MANY-OUT-OF-MANY PROOFS

with applications to *Anonymous Zether*

Benjamin E. Diamond*
J.P. Morgan
benediamond@gmail.com

Abstract

We introduce a family of extensions to the *one-out-of-many proofs* of Groth and Kohlweiss (Eurocrypt 2015), which efficiently prove statements about *many* messages among a list of commitments. These extensions prove knowledge of a secret *subset* of the list, and assert that the commitments in the subset satisfy certain properties (expressed as linear equations). Our communication remains logarithmic; our computation increases only by a logarithmic multiplicative factor. Our work introduces a new "circular rotation" technique, and a novel instantiation of the number-theoretic transform.

Applying these techniques, we construct a protocol for the *Anonymous Zether* payment system—as proposed in Bünz, Agrawal, Zamani, and Boneh (FC'20)—which improves upon the communication complexity attained by existing efforts. We describe an open-source, Ethereum-based implementation of our protocol.

1 Introduction

Blockchain-based cryptocurrencies like Bitcoin [Nak08] allow their mutually distrustful participants to maintain shared computational state. These systems generally encode this state—as well as the *transactions* which incrementally modify it—"in the clear", and so afford to these participants only cursory privacy (we refer to, e.g., Ron and Shamir [RS13]).

This deficiency has impelled the development of "privacy-preserving" alternatives, most notably Zcash [SCG$^+$14] and Monero [SMM16]. These systems encode their state cryptographically, and define transactions which *privately* and securely modify this state (frequently with recourse to non-interactive zero-knowledge proofs). Fauzi, Meiklejohn, Mercer, and Orlandi's *Quisquis* [FMMO19] materially advances this research line, in that it demands of each participating node disk space (per user of the system) which grows only *constantly* in time, and so dispenses with the "monotonically increasing" UTXO sets endemic to Zcash and Monero.

The *Anonymous Zether* paradigm, proposed by Bünz, Agrawal, Zamani, and Boneh [BAZB20], breaks ground in multiple further respects. For one, it withstands a certain "race condition" which threatens high-throughput deployments of Quisquis ([BAZB20] calls this "the front-running problem"; see also [FMMO19, §5.2.6]). Anonymous Zether additionally obviates the practice—required by Quisquis, Monero, and Zcash—whereby each user must continually "scan" through all posted transactions in order to identify relevant ones.

In this paper, we present a construction of Anonymous Zether which improves upon the efficiency attained by existing efforts. [BAZB20] claims a construction featuring linearly growing proofs; Quisquis's proofs also grow linearly (i.e., in the size N of each transaction's "anonymity set"). Our proofs are $O(\log N)$-sized. Our *concrete* proof and transaction sizes are also smaller than those of Quisquis, by an order of magnitude. We thus resolve the "interesting open question" posed by Fauzi, Meiklejohn, Mercer, and Orlandi [FMMO19, §9] (namely, "the design of a special-purpose NIZK for improved communication efficiency").

*I would like to thank Markulf Kohlweiss and Michele Ciampi for many helpful discussions and suggestions.

1.1 Overview of Anonymous Zether

The *Anonymous Zether* [BAZB20, §D] payment paradigm maintains a global table of "accounts", which associates to each public key an El Gamal ciphertext under that key (encrypting that key's account balance "in the exponent"). To send funds, a user selects a ring containing herself and the recipient, and encrypts, under the ring's respective keys, the amounts by which she intends to alter each account's balance. The administering environment (e.g., smart contract) applies these adjustments homomorphically.

Each transaction, finally, must attach a proof attesting that it preserves all monetary invariants. These invariants are expressed by the relation [BAZB20, (8)] (see also (2) below), which encodes, in particular, that value is conserved, and flows only *from* some account whose secret key the prover knows, as well as that no overdrafts occur.

1.2 Technical challenges

An efficient proof protocol for (2) presents a number of challenges. Importantly, it 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).

This very fact precludes an elementary application of Groth and Kohlweiss's *one-out-of-many proofs* [GK15], whose use [BAZB20] recommends. The tempting approach whereby the prover conducts [GK15] N times—"handing" to the verifier, in each execution, a distinct element of the list—would be inefficient (incurring super-linear communication and super-quadratic computation). More subtly, it would prove nothing about how the N secret indices relate to each other, and in particular whether they're *distinct.* Indeed, the prover must deliver something like a verifiable shuffle of the input ciphertexts (so that the verifier can perform checks on the shuffled ciphertexts).

Shuffle proofs too, however, fall short of our needs (we again leave aside for now their inefficiency). Indeed, the adjustment ciphertexts of the Anonymous Zether relation (2) are encrypted under the ring's members' *heterogeneous* public keys, as are the ciphertexts representing their post-adjustment balances. More subtly, shuffle proofs also deliver "more than we need". While they allow a prover to designate a *full* permutation of a list of ciphertexts, our prover need only distinguish *two* among them (namely, the sender's and receiver's); the verifier may complete the permutation arbitrarily. Our protocol fundamentally exploits this insight.

2 Overview of our contribution

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. As we have seen, however, certain applications require more flexible assertions (which, in particular, pertain to more than one element of the list). Informally, many-out-of-many proofs allow a prover to efficiently prove knowledge of a certain (ordered) *subset* of a fixed list of commitments, as well as that the elements of this subset satisfy certain properties.

We briefly sketch a representative example. Given some list $c_0, \ldots, c_{N-1}$ of commitments, and having agreed upon some pre-specified linear map $\Xi \colon \mathbb{F}_q^N \rightarrow \mathbb{F}_q^s$ (say), a prover might wish to demonstrate knowledge of a *secret* permutation $K \in \mathbf{S}_N$, as well as of openings to zero of the image points of $(c_{K(0)}, \ldots, c_{K(N-1)})$ under Ξ. 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 $\{0, 1, \ldots, N-1\}$ 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, $\kappa = (0, 2, \ldots, N-2)(1, 3, \ldots, N-1)$ (for N even, and for specially chosen Ξ, described below) is used in the crucial step of Anonymous Zether (details are given in Subsection 6.3). 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 Ξ (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. We thus have (see Section 4.2 below):

Theorem 2.1. *There exists a sound, honest verifier zero-knowledge protocol for the* many-out-of-many *relation $\mathcal{R}_2$ below, which requires $O(\log N)$ communication, and moreover can be implemented in $O(N \log^2 N)$ time for the prover and $O(N \log N)$ time for the verifier.*

2.1 Review of one-out-of-many proofs

The central technique of one-out-of-many proofs [GK15] (see also [BCC+15]) 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.

The utility of the vector $(p_i)_{i=0}^{N-1}$ resides in its use as the exponent in a multi-exponentiation. Indeed multi-exponentiating the public vector of commitments $(c_0, \ldots, c_{N-1})$ by $(p_i)_{i=0}^{N-1}$ "picks out", modulo lower-order terms, exactly that commitment c_i for which $P_i(X)$ has high degree (namely, c_l). More precisely, the prover sends (before seeing x) correction terms representing the lower-degree (i.e., strictly less than m) parts of the polynomials $P_i(X)$. Once x is released, these correction terms strip away the lower-order parts of the evaluations p_i (while adding blinding randomnesses), and leave the verifier with a re-encryption of c_l alone.

2.2 Idea of many-out-of-many proofs

Our first core idea is that, having reconstructed the vector $(p_i)_{i=0}^{N-1}$ of evaluations, the verifier may "homomorphically permute" this vector and *re-use* its components 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 additional communication).

We fix a permutation $\kappa \in \mathbf{S}_N$ in what follows. Given the vector $(p_i)_{i=0}^{N-1}$, the verifier may iteratively permute its components, and so 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 nevertheless 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* initial 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.

2.3 Correction terms and linear maps

Of course, $\prod_{i=0}^{N-1} c_i^{p_i}$ 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 elements (if $o = \Theta(N)$), 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 "random linear combination" of these terms; this recourse evokes that used (twice) in Bulletproofs [BBB$^+$18, §4.1]. For additional flexibility, we also interpose an arbitrary linear transformation $\Xi\colon \mathbb{F}_q^o \to \mathbb{F}_q^s$. The prover then sends correction terms *only* for the single element

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

where v is a random challenge chosen by the verifier (the right-hand dot is a "module product"). By interleaving v with the many-out-of-many process with appropriate delicacy, we ensure that the resulting protocol is still sound.

2.4 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 *circular shift* matrix to the verifier, who constructs the rest locally.

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

Figure 1: "Prover's view".

$$\begin{bmatrix} \underbrace{\rule{3cm}{0.4pt} \; (p_i)_{i=0}^{N-1} \; \rule{3cm}{0.4pt}}_{\text{“1” at unknown index}} \\ \rule{1cm}{0.4pt} \; \rule{3cm}{0.4pt} \; (p_i)_{i=0}^{N-1} \; \rule{2cm}{0.4pt} \\ \rule{2cm}{0.4pt} \; \rule{3cm}{0.4pt} \; (p_i)_{i=0}^{N-1} \; \rule{1cm}{0.4pt} \\ \vdots \\ \rule{1cm}{0.4pt} \; (p_i)_{i=0}^{N-1} \; \rule{3cm}{0.4pt} \; \rule{2cm}{0.4pt} \\ \rule{2cm}{0.4pt} \; (p_i)_{i=0}^{N-1} \; \rule{3cm}{0.4pt} \; \rule{1cm}{0.4pt} \end{bmatrix}$$

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 a *circulant* matrix, and this multiplication is a circular convolution; the number-theoretic transform can thus be applied (see Subsection 2.5 for additional discussion).

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 Ξ (as well as the "linear combination trick") can be homomorphically applied. (We use the identity $\kappa^j(l) = \kappa^l(j)$, true in particular for $\kappa = (0, 1, \ldots, N-1)$.) Supposing now, in addition, that the prover and verifier have agreed in advance upon a linear *functional* $\Xi\colon \mathbb{F}_q^N \to \mathbb{F}_q$, our general protocol (in this case) thus yields a proof of knowledge of a *secret* permutation $K \in \langle (0, 1, \ldots, N-1) \rangle$, as well as of an opening to zero of the image under Ξ of the *permuted* vector $(c_{K(0)}, \ldots, c_{K(N-1)})$. Heuristically, it asserts that the "messages" of $c_0, \ldots, c_{N-1}$ reside in some hyperplane of $\mathbb{F}_q^N$, after being rotated.

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.

2.5 Circular convolutions and the number-theoretic transform

We remark further upon our use of Fourier-theoretic techniques. General treatments of these ideas—such as that of Tolimieri, An and Lu [TAL97]—tend only to treat the convolution of vectors consisting of (complex) *field elements*. This remains true even for those surveys, like Nussbaumer's [Nus82, §8], which address also the prime field case (often called the "number theoretic transform").

