsf{sk}} \cdot C_{Ln}, \quad h^{\gamma^*} = D'^{-\mathsf{sk}} \cdot C', \quad h^{\gamma'} = C_{Rn}'^{-\mathsf{sk}} \cdot C'_{Ln}.$$

Finally, the prover instead defines $A_t = \left((D \cdot D')^{z^2} \cdot (C_{Rn} \cdot C'_{Rn})^{z^3}\right)^{k_{\mathsf{sk}}}$, while in the check (3) the verifier instead uses $c_{\mathsf{commit}} = \left((D \cdot D')^{z^2} \cdot (C_{Rn} \cdot C'_{Rn})^{z^3}\right)^{s_{\mathsf{sk}}} \cdot \left((C \cdot C')^{z^2} \cdot (C_{Ln} \cdot C'_{Ln})^{z^3}\right)^{-c}$.

The additional Σ-proofs mitigate the soundness attack: they ensure that the messages of (C, D) and (C_{Ln}, C_{Rn}) only have "g parts" (and that (C', D') and that those of (C'_{Ln}, C'_{Rn}) only have "h parts"). The ciphertexts (C', D') and (C'_{Ln}, C'_{Rn}) themselves serve to blind the messages of (C, D) and (C_{Ln}, C_{Rn}). Indeed, the verifier may only decrypt a linear combination of all four ciphertexts; we note that $A_t \cdot {c_{\mathsf{commit}}}^{-1} = \left(g^{b^*}h^{\gamma^*}\right)^{c\cdot z^2} \cdot \left(g^{b'}h^{\gamma'}\right)^{c\cdot z^3}$. This decryption reveals nothing about b^* and b', and can be "fed into" the standard Bulletproofs protocol.

6.3 Anonymous payment

We now discuss our approach to anonymity, which uses the *many-out-of-many* proofs of Fig. 4 in a crucial way. Indeed, the most significant challenge of the Anonymous Zether relation (2) is that *all* N ciphertexts $(C_i, D)_{i=0}^{N-1}$ appear; in particular, it requires not just that $(y_{\mathsf{idx}_0} \cdot y_{\mathsf{idx}_1})^r = C_{\mathsf{idx}_0} \cdot C_{\mathsf{idx}_1}$, but also that $\bigwedge_{i \notin \{\mathsf{idx}_0, \mathsf{idx}_1\}} y_i^r = C_i$, where $g^r = D$ (and idx_0 and idx_1 are the sender's and receiver's secret indices, respectively).

We describe how many-out-of-many proofs can be applied to address this problem. The basic idea is that we run many-out-of-many proofs *twice*—with secrets idx_0 and idx_1, respectively—and using, in each case, the permutation $\kappa = (0, 2, \ldots, N-2)(1, 3, \ldots, N-1)$ (we require that N be even). This procedure has the effect of iterating independently over the orbits of idx_0 and idx_1 (respectively) under κ. These orbits, however, aren't necessarily disjoint; indeed, they're either disjoint or identical, accordingly as idx_0 and idx_1's parities are opposite or equal (respectively). We therefore require in addition that the two executions be run in such a way that the respective secrets idx_0 and idx_1 feature opposite parities (we can ensure this by adapting ideas already present in [GK15] and [BCC+15]).

In addition, instead of using individual matrices M for each execution, we interleave the respective rows yielded by the two executions. We obtain in this way a "double circulant" matrix, represented by the following schematic:

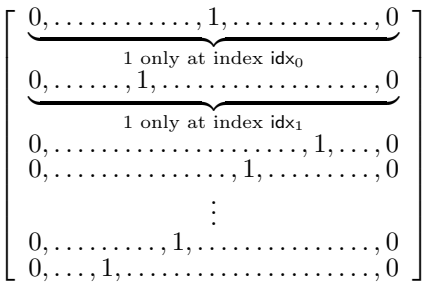

Figure 6: "Prover's view".

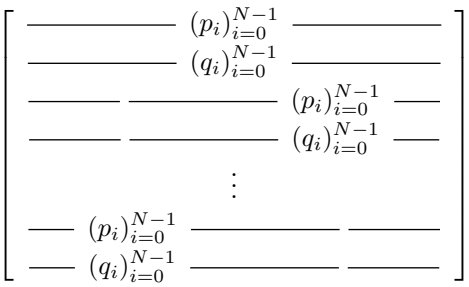

Figure 7: "Verifier's view".