We fix in what follows a commitment scheme whose commitment space is a q-torsion group (for a prime q), or in other words an $\mathbb{F}_q$-module (actually, a vector space). (The Pedersen and El Gamal commitment schemes satisfy this property.) Our setting, unusually, mandates that a vector of *module* elements (i.e., commitments) be convolved with a vector of field elements. Our important 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 fast Fourier transform and the convolution theorem, and that these techniques can be carried out "homomorphically". We thus introduce the efficient convolution of a vector of module elements with a vector of field elements (see also Remark 4.9 below). Though this observation is implicit in existing work, we have not found it stated explicitly in the literature.

2.6 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 general security definitions, deferring specialized definitions to the appropriate sections below.

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 3.1 (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 $x \in \mathbb{F}_q$.
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 for which $\Pr[\mathsf{DLog}_{\mathcal{A},\mathcal{G}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

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

Definition 3.2. 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 $x, y, z \in \mathbb{F}_p$ 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 for which $\Pr[\mathsf{DDH}_{\mathcal{A},\mathcal{G}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

3.2 Commitment schemes

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

We now present security definitions.

Definition 3.3 (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) and (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 $\mathsf{Com}(\mathsf{params}, m_0; r_0) = \mathsf{Com}(\mathsf{params}, m_1; r_1)$.

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

Definition 3.4 (Katz–Lindell [KL15, Def. 5.13])**.** The *commitment hiding experiment* $\mathsf{Hiding}_{\mathcal{A},\mathsf{com}}(\lambda)$ is defined as:

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

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

3.2.1 An alternate notion of hiding

A commitment scheme is *homomorphic* if, for each params, its message, randomness, and commitment spaces are abelian groups, and the corresponding commitment function is a group homomorphism.

We now present a slightly modified version of Definition 3.4. This definition makes sense only for homomorphic schemes; it shall also better suit our purposes below. In this version, the adversary outputs two challenge *commitments*, as opposed to messages; one among these is then re-randomized homomorphically by the experimenter.

Definition 3.5. The *modified hiding experiment* $\mathsf{MHiding}_{\mathcal{A},\mathsf{Com}}(\lambda)$ is defined as:

1. Parameters $\mathsf{params} \leftarrow \mathsf{Gen}(1^\lambda)$ are generated.
2. The adversary $\mathcal{A}$ is given input params, and outputs elements c_0 and c_1 of the commitment space.
3. A uniform bit $b \in \{0, 1\}$ is chosen. The commitment $\mathsf{com} := c_b \cdot \mathsf{Com}(0)$ is computed and given to $\mathcal{A}$.
4. The adversary $\mathcal{A}$ outputs a bit b'. The output of the experiment is 1 if and only if $b' = b$.

Any scheme which is hiding in the sense of Definition 3.5 is also hiding in the classical sense of Definition 3.4. Indeed, any adversary $\mathcal{A}$ targeting $\mathsf{Hiding}_{\mathcal{A},\mathsf{Com}}$ yields an adversary $\mathcal{A}'$ targeting $\mathsf{MHiding}_{\mathcal{A}',\mathsf{Com}}$ in the obvious way. Upon receiving $\mathcal{A}$'s messages m_0 and m_1, $\mathcal{A}'$ outputs $c_0 := \mathsf{Com}(m_0; 0)$ and $c_1 := \mathsf{Com}(m_1; 0)$. Finally, it passes the challenge com to $\mathcal{A}$, and returns whatever $\mathcal{A}$ returns.

On the other hand, the reverse implication is also true, as the following lemma argues:

Lemma 3.6. *Definitions 3.4 and 3.5 are equivalent for any homomorphic commitment scheme* Com.

Proof. It remains to convert any $\mathcal{A}$ attacking $\mathsf{MHiding}_{\mathcal{A},\mathsf{Com}}$ into an adversary $\mathcal{A}'$ attacking $\mathsf{Hiding}_{\mathcal{A}',\mathsf{Com}}$. $\mathcal{A}'$ operates as follows, on input params:

1. For a randomly chosen message r, assign $m_0 = r$ and $m_1 = 0$. Output m_0 and m_1.
2. Upon receiving the experimenter's challenge com and $\mathcal{A}$'s commitments c_0 and c_1, select a random bit $b \in \{0, 1\}$. Give $c_b \cdot \mathsf{com}$ to $\mathcal{A}$.
3. When $\mathcal{A}$ outputs a bit b', return whether $b' = b$.

If the experimenter's bit is 0, then its challenge com is completely random, as is hence $c_b \cdot \mathsf{com}$; we conclude in this case that $\mathcal{A}$'s advantage is 0. If on the other hand the experimenter's bit is 1, then $\mathcal{A}$'s view exactly matches its view in $\mathsf{MHiding}_{\mathcal{A},\mathsf{Com}}$, and in this case $\mathcal{A}'$ wins whenever $\mathcal{A}$ does. We conclude that:

$$\Pr[\mathsf{Hiding}_{\mathcal{A}',\mathsf{Com}}(\lambda) = 1] - \frac{1}{2} = \frac{1}{2} \cdot (0) + \frac{1}{2} \cdot \left(\Pr[\mathsf{MHiding}_{\mathcal{A},\mathsf{Com}}(\lambda) = 1] - \frac{1}{2}\right).$$

In particular, if Com is hiding, then $\Pr[\mathsf{MHiding}_{\mathcal{A},\mathsf{Com}}(\lambda) = 1] - \frac{1}{2}$ is negligible. $\square$

Example 3.7. If a homomorphic commitment scheme is *perfectly* hiding in the classical sense of Definition 3.4, then it's also *perfectly* hiding in the modified sense, as the proof of Lemma 3.6 shows. For example, we have the Pedersen commitment scheme (as in e.g. [BCC+16, §2.2]).

Example 3.8. Specializing Definition 3.5 to the El Gamal encryption scheme (as in e.g. [KL15, Cons. 11.16])—which we view as a commitment scheme—we obtain the following unusual experiment:

1. Parameters $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^\lambda)$, as well as a random keypair $(y, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$, are generated.
2. $\mathcal{A}$ is given $(\mathbb{G}, q, g)$ and y. $\mathcal{A}$ outputs group-element tuples $c_0 = (M_0, m_0)$ and $c_1 = (M_1, m_1)$.
3. A uniform bit $b \in \{0, 1\}$ is chosen. A random element $r \leftarrow \mathbb{F}_q$ is generated, and $(M_b \cdot y^r, m_b \cdot g^r)$ is returned to $\mathcal{A}$.
4. $\mathcal{A}$ outputs a bit b'. The output of the experiment is defined to be 1 if and only if $b' = b$.

In virtue of [KL15, Thm. 11.18] and of Lemma 3.6, we conclude that, under the DDH assumption, each adversary $\mathcal{A}$ has at most negligible advantage in this experiment.

We assume in what follows that all commitment schemes are homomorphic. We also assume that each commitment scheme has randomness space given by $\mathbb{F}_q$, for a λ-bit prime q, as well a q-torsion group for its commitment space.

3.3 Zero-knowledge proofs

We present definitions for zero-knowledge arguments of knowledge, closely following [GK15] and [BCC+16]. We formulate our 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 as $\langle \mathcal{P}(s), \mathcal{V}(t) \rangle = b$.

We now have:

Definition 3.9. 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) for which $(\sigma, u, w) \in \mathcal{R}$
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 $b = 1$.

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 3.10. 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 whose challenges feature no collisions.
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*.

Definition 3.11. 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) for which $(\sigma, u, w) \in \mathcal{R}$, as well as randomness 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 $b' = b$.

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*.

In all of our 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 Many-out-of-Many Proofs

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

Definition 4.1. 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 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 thus also have the notion, for each $l \in \{0, \ldots, N-1\}$, of l's "ordered orbit", given by 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.

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:

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

and the protocol:

$\underline{\mathcal{P}_1(\sigma, B; (b_0, \ldots, b_{m-1}), r_B)}$		$\underline{\mathcal{V}_1(\sigma, B)}$
$r_A, r_C, r_D, a_0, \ldots, a_{m-1} \leftarrow\$\, \mathbb{F}_q$		
$A := \mathsf{Com}\left(a_0, \ldots, a_{m-1}; r_A\right)$		
$C := \mathsf{Com}\left((a_k \cdot (1 - 2b_k))_{k=0}^{m-1}; r_C\right)$		
$D := \mathsf{Com}\left(-a_0^2, \ldots, a_{m-1}^2; r_D\right)$	$\xrightarrow{A, C, D}$	
	$\xleftarrow{x}$	$x \leftarrow\$\, \mathbb{F}_q$
$\forall i : f_i := b_i \cdot x + a_i$		Accept if and only if:
$z_A := r_B \cdot x + r_A$	$\xrightarrow{f_0, \ldots, f_{m-1}, z_A, z_C}$	$B^x A \stackrel{?}{=} \mathsf{Com}\left((f_0, \ldots, f_{m-1}); z_A\right)$
$z_C := r_C \cdot x + r_D$		$C^x D \stackrel{?}{=} \mathsf{Com}\left((f_k \cdot (x - f_k))_{k=0}^{m-1}; z_C\right)$

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

Finally, we have:

Lemma 4.2 (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 in this section 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 $\Xi\colon \mathbb{F}_q^o \to \mathbb{F}_q^s$) of the commitments $c_{\kappa^j(l)}$ represented by l's *ordered orbit*. We represent Ξ 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:

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

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 Ξ.

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

$\mathcal{P}_2(\sigma, (c_0, \ldots, c_{N-1}), \kappa, \Xi; l, (r_0, \ldots, r_{s-1}))$		$\mathcal{V}_2(\sigma, (c_0, \ldots, c_{N-1}), \kappa, \Xi)$
$r_B, \rho_0, \ldots, \rho_{m-1} \leftarrow\$ \mathbb{F}_q$		
$B := \mathsf{Com}(l_0, \ldots, l_{m-1}; r_B)$		
$(A, C, D) \leftarrow \mathcal{P}_1(\sigma, B; (l_0, \ldots, l_{m-1}, r))$	$\xrightarrow{A, B, C, D}$	
	$\xleftarrow{v}$	$v \leftarrow\$ \mathbb{F}_q$
...... $\mathcal{P}_2$ and $\mathcal{V}_2$ evaluate the matrix product $\left[1, v, \ldots, v^{s-1}\right] \cdot \Xi =: [\xi_0, \ldots, \xi_{o-1}]$, say		
for $k \in \{0, \ldots, m-1\}$ **do**		
$G_k := \prod_{j=0}^{o-1} \left(\prod_{i=0}^{N-1} c_i^{P_{\kappa^{-j}(i),k}} \right)^{\xi_j} \cdot \mathsf{Com}(0; \rho_k)$	$\xrightarrow{G_0, \ldots, G_{m-1}}$	
	$\xleftarrow{x}$	$x \leftarrow\$ \mathbb{F}_q$
$(f_0, \ldots, f_{m-1}, z_A, z_C) \leftarrow \mathcal{P}_1(x)$		Accept if and only if
$z := \left(\sum_{i=0}^{s-1} v^i \cdot r_i \right) \cdot x^m - \sum_{k=0}^{m-1} \rho_k \cdot x^k$	$\xrightarrow{f_0, \ldots, f_{m-1}, z_A, z_C, z}$	$\mathcal{V}_1(\sigma, B, x, A, C, D, (f_k)_{k=0}^{m-1}, z_A, z_C) \stackrel{?}{=} 1$
		$\prod_{j=0}^{o-1} \left(\prod_{i=0}^{N-1} c_i^{p_{\kappa^{-j}(i)}} \right)^{\xi_j} \cdot \prod_{k=0}^{m-1} G_k^{-x^k} \stackrel{?}{=} \mathsf{Com}(0; z)$
		where $\forall k, f_{k,1} := f_k, f_{k,0} := x - f_k$
		and $\forall i \in \{0, \ldots, N-1\}, p_i := \prod_{k=0}^{m-1} f_{k,i_k}$

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

Example 4.3. Setting $\kappa = \mathrm{id} \in \mathbf{S}_N$ the identity permutation, and $\Xi = I_1 \colon \mathbb{F}_q \to \mathbb{F}_q$ the identity *map*, exactly recovers the original protocol of Groth and Kohlweiss [GK15].

Example 4.4. More generally, we consider (for o dividing N) the iterate $\kappa := (0, 1, \ldots, N-1)^{N/o}$, and set $\Xi = I_o$ as the identity map on $\mathbb{F}_q^o$. In this setting, the protocol of this section demonstrates knowledge of a secret residue class $l \mod N/o$, as well as of openings to 0 (say, $r_0, \ldots, r_{o-1}$) of those commitments c_i for which $i \equiv l \mod N/o$.

Example 4.5. Subsection 2.4 sketches the choice $\kappa := (0, 1, \ldots, N-1)$ and $\Xi \colon \mathbb{F}_q^N \to \mathbb{F}_q$ a linear functional. This setting gives a proof of knowledge of a secret permutation $K \in \langle (0, 1, \ldots, N-1) \rangle$ for which the "messages" of $c_{K(0)}, \ldots, c_{K(N-1)}$ reside in a prespecified hyperplane of $\mathbb{F}_q^N$.

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 $(0; \sum_{i=0}^{s-1} v^i \cdot r_i)$ opens the matrix product

$$\begin{bmatrix} 1 & v & \ldots & v^{s-1} \end{bmatrix} \cdot \begin{bmatrix} \Xi \end{bmatrix} \cdot \begin{bmatrix} c_l \\ c_{\kappa(l)} \\ \vdots \\ c_{\kappa^{o-1}(l)} \end{bmatrix},$$

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

Moreover, we have:

Theorem 4.6. *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 $(l, (r_0, \ldots, r_{s-1}))$ 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* v, accepting responses $(f_0, \ldots, f_{m-1}, z_A, z_C, z)$ to $m+1$ distinct challenges x. With recourse to the extractor of Lemma 4.2 and responses to 3 distinct challenges x, $\mathcal{X}$ obtains openings $(b_0, \ldots, b_{m-1}; r_B)$ and $(a_0, \ldots, a_{m-1}; r_A)$ 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 $i \in \{0, \ldots, N-1\}$—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:

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

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 using the inverse's bottom row as coefficients), $\mathcal{X}$ obtains a linear combination of the responses z, say z_v, for which:

$$\prod_{j=0}^{o-1} (c_{\kappa^j(l)})^{\xi_j} = \mathsf{Com}\left(0; z_v\right).$$

In fact, an expression of this form can be obtained for *each* challenge v. Furthermore—now using the definition of $[\xi_0, \ldots, \xi_{o-1}]$—we rewrite this expression's left-hand side as the matrix product:

$$\begin{bmatrix} 1 & v & \ldots & v^{s-1} \end{bmatrix} \cdot \begin{bmatrix} \Xi \end{bmatrix} \cdot \begin{bmatrix} c_l \\ c_{\kappa(l)} \\ \vdots \\ c_{\kappa^{o-1}(l)} \end{bmatrix} = \mathsf{Com}\left(0; z_v\right).$$

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

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

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. $\square$

Finally:

Theorem 4.7. *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

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

where $[\xi_0, \ldots, \xi_{o-1}]$ and $(p_i)_{k=0}^{m-1}$ are computed exactly as is prescribed for the verifier.

We posit some $\mathcal{A}$ attacking $\mathsf{SHVZK}_{\mathcal{A},\mathcal{S},\Pi,\mathcal{R}}$, and define an adversary $\mathcal{A}'$ attacking $\mathsf{MHiding}_{\mathcal{A}',\mathsf{Com}}$ (see Definition 3.5) as follows. To simplify the proof, we modify Definition 3.5 so as to give to the adversary an LR-oracle (in the sense of e.g. [KL15, Def. 11.5]); in this way, we obviate a hybrid argument.

$\mathcal{A}'$ operates as follows. It is given input params.

1. Run $\mathcal{G}(1^\lambda)$, and give $(\mathbb{G}, q, g)$ and params to $\mathcal{A}$.
2. Upon receiving $(c_0, \ldots, c_{N-1}), \kappa, \Xi, l, (r_0, \ldots, r_{s-1})$ and the random coins ρ, honestly compute the elements $A, B, C, D, (f_k)_{k=0}^{m-1}, z_A, z_C$ as prescribed by Fig. 3 (i.e., using the witness l). Compute the polynomials $P_i(X)$, as well as their evaluations p_i, in the standard way. For each $k > 1$, submit the pair of commitments $\left(\prod_{j=0}^{o-1} \left(\prod_{i=0}^{N-1} c_i^{P_{\kappa^{-j}(i),k}} \right)^{\xi_j}, \mathsf{Com}(0) \right)$ to the LR-oracle, so as to obtain the commitment G_k. Randomly generate z, and define G_0 using the final verification equation, as above.
3. Give the transcript tr constructed in this way to $\mathcal{A}$. When $\mathcal{A}$ outputs a bit b', return b'.

If the experimenter's hidden bit $b = 0$, then $\mathcal{A}$'s view in its simulation by $\mathcal{A}'$ exactly matches its view in an honest execution of $\mathsf{SHVZK}_{\mathcal{A},\mathcal{S},\Pi,\mathcal{R}}$ (i.e., tr follows the distribution $\langle \mathcal{P}(\sigma, u, w), \mathcal{V}(\sigma, u; \rho) \rangle$). If on the other hand $b = 1$, then $\mathcal{A}'$'s transcript tr differs from the distribution $\mathcal{S}(\sigma, u; \rho)$ only in that its commitments B and C honestly reflect $\mathcal{A}$'s witness l, whereas $\mathcal{S}$'s do not (i.e., they are simulated as prescribed by Lemma 4.2). This difference at most negligibly impacts $\mathcal{A}$'s advantage, as can be shown by a direct reduction to the SHVZK of Fig. 3. Finally, $\mathcal{A}'$ wins whenever $\mathcal{A}$ does. $\square$

4.3 Efficiency

We discuss the efficiency of our protocol, and argue in particular 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 sketch 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 prover constructs the $P_i(X)$ using a procedure which, essentially, arranges the "upward paths" through the array $F_{k,j}(X)$ 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 many-out-of-many proofs

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 somewhat delicate, and depends in particular on how the map Ξ *grows* with N. Indeed—even assuming that the image dimension $s \leq o$ (which doesn't impact generality)—Ξ could take as much as $\Theta(N^2)$ space to represent; the evaluation of $[1, v, \ldots v^{s-1}] \cdot \Xi$ 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 Ξ has only $O(N)$ nonzero entries as N grows. This ensures that the expression $[1, v, \ldots v^{s-1}] \cdot \Xi$ can be evaluated in linear time. (We note that the *unevaluated* matrix product—represented as a matrix in the indeterminate V—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 "linear combination" trick non-vacuous.

Even assuming this condition on Ξ, 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 Ξ, we nonetheless attain:

Theorem 4.8. *Suppose that the number of nonzero entries of Ξ 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 $(\sigma, (c_0, c_1, \ldots, c_{N-1}), \kappa', \Xi)$, 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 $(\sigma, (c_{\kappa'_0}, c_{\kappa'_1}, \ldots, c_{\kappa'_{N-1}}), \kappa, \Xi)$ 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 $(\sigma, (c_0, c_1, \ldots, c_{N-1}), \kappa'', \Xi; l', (r_0, \ldots, r_{s-1}))$, then, the prover and verifier may use the above reduction to execute $(\sigma, (c_0, c_1, \ldots, c_{N-1}), \kappa', \Xi; l', (r_0, \ldots, r_{s-1}))$; 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 (see e.g. [TAL97, (6.5)]), and the above equation's right-hand side is a circular convolution in the sense of [TAL97, p. 103]. (We assume here that N is a power of 2 and that $N \mid (q-1)$, so that the number-theoretic transform can be applied; see [Nus82, Thm. 8.2]). The verifier may thus evaluate this product in $O(N \log N)$ time using the standard Cooley–Tukey algorithm [TAL97, Thm. 4.2] and the convolution theorem [TAL97, Thm. 6.1].

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 $[\xi_0, \ldots, \xi_{o-1}] := \left[1, v, \ldots, v^{s-1}\right] \cdot \Xi$. By hypothesis on Ξ, this can be done in linear time. $\square$

Remark 4.9. The commitment space in which the commitments c_i reside is *not* in general isomorphic (as an $\mathbb{F}_q$-module) to $\mathbb{F}_q$, let alone efficiently computably so. Nonetheless, we observe that an $\mathbb{F}_q$-module structure alone on this space suffices for the application of Theorem 4.8. This fact is implicit in, say, the statement of [TAL97, Thm. 6.1], where the convolution of two vectors is expressed as a matrix product of the latter.

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 (we present definitions below). 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.

To hint at this flexibility, we sketch a basic example. Consider first the standard relation below, adapted from [GK15, §3]:

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

While [GK15, Fig. 2] easily handles $\mathcal{R}_3$, it's less straightforward to see how it might adapt into a proof for, say, the relation:

$$\mathcal{R}_3^* = \left\{(\sigma, (y_{0,0}, \ldots, y_{0,N-1}), (y_{1,0}, \ldots, y_{1,N-1}); l, \mathsf{sk}) : y_{0,l} = g_0^{\mathsf{sk}} \wedge y_{1,l} = g_1^{\mathsf{sk}}\right\},$$

for bases g_0 and g_1 implicit in the reference string σ, and where, crucially, the same secret key sk must be used in both discrete logarithms. (In another closely related variant, the index l is allowed to be different in both places.) Significantly, our protocol easily adapts to this setting.