Essentially, the prover implicitly sends the top *two* rows of an unknown matrix; by performing *two*-step rotations, the verifier constructs the remaining $N-2$ rows. We remark that the ultimate matrix is a permutation matrix if and only if the top two rows attain the value 1 at indices of opposite parity.

We can also express the prover's choice of the secrets idx_0 and idx_1 as that of a certain permutation. Indeed, any indices idx_0 and idx_1 with opposite parities implicitly yield in this way a permutation $K \in \mathbf{S}_N$, defined by setting, for any $i \in \{0, \ldots, N-1\}$:

$$i \mapsto K(i) := \begin{cases} (\mathsf{idx}_0 + 2 \cdot k) \mod N & \text{if } i = 2 \cdot k \\ (\mathsf{idx}_1 + 2 \cdot k) \mod N & \text{if } i = 2 \cdot k + 1. \end{cases}$$

Such permutations K are exactly those residing in the subgroup of $\mathbf{S}_N$ described by the generators $\langle (0, 1, 2, \ldots, N-1), (0, 2, 4, \ldots, N-2) \rangle$. In fact, these generators are "tight" in the sense that the natural map $\langle (0, 1, 2, \ldots, N-1) \rangle \times \langle (0, 2, 4, \ldots, N-2) \rangle \to \mathbf{S}_N$ is injective.

After interleaving the two executions as described above, we finally set M as the $(N-1) \times N$ matrix

$$M = \begin{bmatrix} 1 & 1 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & 1 \end{bmatrix}.$$

This matrix controls the messages of the permuted ciphertexts $(C_{K(0)}, D), (C_{K(1)}, D) \ldots, (C_{K(N-1)}, D)$. Indeed, it encodes exactly the fact that the sum $(C_{\mathsf{idx}_0}, D) \cdot (C_{\mathsf{idx}_1}, D)$ is an encryption of 0, whereas the ciphertexts (C_i, D) for $i \notin \{\mathsf{idx}_0, \mathsf{idx}_1\}$ individually encrypt 0. (We observe that this matrix has $O(N)$ nonzero entries, and so satisfies the requirements of the efficiency analysis above.)

6.4 The opposite parity requirement

We comment further on the requirement that $\mathsf{idx}_0 \not\equiv \mathsf{idx}_1 \mod 2$ (whose necessity is explained by the discussion above). This requirement decreases privacy, but only minimally so. Indeed, it decreases the cardinality of the set of possible pairs $(\mathsf{idx}_0, \mathsf{idx}_1) \in \{0, \ldots, N-1\}^2$ from $N \cdot (N-1)$ to $\frac{N^2}{2}$; put differently, it restricts the set of possible sender–receiver pairs to those represented by the edges of the *complete bipartite* directed graph on $\{0, \ldots, N-1\}$, where the coloring is given by parity (i.e., as opposed to the complete directed graph).

This restricted cardinally still grows quadratically in N, and in mainnet applications the deficit can essentially be remedied simply by picking larger anonymity sets. This recourse is not available in consortium settings, however, where the total size of the network is limited. We consider this to to be an acceptable limitation for practical use.

We briefly mention, though do not further pursue, an avenue by the aid of which the opposite parity requirement could be eliminated. Essentially, the prover and verifier may conduct many-out-of-many proofs only once, using the "canonical" permutation $\kappa = (0, 1, \ldots, N-1)$, and using a single secret idx_0 representing the sender's index. Moreover, they may choose for M the simple "summation" matrix