5.1 Security definitions

We pause to define the security of ring signature schemes, closely following the article of Bender, Katz, and Morselli [BKM09]. We begin with algorithms $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$. Given parameters $\sigma \leftarrow \mathsf{Setup}(1^\lambda)$, $\mathsf{Gen}(1^\lambda)$ outputs a keypair (y, sk), whereas $\pi \leftarrow \mathsf{Sign}_{s,\mathsf{sk}}(m, R)$ signs the message m on behalf of the ring $R = (y_0, \ldots, y_{N-1})$ (where (y_s, sk) is a valid keypair); finally, $\mathsf{Vrfy}_R(m, \pi)$ verifies the purported signature π on m on behalf of R. We fix a polynomial $N(\cdot)$ in what follows.

Definition 5.1 (Bender–Katz–Morselli [BKM09, Def. 7])**.** The *unforgeability with respect to insider corruption experiment* $\mathsf{UnforgeIC}^{N(\cdot)}_{\mathcal{A},\Pi}(\lambda)$ is defined as:

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

We say that $\Pi = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ 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 5.2 (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. Parameters $\sigma \leftarrow \mathsf{Setup}(1^\lambda)$ are generated and given to $\mathcal{A}$.
2. Keypairs $(y_i, \mathsf{sk}_i)_{i=0}^{N(\lambda)-1}$ are generated using $\mathsf{Gen}(1^\lambda)$, and the list of public keys $S := (y_i)_{i=0}^{N(\lambda)-1}$ is given to $\mathcal{A}$.
3. $\mathcal{A}$ is given access to a *signing oracle* $\mathsf{Osign}(\cdot,\cdot,\cdot)$ such that $\mathsf{Osign}(s, m, R)$ returns $\mathsf{Sign}_{\mathsf{sk}_s}(m, R)$, where we require $y_s \in R$.
4. $\mathcal{A}$ outputs a message m, distinct indices i_0 and i_1, and a ring R for which $y_{i_0}, y_{i_1} \in R$.
5. A random bit b is chosen, and $\mathcal{A}$ is given the signature $\pi \leftarrow \mathsf{Sign}_{\mathsf{sk}_{i_b}}(m, R)$. The adversary outputs a bit b'.
6. The output of the experiment is defined to be 1 if and only if $b' = b$.

We say that $\Pi = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ is *anonymous 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 anonymity 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 (namely, Anonymous Zether).

5.2 Ring signature protocol

We continue with our protocol for the simple relation $\mathcal{R}_3$ above. We construct our correction terms differently than do the protocols [GK15, Fig. 2] and [BCC+15, Fig. 5]; we also replace the final revelation of z by a Schnorr knowledge-of-exponent identification protocol (see e.g., [KL15, Fig. 12.2]). Explicitly:

$\underline{\mathcal{P}_3(\sigma, (y_0, \ldots, y_{N-1}); l, \mathsf{sk})}$		$\underline{\mathcal{V}_3(\sigma, (y_0, \ldots, y_{N-1}))}$
$r_B, r_K, \rho_0, \ldots, \rho_{m-1} \leftarrow\!\!\$\, \mathbb{F}_q$		
$B := \mathsf{Com}(l_0, \ldots, l_{m-1}; r_B)$		
$(A, C, D) \leftarrow \mathcal{P}_1(\sigma, B; (l_0, \ldots, l_{m-1}), r_B)$		
for $k \in \{0, \ldots, m-1\}$ **do**		
$Y_k := \prod_{i=0}^{N-1} y_i^{P_{i,k}} \cdot y_l^{\rho_k}, G_k := g^{\rho_k}$	$\xrightarrow{A, B, C, D, (Y_k, G_k)_{k=0}^{m-1}}$	
$(f_0, \ldots, f_{m-1}, z_A, z_C) \leftarrow \mathcal{P}_1(x)$	$\xleftarrow{x}$	$x \leftarrow\!\!\$\, \mathbb{F}_q$
$\overline{g} := g^{x^m - \sum_{k=0}^{m-1} \rho_k \cdot x^k}$		set $\overline{g} := g^{x^m} \cdot \prod_{k=0}^{m-1} G_k^{-x^k}$
$K := \overline{g}^{r_K}$	$\xrightarrow{f_0, \ldots, f_{m-1}, z_A, z_C, K}$	set $\overline{y} := \prod_{i=0}^{N-1} y_i^{p_i} \cdot \prod_{k=0}^{m-1} Y_k^{-x^k}$
	$\xleftarrow{c}$	$c \leftarrow\!\!\$\, \mathbb{F}_q$
$s := c \cdot \mathsf{sk} + r_K$	$\xrightarrow{s}$	Accept if and only if
		$\mathcal{V}_1(\sigma, B, x, A, C, D, (f_k)_{k=0}^{m-1}, z_A, z_C) \stackrel{?}{=} 1$
		$\overline{g}^s \cdot \overline{y}^{-c} \stackrel{?}{=} K$

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—which, in particular, depends on the witness—and not of a generic Pedersen base element (or of a global public key, as in [BCC+15]).

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.) We then define $x = H((y_i)_{i=0}^{N-1}, A, B, C, D, (Y_k, G_k)_{k=0}^{m-1}, m)$ as well as $c = H(x, K)$; 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 5.3. *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 $\mathsf{UnforgeIC}_{\mathcal{A},\Pi}^{N(\cdot)}(\lambda)$; 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 $\mathsf{UnforgeIC}_{\mathcal{A},\Pi}^{N(\cdot)}(\lambda)$ on $\mathcal{A}$, closely following the spirit of [BKM09, p. 20]. 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]).

We elaborate on this point before beginning, referring to the section *Cauchy interpolation* of von zur Gathen and Gerhard [vzGG13, §5.8]. In fact, step 8. below—and in particular, equation (1)—gives exactly the closely related setting [vzGG13, §5.8, (21)], in which no division is performed; we essentially 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). Details are given inline.

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

1. Generate parameters $\sigma \leftarrow \mathsf{Setup}(1^\lambda)$ for which $\mathbb{G}$, q, and g are as given by the experiment input. Give σ to $\mathcal{A}$.
2. 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 $\mathcal{A}$.
3. Respond to each of $\mathcal{A}$'s random oracle queries with a random element of $\mathbb{F}_q$.
4. 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]).
5. For each query $\mathsf{Corrupt}(i)$ for which $i \neq l$, return sk_i; if $i = l$, abort.
6. When $\mathcal{A}$ outputs (R^*, m^*, π^*), by rewinding it and freshly simulating its random oracle queries, obtain a $(2m+1, 2)$-tree of signatures π^* on R^* and m^* (i.e., for $2m+1$ values of x and, for each one, 2 values of c). If any of the $(2m+1) \cdot 2$ resulting signatures fail to meet the winning condition of $\mathsf{UnforgeIC}_{\mathcal{A},\Pi}^{N(\cdot)}$, or if any collisions occur between the x or c challenges (respectively), then abort.
7. By running the extractor of Lemma 4.2, 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, abort (having obtained a violation of the binding property of the commitment $B^x A$). Otherwise, determine whether the element y^* (say) whose index in R^* is given (in binary) by $b_0, \ldots, b_{m-1}$ equals y_l. If it doesn't, abort.
8. Given these conditions, use the following procedure to obtain, with probability 1, a discrete logarithm $\mathsf{sk} \in \mathbb{F}_q$ for which $g^{\mathsf{sk}} = y_l = h$. 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:

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

for easily computable elements $\widehat{Y_k}$ independent of x. For each particular x, meanwhile, use the standard Schnorr extractor on the two equations $\overline{g}^s \cdot \overline{y}^{-c} = K$ to obtain some quantity $\widehat{\mathsf{sk}}$ (possibly depending on x) for which $\overline{g}^{\widehat{\mathsf{sk}}} = \overline{y}$. Each such pair $(x, \widehat{\mathsf{sk}})$ thus satisfies:

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

By taking the discrete logarithms with respect to g, re-express this relationship as an algebraic equation in two variables, with *unknown* coefficients, which the point $(x, \widehat{\mathsf{sk}})$ satisfies:

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

View the $2m+1$ satisfying pairs $(x, \widehat{\mathsf{sk}})$ of (1) as an instance of [vzGG13, (21)] (using the parameters $n = 2m+1$, $k = m+1$), and in particular 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]. Finally, set $\mathsf{sk} = \log(y_l)$ as $(\mathrm{lc}(t_j))^{-1} \cdot \mathrm{lc}(r_j)$ (where lc returns a polynomial's leading coefficient).

We pause to explain the correctness of step 8. Adapting the proof of [vzGG13, Thm. 5.16, (ii)], we argue that, under the conditions of step 8., there necessarily 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 nonetheless note that because t is monic, α's leading coefficient must equal τ^{-1} (as in [vzGG13, §5.18, (ii)], we denote by $\tau = \text{lc}(t_j)$ the leading coefficient of t_j). We conclude that r's leading coefficient—namely, the desired quantity $\log(y_l)$—is given explicitly by $\tau^{-1} \cdot \text{lc}(r_j)$.

We turn to $\mathcal{A}'$'s correctness probability. 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 output (R^*, m^*, π^*), 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 c 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^*, π^*) yield a violation of the binding property occurs in at most negligibly many executions of $\mathsf{UnforgeIC'}^{N(\cdot)}_{\mathcal{A},\Pi}(\lambda) = 1$ (say 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, c for which $\mathcal{A}$ has already queried (x, K) (where $K := \overline{g}^s \cdot \overline{y}^{-c}$) and for which $H(x, K) \neq c$. This happens with negligible probability (say negl'').

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:

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

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

Remark 5.4. Because the final portion of Fig. 5 is simply a Schnorr protocol on the elements $\overline{g}$ and $\overline{y}$, we may, having applied the Fiat–Shamir transform, rely on the Schnorr proof's SHVZK property, and in particular its simulator. This observation underlies $\mathcal{A}'$'s step 4. above.

Remark 5.5. The proof of Theorem 5.3 (and in particular, $\mathcal{A}'$'s steps 7. and 8.) implicitly demonstrates that the interactive protocol Fig. 5 is $(2m+1, 2)$-special sound for the relation $\mathcal{R}_3$.

We turn to anonymity. We note that the interactive protocol of Fig. 5 is *not* zero-knowledge, or even witness-indistinguishable (see e.g. [GK15, Def. 8]). Indeed, having chosen a statement and candidate witnesses (l_0, sk_0) and (l_1, sk_1), $\mathcal{A}$ (say) may simply return whichever $b' \in \{0,1\}$ satisfies:

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

Analogously, Π is not anonymous against full key exposure (recall [BKM09, Def. 4]). Heuristically, these failures stem 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. Nonetheless, we obtain:

Theorem 5.6. *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 $\mathsf{AnonACK}_{\mathcal{A},\Pi}^{N(\cdot)}$ into an adversary $\mathcal{A}'$ who, assuming that Com is hiding, wins $\mathsf{DDH}_{\mathcal{A}',\mathcal{G}}$ with polynomially related probability. The essential difficulty is that *both* the "messages" of the "ciphertexts" (Y_k, G_k) *and* the key under which they are encrypted depend on the experimenter's hidden bit $b \in \{0,1\}$. This makes the argument below somewhat delicate.

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

1. Generate parameters $\sigma \leftarrow \mathsf{Setup}(1^\lambda)$ for which $\mathbb{G}$, q, and g are as given by the experiment input. Give σ to $\mathcal{A}$.

2. 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 $\mathcal{A}$.

3. Respond to each of $\mathcal{A}$'s random oracle queries with a random element of $\mathbb{F}_q$.

4. 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]).

5. 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.

6. If on the other hand $i_b = l$, construct a signature π 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 protocol, as in [KL15, p. 456]. Give the resulting signature to $\mathcal{A}$.

7. When $\mathcal{A}$ outputs a bit b', return whether $b' \stackrel{?}{=} b$.

For notational ease, we first introduce a modified experiment $\mathsf{AnonACK'}_{\mathcal{A},\Pi}^{N(\cdot)}$, which differs from the standard $\mathsf{AnonACK}_{\mathcal{A},\Pi}^{N(\cdot)}$ *only* in the construction strategy of the signature π. 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 $\mathsf{AnonACK}_{\mathcal{A},\Pi}^{N(\cdot)}$. Otherwise, the experimenter generates a random element, say $y^* \leftarrow \mathbb{G}$, and replaces y_{i_b} with y^* in the construction of π (and also simulates the final Schnorr protocol).

We consider the winning probability of $\mathcal{A}$ in $\mathsf{AnonACK'}_{\mathcal{A},\Pi}^{N(\cdot)}$. If $b'' = 1$, this probability exactly matches that of $\mathcal{A}$ in $\mathsf{AnonACK}_{\mathcal{A},\Pi}^{N(\cdot)}$ (by construction of the former experiment). If on the other hand $b'' = 0$, we argue—assuming now that Com is hiding—that $\Pr[\mathsf{AnonACK'}_{\mathcal{A},\Pi}^{N(\cdot)}(\lambda) = 1]$ differs at most negligibly from $\frac{1}{2}$ (say by negl). To make this explicit, we define a further experiment, $\mathsf{AnonACK''}_{\mathcal{A},\Pi}^{N(\cdot)}$, which represents the case $b'' = 0$ of $\mathsf{AnonACK}'^{N(\cdot)}_{\mathcal{A},\Pi}$; that is, the experimenter *always* replaces y_{i_b} with a random element y^* in the construction of π (and simulates the Schnorr protocol). We claim now that $\Pr[\mathsf{AnonACK}''^{N(\cdot)}_{\mathcal{A},\Pi}(\lambda) = 1] - \frac{1}{2} \leq \mathsf{negl}(\lambda)$.

To this end, we pause to sketch an adversary $\mathcal{A}''$ who, given an algorithm $\mathcal{A}$ targeting $\mathsf{AnonACK}''^{N(\cdot)}_{\mathcal{A},\Pi}$, attacks the *multiple encryptions* experiment $\mathsf{PubK}^{\mathsf{LR\text{-}cpa}}_{\mathcal{A}'',\Pi}$ of [KL15, Def. 11.5]. (We specialize this latter experiment to the El Gamal scheme, invoking both [KL15, Thm. 11.18] and [KL15, Thm. 11.6].) $\mathcal{A}''$ operates as follows, upon receiving a public key y^*:

1. Generate keys $(y_i, \mathsf{sk}_i)_{i=0}^{N(\lambda)-1}$ and give them to $\mathcal{A}$.

2. Respond to $\mathsf{Osign}(\cdot, \cdot, \cdot)$ queries in the obvious way (i.e., honestly).

3. Upon receiving m, i_0, i_1, R from $\mathcal{A}$ run the SHVZK simulator of Lemma 4.2 to construct quantities $A, B, C, D, x, (f_k)_{k=0}^{m-1}, z_A, z_C$, and also construct $(p_i)_{i=0}^{N-1}$ as prescribed by Fig. 5. For each $b \in \{0, 1\}$ and $k \in \{0, \ldots, m-1\}$, assign $b_{b,k} := (i_b)_k$ (i.e., the k^{th} bit of the index i_b) and $a_{b,k} := f_k - b_{b,k} \cdot x$; additionally, define $F_{b,k,1}(X) := b_{b,k} \cdot X + a_{b,k}$ and $F_{b,k,0}(X) := X - F_{b,k,1}(X)$. Finally, set $P_{b,i}(X) := \sum_{k=0}^{m} P_{b,i,k} \cdot X^k := \prod_{k=0}^{m-1} F_{b,k,i_k}(X)$ (for each $b \in \{0, 1\}$ and $i \in \{0, \ldots, N(\lambda) - 1\}$). For each $k \in \{0, \ldots, m-1\}$, submit the pair $\left(\prod_{i=0}^{N(\lambda)-1} y_i^{P_{0,i,k}}, \prod_{i=0}^{N(\lambda)-1} y_i^{P_{1,i,k}}\right)$ to the oracle $\mathsf{LR}_{y^*,b}$, so as to obtain an encryption; call the result (Y_k, G_k). Finally, simulate the Schnorr proof. Give the resulting signature to $\mathcal{A}$.

4. When $\mathcal{A}$ returns a bit b', return whatever $\mathcal{A}$ returns.

We argue that $\mathcal{A}$'s view in its simulation by $\mathcal{A}''$ differs its view in an honest execution of $\mathsf{AnonACK}''^{N(\cdot)}_{\mathcal{A},\Pi}$ only in the (simulated) commitments B and C, and hence (in view of the assumed hiding property of Com) that its advantage drops only negligibly; finally, $\mathcal{A}''$ wins whenever $\mathcal{A}$ wins. This completes the construction.

We return to $\mathsf{AnonACK}'^{N(\cdot)}_{\mathcal{A},\Pi}$. 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 $\mathsf{AnonACK}'^{N(\cdot)}_{\mathcal{A},\Pi}$. (This is by design of the latter experiment; 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.)

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

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

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

6 Application: Anonymous Zether

We turn to our main 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 [BAZB20].

Zether's global state consists of a mapping acc from El Gamal public keys to El Gamal ciphertexts; each y's table entry contains an encryption of y's balance b (in the exponent). In other words:

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

for some randomness r which y in general does not know. (For details on the synchronization issues surrounding "epochs", we refer to [BAZB20].)

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 $(\overline{C}, D)$ (i.e., with the same randomness). 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-add (C, D) and $(\overline{C}, D)$ to 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 [BAZB20, (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 [BAZB20, §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 l_0 and l_1 for which y_{l_0} and y_{l_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_{l_0}, D) encrypts g^{-b^*} under y_{l_0}, (C_{l_1}, D) encrypts g^{b^*} under y_{l_1}, and (C_i, D) for each $i \notin \{l_0, l_1\}$ encrypts g^0 under y_i. To apply the transfer, the contract should (C_i, D) to y_i's balance for each i; 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:

- $l_0, l_1 \in \{0, \ldots, N-1\}$ (sender's and recipient's secret indices),
- sk for which $g^{\mathsf{sk}} = y_{l_0}$ (knowledge of secret key),
- r for which:
  - $g^r = D$ (knowledge of randomness),

- $(y_{l_0} \cdot y_{l_1})^r = C_{l_0} \cdot C_{l_1}$ (sender's and receiver's ciphertexts encrypt opposite balances),
- for each $i \notin \{l_0, l_1\}$, $y_i^r = C_i$ (all ciphertexts other than the sender's and recipient's encrypt 0),

- b^* and b' in $\{0, \ldots, \mathsf{MAX}\}$ for which $C_{l_0} = g^{-b^*} \cdot D$ and $C_{Ln,l_0} = g^{b'} \cdot C_{Rn,l_0}$ (overflow and overdraft protection).

We group these facts into a formal relation, adapting [BAZB20, (8)]. For technical reasons (discussed in Subsection 6.3 below), we actually prove a slight variant of this relation, in which N is required to be *even* and l_0 and l_1 are required to have opposite parities. Formally:

$$\mathsf{st}_{\mathsf{AnonTransfer}} : \quad \begin{gathered} \Big\{ \big((y_i, C_i, C_{Ln,i}, C_{Rn,i})_{i=0}^{N-1}, D, u, g_{\mathsf{epoch}}; \mathsf{sk}, b^*, b', r, l_0, l_1 \big) : \\ g^{\mathsf{sk}} = y_{l_0} \wedge C_{l_0} = g^{-b^*} D^{\mathsf{sk}} \wedge C_{Ln,l_0} = g^{b'} C_{Rn,l_0}^{\mathsf{sk}} \wedge \\ D = g^r \wedge (y_{l_0} \cdot y_{l_1})^r = C_{l_0} \cdot C_{l_1} \wedge \bigwedge_{i \notin \{l_0, l_1\}} y_i^r = C_i \wedge \\ g_{\mathsf{epoch}}^{\mathsf{sk}} = u \wedge b^* \in \{0, \ldots, \mathsf{MAX}\} \wedge b' \in \{0, \ldots, \mathsf{MAX}\} \wedge \\ N \equiv 0 \mod 2 \wedge l_0 \not\equiv l_1 \mod 2 \Big\}. \end{gathered} \tag{2}$$

6.2 Insider and "rogue-key" attacks

We now turn to anonymous payment. We begin with a comment regarding the Anonymous Zether *statement*; in particular this subsection applies equally to this work and to [BAZB20, §D].

An important aspect of the statement (2) is that the same randomness D is used in each El Gamal ciphertext (C_i, D). Yet the appeal of [BAZB20] to Kurosawa [Kur02] (in defense of this measure) appears to misunderstand the latter work. Indeed, as Bellare, Boldyreva and Staddon [BBS03, §1.2] observe, Kurosawa's security definitions are weak, and assume in particular that each adversary is an "outsider".

In contrast, we sketch a plausible insider attack on privacy—analogous to that described in [BBS03, §4]—on any protocol following the paradigm of [BAZB20, §D] (i.e., regardless of its proof system). The attacker, targeting some honest user y, generates a rogue public key $y^* := y^{\mathsf{sk}^*}$ (for some secret and arbitrary sk^*). The attacker then induces some honest user (possibly, but not necessarily, y) to include both the attacker and y in the honest user's anonymity set. The attacker finally obtains the quantity b of y's change in balance (and in particular, determines whether y was the sender, the recipient, or neither) using the following procedure. If y and y^* reside at the indices l and l^* (respectively) of the anonymity set $(y_i)_{i=0}^{N-1}$ (and assuming for simplicity that y^* was neither the recipient nor the sender), the attacker simply determines b using $g^b = C_l \cdot \left((C_{l^*})^{(\mathsf{sk}^*)^{-1}} \right)^{-1}$. The essential mechanism is that the Diffie–Hellman elements of y and y^* with respect to D differ by the same logarithm by which y and y^* differ (namely, sk^*).