$$M = \begin{bmatrix} 1 & 1 & \ldots & 1 \end{bmatrix},$$

ensuring thereby that the (respective) quantities encrypted by $(C_i, D)_{i=0}^{N-1}$ sum to 0. The non-trivial use of many-out-of-many proofs would arise in the protocol's *range checks*. Whereas standard Anonymous Zether checks the ranges only of $(C_{Ln,\mathsf{idx}_0}, C_{Rn,\mathsf{idx}_0})$ and (C_{idx_0}, D), this modified approach would check the ranges of $(C_{Ln,\mathsf{idx}_0}, C_{Rn,\mathsf{idx}_0})$ and *all other* ciphertexts $(C_i, D)_{i \neq \mathsf{idx}_0}$. (We assume now a sign representation whereby (C_{idx_0}, D) encrypts a negative amount and all other ciphertexts encrypt positive amounts.) Effectively, the many-out-of-many proof would guide the verifier as to which ciphertext to *exempt* from range checking (while concealing its index in the original list). This approach has the additional benefit of allowing more general sorts of transactions, in which a single sender "spreads funds" to many receivers.

The drawback of this approach is that it would require running Bulletproofs on N ranges, as opposed to just two. This in turn would require (in addition to increased computation) a linearly-sized common reference string containing $n \cdot N$ (in practice, $32 \cdot N$) curve points; this would be cumbersome, especially in mainnet settings (where contract storage is expensive).

6.5 Full protocol

We now specify in detail our protocol for Anonymous Zether. We denote by n that integer for which $\mathsf{MAX} = 2^n - 1$ and by m that integer for which $N = 2^m$. All vector indices are understood to be taken modulo the size of the vector (in case of overflow). We define and make use of the following functions:

- $\mathsf{Shift}(\mathbf{v}, i)$ circularly shifts the vector $\mathbf{v}$ of field elements (i.e., $\mathbb{F}_q$) by the integer i.
- $\mathsf{MultiExp}(\mathbf{V}, \mathbf{v})$ multi-exponentiates the vector $\mathbf{V}$ of curve points by the vector $\mathbf{v}$ of field elements.

We mark in blue font those steps which do not appear in [BAZB], [BBB+18], [GK15] or [BCC+15].

Protocol Anonymous Zether

1: $\mathcal{P}$ **computes...**
2: $\alpha, \rho \leftarrow_\$ \mathbb{Z}_q$ ▷ Begin Bulletproof [BBB+18, §4]
3: $\mathbf{a}_L \in \{0,1\}^{2 \cdot n}$ s.t. $\langle \mathbf{a}_{L[:n]}, \mathbf{2}^n \rangle = b^*, \langle \mathbf{a}_{L[n:]}, \mathbf{2}^n \rangle = b'$
4: $\mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^{2 \cdot n}$

$\mathbf{A} = h^{\alpha}\mathbf{g}^{\mathbf{a}_L}\mathbf{h}^{\mathbf{a}_R}$

$\mathbf{s}_L, \mathbf{s}_R \leftarrow_{\$} \mathbb{Z}_q^{2\cdot n}$

$\mathbf{S} = h^{\rho}\mathbf{g}^{\mathbf{s}_L}\mathbf{h}^{\mathbf{s}_R}$

$r_A, r_B, r_C, r_D, r_E, r_F \leftarrow_{\$} \mathbb{Z}_q$ ▷ Begin one-out-of-many proof [BCC+15]

for all $j \in \{0,1\}$, $k \in \{0,\ldots,m-1\}$ **do**

sample $a_{j,k} \leftarrow_{\$} \mathbb{Z}_q$

set $b_{j,k} = (\mathsf{idx}_j)_k$, i.e., the k^{th} (little-endian) bit of idx_j

end for

$A = \mathrm{Com}(a_{0,0},\ldots,a_{1,m-1}; r_A)$

$B = \mathrm{Com}(q_{0,0},\ldots,q_{1,m-1}; r_B)$

$C = \mathrm{Com}((a_{j,k}(1-2b_{j,k}))_{j,k=0}^{1,m-1}; r_C)$

$D = \mathrm{Com}(-a_{0,0}^2,\ldots,-a_{0,m-1}^2; r_D)$

$E = \mathrm{Com}\left((a_{0,0}\cdot a_{1,0}, a_{0,0}\cdot a_{1,0}); r_E\right)$

$F = \mathrm{Com}\left((a_{b_{0,0},0}, -a_{b_{1,0},0}); r_F\right)$; the bits $b_{0,0}$ and $b_{1,0}$ are used as indices here.