We observe that the conduct of the attacker is completely undetectable to y. We emphasize, moreover, that y is at risk even during transactions which she does not initiate; indeed, the sender whom y^* tricks (i.e., into including y and y^*) may be arbitrary (say, unknown to y, for example).

Finally, we note that recourse whereby *separate* randomnesses $(D_i)_{i=0}^{N-1}$ are used is (even were we to leave aside its inefficiency, and in particular its almost-doubling of each transaction's size) not available. In fact, the reuse of D is critical in our cryptographic approach to Anonymous Zether, as we explain below (in Subsection 6.3).

We adopt the remedy suggested by [BBS03, §1.2]. That is, we require each participant to prove knowledge of her *own* public key before participating in the contract (i.e., before appearing in any anonymity set). We implement a "registration" procedure, whereby each public key must sign a specified, fixed message before it participates. This requirement is minimally cumbersome (especially in light of the superfluity of multiple-account use in Anonymous Zether). We suggest the elimination of this requirement as a problem for future work.

We note that this issue does affect *basic* Zether, but vacuously so, in that each transaction's "insiders" (i.e., its sender and recipient) already know each other's respective roles, as well as the amount of funds sent.

6.3 Cryptographic approach to anonymity

We now summarize our proof protocol for Anonymous Zether; our approach uses *many-out-of-many* proofs in a crucial way. 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 $(y_{l_0} \cdot y_{l_1})^r = C_{l_0} \cdot C_{l_1}$, but also that $\bigwedge_{i \notin \{l_0, l_1\}} y_i^r = C_i$, where $g^r = D$ (and l_0 and l_1 are the sender's and receiver's secret indices, respectively).

Put differently, the verifier must iterate (provably) bijectively through *all* N ciphertexts, after having received only two from the prover. To achieve this, the prover and verifier run many-out-of-many proofs *twice*—with secrets l_0 and l_1, respectively—using, in each case, the permutation $\kappa = (0, 2, \ldots, N-2)(1, 3, \ldots, N-1)$ (they also require that N be even). The verifier in this way iterates over the respective orbits of l_0 and l_1 under κ. These orbits, however, aren't necessarily disjoint; indeed, they're either disjoint or identical, accordingly as l_0 and l_1's parities are opposite or equal (respectively). The verifier therefore requires in addition that the two executions be run in such a way that the secrets l_0 and l_1 feature opposite parities. The prover may demonstrate that this condition holds by adapting ideas already present in [GK15] and [BCC+15], as we argue in Subsection 6.5 below.

The verifier, then—instead of using individual matrices Ξ for each execution—interleaves the respective rows yielded by the two executions. The verifier constructs in this way a "double circulant" matrix, represented by the schematic:

$$\begin{bmatrix} \underbrace{0, \ldots\ldots\ldots, 1, \ldots\ldots\ldots, 0}_{\text{1 only at index } l_0} \\ \underbrace{0, \ldots\ldots, 1, \ldots\ldots\ldots\ldots, 0}_{\text{1 only at index } l_1} \\ 0, \ldots\ldots\ldots\ldots\ldots, 1, \ldots, 0 \\ 0, \ldots\ldots\ldots\ldots, 1, \ldots\ldots, 0 \\ \vdots \\ 0, \ldots\ldots, 1, \ldots\ldots\ldots\ldots, 0 \\ 0, \ldots, 1, \ldots\ldots\ldots\ldots\ldots, 0 \end{bmatrix}$$

Figure 6: "Prover's view".

$$\begin{bmatrix} \underbrace{\text{———} \; (p_{0,i})_{i=0}^{N-1} \; \text{———}}_{\text{“1” at unknown even (resp.) or odd index}} \\ \underbrace{\text{———} \; (p_{1,i})_{i=0}^{N-1} \; \text{———}}_{\text{“1” at unknown odd (resp.) or even index}} \\ \text{——} \;\; \text{———} \; (p_{0,i})_{i=0}^{N-1} \; \text{—} \\ \text{——} \;\; \text{———} \; (p_{1,i})_{i=0}^{N-1} \; \text{—} \\ \vdots \\ \text{—} \; (p_{0,i})_{i=0}^{N-1} \; \text{———} \;\; \text{——} \\ \text{—} \; (p_{1,i})_{i=0}^{N-1} \; \text{———} \;\; \text{——} \end{bmatrix}$$

Figure 7: "Verifier's view".

(The vectors $(p_{0,i})_{i=0}^{N-1}$ and $(p_{1,i})_{i=0}^{N-1}$ correspond respectively to the two many-out-of-many executions.) Informally, 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. The matrix so constructed 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 l_0 and l_1 group-theoretically, as that of a certain permutation. Indeed, any indices l_0 and l_1 with opposite parities implicitly yield in this way a permutation $K \in \mathbf{S}_N$, defined by setting, for any $i \in \{0, \ldots, N-1\}$:

$$i \mapsto K(i) := \begin{cases} (l_0 + 2 \cdot k) \mod N & \text{if } i = 2 \cdot k \\ (l_1 + 2 \cdot k) \mod N & \text{if } i = 2 \cdot k + 1. \end{cases}$$

Such permutations K are exactly those residing in the subgroup of $\mathbf{S}_N$ (of order $\frac{N^2}{2}$) given by the generators $\langle (0, 1, \ldots, N-1), (0, 2, \ldots, N-2) \rangle$. This setting thus extends that of standard many-out-of-many proofs (which restrict K to an order-N subgroup).

After interleaving the two executions as described above, the prover and verifier set as Ξ the $(N-1) \times N$

matrix

$$\Xi = \begin{bmatrix} 1 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 \end{bmatrix}.$$

This matrix controls the messages of the permuted ciphertexts $(C_{K(0)}, D), (C_{K(1)}, D) \dots, (C_{K(N-1)}, D)$. Indeed, it encodes exactly that the sum $(C_{l_0}, D) \cdot (C_{l_1}, D)$ is an encryption of 0, whereas the ciphertexts (C_i, D) for $i \notin \{l_0, l_1\}$ individually encrypt 0. (We observe that this matrix has $O(N)$ nonzero entries, and so satisfies the hypothesis of Theorem 4.8.)

We remark finally on the heterogeneity of the keys $y_0, \dots, y_{N-1}$ under which the ciphertexts $(C_0, D), \dots, (C_{N-1}, D)$ are encrypted. We accommodate this challenge using a further trick. We view each pair (C_i, y_i) as a "ciphertext" under the "public key" D (whose "private key" is r). Viewed this way, these pairs indeed *are* homomorphic; we thus conduct our many-out-of-many proofs on the vector $(C_i, y_i)_{i=0}^{N-1}$. Finally, instead of revealing its "randomness" (which the prover generally doesn't know), the prover instead argues that the final result is an encryption of 0 under the "public key" D (using a simple Σ-protocol). Put differently, the prover shows that the linear combination

$$\left(C_{l_0} \cdot C_{l_1} \cdot C_{l_0+2}^{v} \cdot C_{l_1+2}^{v^2} \cdot \dots \cdot C_{l_1-2}^{v^{N-2}}, y_{l_0} \cdot y_{l_1} \cdot y_{l_0+2}^{v} \cdot y_{l_1+2}^{v^2} \cdot \dots \cdot y_{l_1-2}^{v^{N-2}}\right)$$

is an "encryption" of 0 under the "public key" D. This fact suffices for our purposes. The indices l_0 and l_1 of course remain secret.

In a sense, this approach directly generalizes that of basic Zether, which proves that $(y \cdot \overline{y})^r = C \cdot \overline{C}$ (where $g^r = D$).

6.4 Reducing prover runtime to $O(N \log N)$

We reduce the prover's runtime complexity from $O(N \log^2 N)$ to $O(N \log N)$ using a further trick, which we presently sketch. We exploit the special structure of the Anonymous Zether ciphertexts $(C_i, D)_{i=0}^{N-1}$, only $O(1)$ (i.e., 2) of whose messages are nonzero.

We continue to view the pairs (C_i, y_i) as "encryptions" under the "public key" D. A direct application of Fig. 4 would prescribe, for each k, that the prover construct—and the verifier eliminate—the entire k^{th}-order part $\prod_{\iota, j=0}^{1, \frac{N}{2}-1} \left(\prod_{i=0}^{N-1} (C_i, y_i)^{P_{\iota,i,k}}\right)^{\xi_{2 \cdot j + \iota}}$ of the uncorrected many-out-of-many product. Each such correction term *alone* would require $O(N \log N)$ time for the prover to compute (see the treatment of Theorem 4.8).

Yet the messages of $(C_i, D)_{i=0}^{N-1}$ are nonzero only at 2 indices i (namely, l_0 and l_1), so that, for each k, a "messages-only" version of the above correction term can be computed in $O(N)$ time (the messages of the inner expression can be accumulated in constant time for each value (ι, j) of the outer index). The total runtime of the prover is thus $O(N \log N)$. Our approach is to subtract these messages alone; the resulting ciphertext is *still* an encryption of 0 under D (just with more complicated randomness), and so passes verification all the same.

6.5 The opposite parity requirement

We comment further on the requirement that $l_0 \not\equiv l_1 \mod 2$ (whose necessity is explained by the discussion above).

6.5.1 Technique

Cryptographically, we must ensure that two executions of the many-out-of-many procedure correspond to secrets l_0 and l_1 featuring opposite parities. Our technique is to require that the prover commit to the constant- and first-order parts (i.e., respectively, in separate commitments E and F) of the polynomials $F_{0,0}(X) \cdot F_{1,0}(X)$ and $(X - F_{0,0}(X)) \cdot (X - F_{1,0}(X))$. These products are both *a priori* quadratic in X, with leading coefficients given respectively by $b_{0,0} \cdot b_{1,0}$ and $(1 - b_{0,0}) \cdot (1 - b_{1,0})$ (where $b_{0,0}$ and $b_{1,0}$ are the least-significant bits of l_0 and l_1).

The verifier's check $F^x E \stackrel{?}{=} \mathsf{Com}\left((f_{0,0} \cdot f_{1,0}, (x - f_{0,0})(x - f_{1,0}))\right)$, then, enforces exactly that both polynomials above are in fact *linear*, and hence that their leading coefficients are zero. This in turn exactly encodes the logical fact that both $b_{0,0} \wedge b_{1,0} = 0$ and $\neg b_{0,0} \wedge \neg b_{1,0} = 0$, or in other words that $b_{0,0} \oplus b_{1,0} = 1$ (and hence $l_0 \not\equiv l_1 \mod 2$).