end $\mathcal{P}$

$\mathcal{P} \to \mathcal{V} : \mathbf{A}, \mathbf{S}, A, B, C, D, E, F$

$\mathcal{V} : d \leftarrow_{\$} \mathbb{Z}_q$

$\mathcal{V} \to \mathcal{P} : d$

$\mathcal{P}$ **sets...** ▷ in what follows, we denote by i_k the k^{th} bit of i.

for all $j \in \{0,1\}$ **do**

for all $k \in \{0,\ldots,m-1\}$ **do**

set $F_{j,k,1}(W) := b_{j,k}\cdot W + a_{j,k}$

set $F_{j,k,0}(W) := W - F_{j,k,1}(W)$

end for

for all $i \in \{0,\ldots,N-1\}$ **do**

set $R_{j,i}(W) := \sum_{k=0}^{m} R_{j,i,k}\cdot W^k := \prod_{k=0}^{m-1} F_{j,k,i_k}(W)$

end for

end for

$(\pi_k, \chi_k, \sigma_k, \omega_k)_{k=0}^{m-1} \leftarrow_{\$} \mathbb{Z}_q$

for all $k \in \{0,\ldots,m-1\}$ **do**

$\widetilde{C_{Ln,k}} = \mathsf{MultiExp}\left((C_{Ln,i})_{i=0}^{N-1}, (R_{0,i,k})_{i=0}^{N-1}\right)\cdot(y_{\mathsf{idx}_0})^{\pi_k}$

$\widetilde{C_{Rn,k}} = \mathsf{MultiExp}\left((C_{Rn,i})_{i=0}^{N-1}, (R_{0,i,k})_{i=0}^{N-1}\right)\cdot g^{\pi_k}$

$\widetilde{C_{0,k}} = \mathsf{MultiExp}\left((C_i)_{i=0}^{N-1}, (R_{0,i,k})_{i=0}^{N-1}\right)(y_{\mathsf{idx}_0})^{\chi_k}$

$\widetilde{D_k} = g^{\chi_k}$

$\widetilde{y_{0,k}} = \mathsf{MultiExp}\left((y_i)_{i=0}^{N-1}, (R_{0,i,k})_{i=0}^{N-1}\right)(y_{\mathsf{idx}_0})^{\sigma_k}$

$\widetilde{g_k} = g^{\sigma_k}$

set $\mathbf{d} = (\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_{N-1}) = \left(1, 1, d, d^2, \ldots, d^{N-2}\right)$

$\widetilde{C_{X,k}} = \left(\prod_{j,i=0}^{1,\frac{N}{2}-1} g^{b^*\cdot(R_{j,\mathsf{idx}_0-2\cdot i,k} - R_{j,\mathsf{idx}_1-2\cdot i,k})}\right)^{\mathbf{d}_{2\cdot i+j}}\cdot D^{\omega_k}$

$\widetilde{y_{X,k}} = g^{\omega_k}$

end for

end $\mathcal{P}$

$\mathcal{P} \to \mathcal{V} : \left(\widetilde{C_{Ln,k}}, \widetilde{C_{Rn,k}}, \widetilde{C_{0,k}}, \widetilde{D_k}, \widetilde{y_{0,k}}, \widetilde{g_k}, \widetilde{C_{X,k}}, \widetilde{y_{X,k}}\right)_{k=0}^{m-1}$

$\mathcal{V} : w \leftarrow_{\$} \mathbb{Z}_q$

$\mathcal{V} \to \mathcal{P} : w$

$\mathcal{P}$ **sets...**

for all $j \in \{0,1\}$, $k \in \{0,\ldots,m-1\}$ **do**

set $f_{j,k} := F_{j,k,1}(w)$

end for

$z_A = r_B \cdot w + r_A$

$z_C = r_C \cdot w + r_D$

$z_E = r_F \cdot w + r_E$

$\overline{C_{Rn}} = (C_{Rn,\mathsf{idx}_0})^{w^m} \cdot \left(\prod_{k=0}^{m-1} g^{-\pi_k \cdot w^k}\right)$ ▷ Prover "anticipates" certain re-encryptions

$\overline{D} = D^{w^m} \cdot g^{-\sum_{k=0}^{m-1} \chi_k \cdot w^k}$

$\overline{y_0} = (y_{\mathsf{idx}_0})^{w^m} \cdot \left(\prod_{k=0}^{m-1} y_{\mathsf{idx}_0}^{-\sigma_k \cdot w^k}\right)$

$\overline{g} = g^{w^m - \sum_{k=0}^{m-1} \sigma_k \cdot w^k}$

$\overline{y_{X,k}} = \prod_{j,i=0}^{1,\frac{N}{2}-1} \mathsf{MultiExp}\left((y_i)_{i=0}^{N-1}, \mathsf{Shift}\left((R_{j,i}(w))_{i=0}^{N-1}, 2 \cdot i\right)\right)^{\mathbf{d}_{2 \cdot i + j}} \cdot \left(\prod_{k=0}^{m-1} g^{-\omega_k \cdot w^k}\right)$

$\gamma^*, \gamma', \zeta^*, \zeta' \leftarrow_\$ \mathbb{Z}_q$ ▷ Begin computation of blinding ciphertexts

$(C', D') = \left(h^{w^m \cdot \gamma^*} \cdot \overline{y_0}^{\zeta^*}, \overline{g}^{\zeta^*}\right)$

$(C'_{Ln}, C'_{Rn}) = \left(h^{w^m \cdot \gamma'} \cdot \overline{y_0}^{\zeta'}, \overline{g}^{\zeta'}\right)$

end $\mathcal{P}$

$\mathcal{P} \rightarrow \mathcal{V} : (f_{j,k})_{j,k=0}^{1,m-1}, z_A, z_C, z_E, C', D', C'_{Ln}, C'_{Rn}$

$\mathcal{V} : y, z \leftarrow_\$ \mathbb{Z}_q$

$\mathcal{V} \rightarrow \mathcal{P} : y, z$

$\mathcal{P}$ **:**

$l(X) = (\mathbf{a}_L - z \cdot \mathbf{1}^n) + \mathbf{s}_L \cdot X$

$r(X) = \mathbf{y}^n \circ (\mathbf{a}_R + z \cdot \mathbf{1}^n + \mathbf{s}_R \cdot X) + z^2 \cdot (\mathbf{2}^n \parallel \mathbf{0}^n) + z^3 \cdot (\mathbf{0}^n \parallel \mathbf{2}^n)$

$t(X) = \langle l(X), r(X) \rangle = t_0 + t_1 \cdot X + t_2 \cdot X^2$ ▷ l and r are elements of $\mathbb{Z}_q^{2 \cdot n}[X]$; $t \in \mathbb{Z}_q[X]$

$\tau_1, \tau_2 \leftarrow_\$ \mathbb{Z}_q$

$T_i = g^{t_i} h^{\tau_i}$ **for** $i \in \{1, 2\}$

end $\mathcal{P}$

$\mathcal{P} \rightarrow \mathcal{V} : T_1, T_2$

$\mathcal{V} : x \leftarrow_\$ \mathbb{Z}_q$

$\mathcal{V} \rightarrow \mathcal{P} : x$

$\mathcal{P}$ **sets...**

$\mathbf{l} = l(x) = \mathbf{a}_L - z \cdot \mathbf{1}^{2 \cdot n} + \mathbf{s}_L \cdot x$

$\mathbf{r} = r(x) = \mathbf{y}^{2 \cdot n} \circ (\mathbf{a}_R + z \cdot \mathbf{1}^{2 \cdot n} + \mathbf{s}_R \cdot x) + z^2 \cdot (\mathbf{2}^n \parallel \mathbf{0}^n) + z^3 \cdot (\mathbf{0}^n \parallel \mathbf{2}^n)$

$\hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle$ ▷ $\mathbf{l}$ and $\mathbf{r}$ are elements of $\mathbb{Z}_q^{2 \cdot n}$; $\hat{t} \in \mathbb{Z}_q$

$\tau_x = \tau_2 \cdot x^2 + \tau_1 \cdot x + z^2 \cdot \gamma^* + z^3 \cdot \gamma'$