Methodologically, this approach naturally extends the standard bit-commitment protocol of [BCC+15, Fig. 4]. In that protocol, the check $B^x \cdot A \stackrel{?}{=} \mathsf{Com}\left((f_{\iota,k})_{\iota,k=0}^{1,m-1}\right)$ encodes exactly the requirement that each $F_{\iota,k}(X)$ be linear, whereas the additional check $C^x D \stackrel{?}{=} \mathsf{Com}\left((f_{\iota,k}(x - f_{\iota,k}))_{\iota,k=0}^{1,m-1}\right)$ enforces that each $F_{\iota,k}(X) \cdot (X - F_{\iota,k})$ is linear (and hence that $F_{\iota,k}(X)$'s leading coefficient $b_{\iota,k} \in \{0, 1\}$).

6.5.2 Privacy implications

The requirement that $l_0 \not\equiv l_1 \mod 2$ decreases privacy, but only minimally so. Indeed, it decreases the cardinality of the set of possible pairs $(l_0, l_1) \in \{0, \ldots, N-1\}^2$ 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 would not be 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.

6.5.3 Eliminating the requirement

We briefly mention, though do not further pursue, an avenue by the aid of which the opposite parity requirement could be eliminated. The prover and verifier could run *standard* many-out-of-many proofs just once, using the "canonical" permutation $\kappa = (0, 1, \ldots, N-1)$, as well as a single secret l_0 representing the sender's index. Moreover, they could set for Ξ the simple "summation" matrix

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

ensuring thereby that the (respective) quantities encrypted by $(C_i, D)_{i=0}^{N-1}$ sum to 0. (In fact, this could be done independently of the many-out-of-many procedure, as this transformation is a symmetric function.) 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,l_0}, C_{Rn,l_0}) and (the negation of) (C_{l_0}, D), this approach would check the ranges of (C_{Ln,l_0}, C_{Rn,l_0}) and *all other* ciphertexts $(C_i, D)_{i \neq l_0}$. (We continue to use a sign representation whereby (C_{l_0}, D) encrypts a nonpositive amount and all other ciphertexts encrypt nonnegative amounts.) Effectively, the many-out-of-many proof would guide the verifier as to which adjustment ciphertext to *exempt* from range checking (while concealing its index in the original list). This approach would convey the additional benefit of allowing more general sorts of transactions, in which a single sender "spreads funds" to many receivers (as is possible in Quisquis [FMMO19]).

Unfortunately, these range checks would require that the N permuted ciphertexts be corrected *individually* (i.e., prior to the application of Ξ). In effect, "being in range" is not invariant under linear transformation. Correcting each term individually would require at least $\Omega(N)$ communication. (In fact, it could be made $\Theta(N)$ by setting the radix of [BCC+15, Fig. 4] equal to N, and sending only 1 correction term per element.)

As this approach would require $\Theta(N)$-sized proofs, we consider it an inferior point in the design space. In fact, its use of Bulletproofs on N quantities (as opposed to just two) would further mandate, in addition to increased computation, a linearly-sized common reference string containing $n \cdot N$ (in practice, $32 \cdot N$) curve points. This too would be cumbersome, as contract storage is expensive.

6.6 Use of ring signatures

We describe how Anonymous Zether uses the ring signature of Fig 5. The relation (2) demands not just that $y_{l_0} = g^{\mathsf{sk}}$, but also that $C_{l_0} = g^{b^*} D^{\mathsf{sk}}$ and $C_{Ln,l_0} = g^{b'} C_{Rn,l_0}^{\mathsf{sk}}$; importantly, the same l_0 *and the same* sk must be used in all three equalities.

Roughly, our approach is as follows. The prover and verifier run the first part of Fig. 5 simultaneously on $(y_i)_{i=0}^{N-1}$ and on $(C_i)_{i=0}^{N-1}$, as well as on the list of pairs $(C_{Ln,i}, C_{Rn,i})_{i=0}^{N-1}$ (in this latter case, the prover adds appropriate correction elements to *both components of* each random pair $(y_{l_0}^{\rho_k}, g^{\rho_k})$). In particular, they use the same values $A, B, C, D, f_k, z_A, z_C$ in each execution, though the prover sends separate correction terms for each. The verifier obtains in this way re-encryptions $\overline{y_0}, \overline{g}, \overline{C_0}, \overline{D}, \overline{C_{Ln}}, \overline{C_{Rn}}$, respectively, of the elements $y_{l_0}, g, C_{l_0}, D, C_{Ln,l_0}, C_{Rn,l_0}$, where the index l_0 is necessarily chosen consistently throughout. Finally, after conducting the Schnorr protocol for $y_{l_0} = g^{\mathsf{sk}}$, as specified by Fig. 5—that is, after verifying that $\overline{g}^{s_{\mathsf{sk}}} \cdot \overline{y_0}^{-c} = A_y$—the prover and verifier continue to use the *same* values k_{sk} and s_{sk} during the rest of the Σ-Bullets procedure (and perform it likewise on the *re-encrypted* elements). In particular, the prover sets $A_b := g^{k_b} \cdot \left(\overline{D}^{-z^2} \cdot \overline{C_{Rn}}^{z^3} \right)^{k_{\mathsf{sk}}}$, whereas the verifier checks

$$g^{-s_b} \cdot A_b \stackrel{?}{=} \left(\overline{D}^{-z^2} \cdot \overline{C_{Rn}}^{z^3} \right)^{s_{\mathsf{sk}}} \cdot \left(\overline{C_0}^{-z^2} \cdot \overline{C_{Ln}}^{z^3} \right)^{-c}.$$

This technique ensures that the same sk for which $y_{l_0} = g^{\mathsf{sk}}$ is used to decrypt (C_{l_0}, D) and (C_{Ln,l_0}, C_{Rn,l_0}). Explicit details are given in our security analyses below.

6.7 Security definitions

We present security definitions for Anonymous Zether, adapting those of Quisquis [FMMO19, §4], as well as the original treatment of [BAZB20, §C]. We refer also to Ben-Sasson, Chiesa, Garman, Green, Miers, Tromer and Virza [SCG+14]. Unlike [BAZB20, §C], we treat a simplified version of Anonymous Zether in which *only* transfers—and no funds or burns—exist (and in which the contract is initialized with its full capacity). This version is simpler to analyze, and evokes the approach of Quisquis [FMMO19, §4.3]. For simplicity, we also ignore the existence of epochs in our analysis, and assume that each transaction takes effect immediately.

We first recall the auxiliary algorithms:

- $(y, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ returns a random keypair (satisfying $y = g^{\mathsf{sk}}$).
- $b \leftarrow \mathsf{Read}(\mathsf{acc}, \mathsf{sk})$ decrypts $\mathsf{acc}[y]$, where $y := g^{\mathsf{sk}}$, and returns its balance b (obtained by "brute-forcing" the exponent, assumed to be in the range $\{0, \ldots, \mathsf{MAX}\}$).

We now have the constituent algorithms:

- $\sigma \leftarrow \mathsf{Setup}\left(1^\lambda\right)$ runs a group-generation algorithm $\mathcal{G}(1^\lambda)$ and generates commitment scheme params.
- $\mathsf{tx} := \left((C_i, y_i)_{i=0}^{N-1}, D, \pi\right) \leftarrow \mathsf{Trans}(\mathsf{acc}, \mathsf{sk}, \overline{y}, R, b^*)$ generates a transfer transaction, given an anonymity set $R = (y_i)_{i=0}^{N-1}$ containing both $y := g^{\mathsf{sk}}$ and $\overline{y}$ (at indices of opposite parity), as well as the contract's current state.
- $\mathsf{acc} \leftarrow \mathsf{Verify}(\mathsf{acc}, \mathsf{tx})$ verifies the transaction tx against acc. If $\mathsf{tx} = \left((C_i, y_i)_{i=0}^{N-1}, D, \pi\right)$ is invalid with respect to acc, Verify sets $\mathsf{acc} = \bot$; otherwise, it returns a new state acc obtained by updating $\mathsf{acc}[y_i] \mathrel{\bullet}= (C_i^{-1}, D^{-1})$ for each $i \in \{0, \ldots, N-1\}$.

We denote by $\Pi = (\mathsf{Setup}, \mathsf{Trans}, \mathsf{Verify})$ the *payment system* defined by these algorithms. For each such Π, we define in addition a stateful smart contract oracle $\mathcal{O}_{\mathsf{SC}}$, which maintains a global state $\mathsf{acc}\colon \mathbb{G} \to \mathbb{G}^2$, and also accepts transactions (upon each of which it calls Verify). $\mathcal{O}_{\mathsf{SC}}$'s state, as well as each transaction sent to it, are visible to all adversaries defined below.

We introduce a generic experiment setup, upon which our further definitions will build:

Definition 6.1. The *generic cryptocurrency experiment* $\mathsf{Crypt}_{\mathcal{A},\Pi}(\lambda)$ is defined as:

1. Parameters $\sigma \leftarrow \mathsf{Setup}\left(1^\lambda\right)$ are generated and given to $\mathcal{A}$.

2. $\mathcal{A}$ outputs a list $(b_i)_{i=0}^{N-1}$ for which each $b_i \in \{0, \ldots, \mathsf{MAX}\}$ and $\sum_{i=0}^{N-1} b_i = \mathsf{MAX}$. For each $i \in \{0, \ldots, N-1\}$, a keypair $(y_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\lambda)$ is generated, and $\mathsf{acc}[y_i] := \mathsf{Enc}_{y_i}(b_i)$ is stored. $\mathcal{O}_{\mathsf{SC}}$ is initialized using the table acc, and $S = (y_i)_{i=0}^{N-1}$ is given to $\mathcal{A}$.

3. $\mathcal{A}$ is given access to an oracle $\mathsf{Transact}(\cdot, \cdot, \cdot, \cdot)$. For each particular call $\mathsf{Transact}(s, \overline{y}, R, b^*)$ for which y_s and $\overline{y}$ reside in R (and occupy indices of opposite parity), a transaction $\mathsf{tx} \leftarrow \mathsf{Trans}(\mathsf{acc}, \mathsf{sk}_s, \overline{y}, R, b^*)$ is generated, and is sent to $\mathcal{O}_{\mathsf{SC}}$ (which executes $\mathsf{acc} \leftarrow \mathsf{Verify}(\mathsf{acc}, \mathsf{tx})$).

4. $\mathcal{A}$ is given access to an oracle $\mathsf{Insert}(\cdot)$, where $\mathsf{Insert}(\mathsf{tx})$ sends tx directly to $\mathcal{O}_{\mathsf{SC}}$.

We define our security properties by adding steps to the above setup.