$\mu = \alpha + \rho \cdot x$

$A_y = \overline{g}^{k_{\mathsf{sk}}}$ ▷ Begin sigma protocol proving

$A_D = g^{k_r}$

$A_u = g_{\mathsf{epoch}}^{k_{\mathsf{sk}}}$

$A_X = \overline{y_X}^{k_r}$

$A_t = \left((\overline{D} \cdot D')^{z^2} \cdot (\overline{C_{Rn}} \cdot C'_{Rn})^{z^3}\right)^{k_{\mathsf{sk}}}$

$A_{\overline{C_0}} = g^{k_{v^*}} \cdot \overline{D}^{k_{\mathsf{sk}}}$

$A_{\overline{C_{Ln}}} = g^{k_{v'}} \cdot \overline{C_{Rn}}^{-k_{\mathsf{sk}}}$

$A_{C'} = h^{k_{\nu^*}} \cdot D'^{k_{\mathsf{sk}}}$

$A_{C'_{Ln}} = h^{k_{\nu'}} \cdot C'^{k_{\mathsf{sk}}}_{Rn}$

end $\mathcal{P}$

$\mathcal{P} \rightarrow \mathcal{V} : \hat{t}, \tau_x, \mu, A_y, A_D, A_u, A_X, A_t, A_{\overline{C_0}}, A_{\overline{C_{Ln}}}, A_{C'}, A_{C'_{Ln}}$

$\mathcal{V} : c \leftarrow_\$ \mathbb{Z}_q$

$\mathcal{V} \rightarrow \mathcal{P} : c$

$\mathcal{P}$ **sets...**

$s_{\mathsf{sk}} = k_{\mathsf{sk}} + c \cdot \mathsf{sk}$

$s_r = k_r + c \cdot r$

$s_{v^*} = k_{v^*} + c \cdot w^m \cdot b^*$

$s_{v'} = k_{v'} + c \cdot w^m \cdot b'$

$s_{\nu^*} = k_{\nu^*} + c \cdot w^m \cdot \gamma^*$

$s_{\nu'} = k_{\nu'} + c \cdot w^m \cdot \gamma'$

end $\mathcal{P}$

$\mathcal{P} \to \mathcal{V} : s_{\mathsf{sk}}, s_r, s_{v^*}, s_{v'}, s_{\nu^*}, s_{\nu'}$

$\mathcal{V}$ **:**

  for all $j \in \{0,1\}$, $k \in \{0, \ldots, m-1\}$ **do**

    set $f_{j,k,1} = f_{j,k}$

    set $f_{j,k,0} = w - f_{j,k}$

    for all $i \in \{0, \ldots, N-1\}$ **do**

      set $r_{j,i} = \prod_{k=0}^{m-1} f_{j,k,k_i}$

    end for

  end for

  $B^w A \stackrel{?}{=} \mathrm{Com}\left(f_{0,0}, \ldots, f_{1,m-1}; z_A\right)$

  $C^w D \stackrel{?}{=} \mathrm{Com}\left((f_{j,k}(w - f_{j,k}))_{j,k=0}^{1,m-1}; z_C\right)$

  $F^w E \stackrel{?}{=} \mathrm{Com}\left((f_{0,0} \cdot f_{1,0}, (w - f_{0,0})(w - f_{1,0})); z_E\right)$ ▷ Opposite parity check

  $\overline{C_{Ln}} = \mathsf{MultiExp}\left((C_{Ln,i})_{i=0}^{N-1}, (r_{0,i})_{i=0}^{N-1}\right) \cdot \prod_{k=0}^{m-1} \widetilde{C_{Ln,k}}^{-w^k}$

  $\overline{C_{Rn}} = \mathsf{MultiExp}\left((C_{Rn,i})_{i=0}^{N-1}, (r_{0,i})_{i=0}^{N-1}\right) \cdot \prod_{k=0}^{m-1} \widetilde{C_{Rn,k}}^{-w^k}$

  $\overline{C_0} = \mathsf{MultiExp}\left((C_i)_{i=0}^{N-1}, (r_{0,i})_{i=0}^{N-1}\right) \cdot \prod_{k=0}^{m-1} \widetilde{C_{0,k}}^{-w^k}$

  $\overline{D} = D^{w^m} \cdot \prod_{k=0}^{m-1} \widetilde{D_k}^{-w^k}$

  $\overline{y_0} = \mathsf{MultiExp}\left((y_i)_{i=0}^{N-1}, (r_{0,i}(w))_{i=0}^{N-1}\right) \cdot \prod_{k=0}^{m-1} \widetilde{y_{0,k}}^{-w^k}$

  $\overline{g} = g^{w^m} \cdot \prod_{k=0}^{m-1} \widetilde{g_k}^{-w^k}$

  set $\mathbf{d} = (\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_{N-1}) = \left(1, 1, d, d^2, \ldots, d^{N-2}\right)$

  $\overline{C_X} = \prod_{j,i=0}^{1,\frac{N}{2}-1} \mathsf{MultiExp}\left((C_i)_{i=0}^{N-1}, \mathsf{Shift}\left((r_{j,i})_{i=0}^{N-1}, 2 \cdot i\right)\right)^{\mathbf{d}_{2 \cdot i + j}} \cdot \prod_{k=0}^{m-1} \widetilde{C_{X,k}}^{-w^k}$

  $\overline{y_X} = \prod_{i,j=0}^{1,\frac{N}{2}-1} \mathsf{MultiExp}\left((y_i)_{i=0}^{N-1}, \mathsf{Shift}\left((r_{j,i})_{i=0}^{N-1}, 2 \cdot i\right)\right)^{\mathbf{d}_{2 \cdot i + j}} \cdot \prod_{k=0}^{m-1} \widetilde{y_{X,k}}^{-w^k}$

  $A_y \stackrel{?}{=} \overline{g}^{s_{\mathsf{sk}}} \cdot (\overline{y_0})^{-c}$ ▷ Begin sigma protocol verification

  $A_D \stackrel{?}{=} g^{s_r} \cdot D^{-c}$

  $A_u \stackrel{?}{=} g_{\mathsf{epoch}}^{s_{\mathsf{sk}}} \cdot u^{-c}$

  $A_X \stackrel{?}{=} \overline{y_X}^{s_r} \cdot \overline{C_X}^{-c}$

  $\delta(y, z) = (z - z^2) \cdot \left\langle \mathbf{1}^{2 \cdot n}, \mathbf{y}^{2 \cdot n} \right\rangle - \left(z^3 \cdot \langle \mathbf{1}^n, \mathbf{2}^n \rangle + z^4 \cdot \langle \mathbf{1}^n, \mathbf{2}^n \rangle\right)$

  $c_{\mathsf{commit}} = \left((\overline{D} \cdot D')^{z^2} \cdot (\overline{C_{Rn}} \cdot C'_{Rn})^{z^3}\right)^{s_{\mathsf{sk}}} \cdot \left((\overline{C_0} \cdot C')^{z^2} \cdot (\overline{C_{Ln}} \cdot C'_{Ln})^{z^3}\right)^{-c}$

  $g^{w^m \cdot c \cdot \hat{t}} \cdot h^{w^m \cdot c \cdot \tau_x} \stackrel{?}{=} g^{w^m \cdot c \cdot \delta(y,z)} \cdot A_t \cdot {c_{\mathsf{commit}}}^{-1} \cdot \left(T_1^x \cdot T_2^{x^2}\right)^{w^m \cdot c}$

  $A_{\overline{C_0}} \stackrel{?}{=} g^{s_{v^*}} \cdot \overline{D}^{s_{\mathsf{sk}}} \cdot \overline{C_0}^{-c}$

  $A_{\overline{C_{Ln}}} \stackrel{?}{=} g^{s_{v'}} \cdot \overline{C_{Rn}}^{s_{\mathsf{sk}}} \cdot \overline{C_{Ln}}^{-c}$

  $A_{C'} \stackrel{?}{=} h^{s_{\nu^*}} \cdot D'^{s_{\mathsf{sk}}} \cdot C'^{-c}$

  $A_{C'_{Ln}} \stackrel{?}{=} h^{s_{\nu'}} \cdot C'^{s_{\mathsf{sk}}}_{Rn} \cdot C'^{-c}_{Ln}$

end $\mathcal{V}$

$\mathbf{h}' = \left(h_1, h_2^{(y^{-1})}, h_3^{(y^{-2})}, \ldots, h_{2 \cdot n}^{(y^{-2 \cdot n + 1})}\right)$ ▷ Complete inner product argument

$Z = \mathbf{A} \cdot \mathbf{S}^x \cdot \mathbf{g}^{-z} \cdot \mathbf{h}'^{z \cdot \mathbf{y}^{2 \cdot n}} \cdot \mathbf{h}'^{z^2 \cdot (\mathbf{2}^n \| \mathbf{0}^n) + z^3 \cdot (\mathbf{0}^n \| \mathbf{2}^n)}$

$\mathcal{P}$ and $\mathcal{V}$ engage in Protocol 1 of [BBB+18] on inputs $(\mathbf{g}, \mathbf{h}', Zh^{-\mu}, \hat{t}; \mathbf{l}, \mathbf{r})$

6.6 Performance

We describe our implementation of Anonymous Zether. Verification takes place in Solidity contracts. Proving takes place in a JavaScript library, which is in turn invoked by our front-end (also written in JavaScript).

We report performance measurements below. We note that *gas used* includes not just verification itself, but also the relevant account maintenance associated with the Zether Smart Contract; our gas measurements *do incorporate EIP-1108*. The *verification time* we report reflects only the time taken by the local EVM in evaluating a read-only call to the verification contract. Proving time is self-explanatory. Each number next to *Transfer* indicates the size of the anonymity set used (including the actual sender and recipient). Our "Burn" transaction is actually a *partial burn*, in contrast to that of [BAZB]; in other words, we allow a user to withdraw only part of her balance (using a single range proof).

	Prov. Time (ms)	Verif. Time (ms)	Prf. Size (bytes)	Gas Used
Burn	907	83	1,312	2,393,134
Transfer (2)	2,013	120	2,720	5,078,866
Transfer (4)	2,155	142	3,296	6,126,771
Transfer (8)	2,420	172	3,872	8,271,458
Transfer (16)	3,043	247	4,448	13,214,370
Transfer (32)	4,525	394	5,024	23,709,407
Transfer (64)	7,932	705	5,600	47,503,518
Transfer (N)	$O(N \log N)$	$O(N \log N)$	$O(\log N)$	$O(N \log N)$

References

[BAZB] Benedikt Bünz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. Zether: Towards privacy in a smart contract world. Unpublished manuscript.

[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 (SP)*, volume 1, 2018. Full version.

[BCC+15] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, Jens Groth, and Christophe Petit. Short accountable ring signatures based on DDH. In Günther Pernul, Peter Y A Ryan, and Edgar Weippl, editors, *Computer Security – ESORICS 2015*, volume 9326 of *Lecture Notes in Computer Science*, pages 243–265. Springer International Publishing, 2015.

[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, *Advances in Cryptology – EUROCRYPT 2016*, volume 9666 of *Lecture Notes in Computer Science*, pages 327–357. Springer Berlin Heidelberg, 2016.

[BKM09] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. *Journal of Cryptology*, 22:114–138, 2009.

[Coh74] P.M. Cohn. *Algebra*, volume 1. John Wiley & Sons, 1974.

[ESS+19] Muhammed F. Esgin, Ron Steinfeld, Amin Sakzad, Joseph K. Liu, and Dongxi Liu. Short lattice-based one-out-of-many proofs and applications to ring signatures. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *Applied Cryptography and Network Security*, pages 67–88. Springer International Publishing, 2019.

[FMMO] Prastudy Fauzi, Sarah Meiklejohn, Rebekah Mercer, and Claudio Orlandi. Quisquis: A new design for anonymous cryptocurrencies. Unpublished manuscript.

[GK15] Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, volume 9057 of *Lecture Notes in Computer Science*, pages 253–280. Springer Berlin Heidelberg, 2015.

[JW90] J. Jeong and W. J. Williams. A fast recursive bit-reversal algorithm. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages 1511–1514, 1990.

[KL15] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. CRC Press, second edition, 2015.

[Nus82] H. Nussbaumer. *Fast Fourier Transform and Convolution Algorithms*. Springer-Verlag, 1982.

[Pol71] J. M. Pollard. The fast Fourier transform in a finite field. *Mathematics of Computation*, 25(114):365–374, 1971.

[vzGG13] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, third edition, 2013.