We first consider "overdraft safety", called "theft prevention" in Quisquis [FMMO19]. The adversary, to win, must produce a valid transaction which increases the total balance of a set of accounts she controls (or, alternatively, which siphons value from an honest account). Unlike Quisquis, we explicitly allow adversarially generated keys; in this light, we require the adversary to reveal its accounts' secret keys (mirroring the revelation of coins in [SCG+14, Def. C.3]).

Definition 6.2. The *overdraft-safety experiment* $\mathsf{Overdraft}_{\mathcal{A},\Pi}(\lambda)$ is defined by adding the following step to $\mathsf{Crypt}_{\mathcal{A},\Pi}(\lambda)$:

5. $\mathcal{A}$ is given access to an oracle $\mathsf{Corrupt}(\cdot)$, where $\mathsf{Corrupt}(i)$ returns sk_i. (Denote by $C \subset S$ the set of corrupted public keys at any particular time.)

6. $\mathcal{A}$ outputs a transaction tx^*, as well as a list of keypairs (y_i^*, sk_i^*). Consider the following conditions:

   (i) There exists some i for which $y_i \in S \backslash C$ and $b \leftarrow \mathsf{Read}(\mathsf{acc}, \mathsf{sk}_i)$ decreases as a result of tx^*.

   (ii) The sum $\sum_{y_i^* \notin S \backslash C} \mathsf{Read}(\mathsf{acc}, \mathsf{sk}_i^*)$ increases as a result of tx^*.

   The result of the experiment is defined to be 1 if (tx^* is valid and) either of these conditions hold. The result is also 1 if any of the Read calls fail to terminate (i.e., with a result $b \in \{0, \ldots, \mathsf{MAX}\}$). Otherwise, the result is 0.

We say that Π is *overdraft-safe* if, for each PPT adversary $\mathcal{A}$, there exists a negligible function negl for which $\Pr[\mathsf{Overdraft}_{\mathcal{A},\Pi}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

We now consider privacy, which we call *ledger-indistinguishability* (following [SCG+14, Def. C.1]). Importantly, we must heed the insecurity of multi-recipient El Gamal under insider attacks, discussed for example in Bellare, Boldyreva and Staddon [BBS03, §4]. Our solution is exactly that of [BBS03, Def. 4.1]; in other words, we require that the adversary reveal the *secret* keys of all adversarially chosen accounts. In particular, this measure prevents the adversary from using a public keys whose secret key it does not know (as in the attack of Subsection 6.2).

In practice, this requirement is captured by our registration procedure, which demands that each account issue a signature on its own behalf. As [BBS03, §4] mention, we could equally well perform key extractions (i.e., from these signatures) *during* our security proofs; this alternative would be unenlightening and cumbersome.

We finally compare this requirement (i.e., that the secret keys be revealed) with that of step 6. of $\mathsf{Overdraft}_{\mathcal{A},\Pi}$ above. Though the two requirements are syntactically similar, that of step 5. below is more restrictive, as we explain now. Requiring that an adversary reveal the keys of those accounts whose value it inflates does not materially hinder the adversary, who needs those keys *anyway* to spend the accounts' funds (as the soundness property of the proof protocol guarantees). On the other hand, an adversary seeking only to distinguish two transactions could *a priori* use arbitrary keys—on whose behalf it does not plan to sign—with no consequence.

Definition 6.3. The *ledger-indistinguishability experiment* $\mathsf{L\text{-}IND}_{\mathcal{A},\Pi}(\lambda)$ is defined by adding the following steps to $\mathsf{Crypt}_{\mathcal{A},\Pi}(\lambda)$:

5. $\mathcal{A}$ outputs indices s_0, s_1, public keys $\overline{y}_0, \overline{y}_1$, quantities b_0^*, b_1^*, and an anonymity set R^*. $\mathcal{A}$ also outputs a secret key sk_i for *each* $y_i \in R^* \backslash S$. Consider the conditions:

   (i) For each $b \in \{0, 1\}$, both y_{s_b} and $\overline{y}_b$ reside in R^* (and occupy indices of opposite parity).

   (ii) If either $\overline{y}_0 \notin S$ or $\overline{y}_1 \notin S$, then $\overline{y}_0 = \overline{y}_1$ (i.e., at the same index) and $b_0^* = b_1^*$.

   If either of these conditions fails to hold, output 0. Otherwise, select a random bit $b \leftarrow \{0, 1\}$ and generate $\mathsf{tx} \leftarrow \mathsf{Trans}(\mathsf{acc}, \mathsf{sk}_{s_b}, \overline{y}_b, R^*, b_b^*)$. Finally, send tx to $\mathcal{O}_{\mathsf{SC}}$.

6. $\mathcal{A}$ outputs a bit b'. The output of the experiment is defined to be 1 if and only if $b' = b$.

We say that Π is *ledger-indistinguishable* if, for each PPT adversary $\mathcal{A}$, there exists a negligible function negl for which $\Pr[\mathsf{L\text{-}IND}_{\mathcal{A},\Pi}(\lambda) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$.

The condition 5.(ii), inspired by Quisquis [FMMO19, §4.3], also exactly encodes the consistency condition of [BBS03, Def. 4.1] (namely that adversary's message vectors coincide over the corrupt keys).

6.8 Protocol and security properties

An explicit (interactive) protocol for Anonymous Zether relation (2) is given in Appendix A. We define the algorithms Trans and Verify by applying the Fiat–Shamir heuristic to this interactive protocol. In this way, we obtain a payment system $\Pi = (\mathsf{Setup}, \mathsf{Trans}, \mathsf{Verify})$.

Theorem 6.4. *If the discrete logarithm problem is hard with respect to $\mathcal{G}$, then Π is overdraft-safe.*

Proof. Deferred to Appendix B. □

We turn to ledger-indistinguishability. We remark that the *interactive* protocol of Appendix A is *not* zero-knowledge, for the same reason that that of Fig. 5 fails to be. Yet just as Fig. 5's non-interactive version *is* anonymous with respect to adversarially chosen keys, so is Anonymous Zether ledger-indistinguishable:

Theorem 6.5. *If the DDH problem is hard with respect to $\mathcal{G}$, then Π is ledger-indistinguishable.*

Proof. Deferred to Appendix C. □

6.9 Performance

We describe our implementation of Anonymous Zether. This implementation is open-source (available at `jpmorganchase / anonymous-zether`), and is fully equipped for use "today". In particular, it supports not just the generation and verification of proofs, but also all "wallet-like" functionality necessary for account maintenance. A simple API facilitates all necessary contract deployments, and exposes "fund", "burn", and "transfer" methods. Proving takes place in a JavaScript library, which is in turn invoked by our wallet (also written in JavaScript). Verification takes place in Solidity contracts.

We report performance measurements below. Each number next to "Transfer" indicates the size of the anonymity set used (including the actual sender and recipient). The *proving time* we report does *not* include the time taken to construct the statement (which is minimal in any case). The *verification time* given reflects the time taken by the local EVM in evaluating a read-only call to the verification contract. Our *proof size* assumes 32-byte field elements and 64-byte (i.e., uncompressed) points, as Ethereum's precompiled contracts require. Our *transaction size* reflects the size of the full Solidity ABI-encoded data payload, which itself includes both a statement and its proof. *Gas used* incorporates not just verification itself, but also the relevant account maintenance associated with the Zether Smart Contract; our gas measurements *do incorporate EIP-1108.*

Our "Burn" transaction is actually a "partial burn", in contrast to that of [BAZB20]; in other words, it allows a user to withdraw only part of her balance (using a single range proof).

	Proving Time (ms)	Verif. Time (ms)	Proof Size (bytes)	Tx. Size (bytes)	Gas Used (units)
Burn	992	50	1,152	1,380	2,405,339
Transfer (2)	1,974	82	2,368	2,948	5,152,200
Transfer (4)	2,139	91	2,944	3,780	5,929,196
Transfer (8)	2,407	110	3,520	4,868	8,519,260
Transfer (16)	3,071	170	4,096	6,468	13,670,599
Transfer (32)	4,569	254	4,672	9,092	24,427,623
Transfer (64)	8,295	491	5,248	13,764	48,764,759
Transfer (N)	$O(N \log N)$	$O(N \log N)$	$O(\log N)$	$O(N)$	$O(N \log N)$

We compare our measurements to those of competing protocols, following the benchmarks given by [FMMO19, §7]. Our proving time is an order of magnitude faster than that of Zcash. Both our proving and verification times are mildly slower than those of Monero and Quisquis (by about two- or three-fold); we emphasize, however, that our reference implementation is in *JavaScript* and Solidity, whereas those systems use C++ and Go (respectively). A "native" implementation of Anonymous Zether would almost certainly challenge those protocols.

Quisquis's proofs grow linearly, whereas ours grow logarithmically. Specifically, our proofs consists of $8\log(N)+22$ group elements and $2\log(N)+12$ field elements, whereas Quisquis's consist of $6N+22\sqrt{N}+52$ group elements and $6N+10\sqrt{N}+39$ field elements (assuming the number of receivers $t=1$). In fact, these latter measurements *exclude* the "intermediate account lists" $\mathsf{inputs}', \mathsf{outputs}'$, $\{\mathsf{acct}_{\delta,i}\}$, and $\{\mathsf{acct}_{\epsilon,i}\}$; were they included, Quisquis's proofs would each include an additional $16N$ group elements.

While our statements grow linearly (a limitation discussed in [BAZB20, §D.2]), we also significantly improve upon Quisquis's constants. Our full transactions (i.e., each including both a statement and its proof) feature $2N+8\log(N)+24$ group elements and $2\log(N)+12$ field elements (plus an additional 196 bytes used in the Solidity ABI encoding), whereas Quisquis's contain $30N+22\sqrt{N}+52$ group elements and $6N+10\sqrt{N}+39$ field elements. Concretely—and using an *uncompressed* (i.e., 64-byte) point representation in both cases, for fair comparison—this amounts, in the case $N=16$, to total transaction sizes of 6.5 kB and 45.3 kB, respectively. (Quisquis's reported 13.4 kB "tx size" actually reflects only its proof, which further still excludes the intermediate account lists; moreover, it assumes a compressed point representation.)

In light of these improvements, our work answers in the affirmative an open problem raised by Fauzi, Meiklejohn, Mercer, and Orlandi [FMMO19, §9], and represents a new state-of-the-art (especially among protocols with constant long-term space overhead per participant).

References

[BAZB20] Benedikt Bünz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. Zether: Towards privacy in a smart contract world. In *International Conference on Financial Cryptography and Data Security*, 2020. Full version.

[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.

[BBS03] Mihir Bellare, Alexandra Boldyreva, and Jessica Staddon. Randomness re-use in multi-recipient encryption schemes. In *Proceedings of the 6th International Workshop on Theory and Practice in Public Key Cryptography*, pages 85–99, 2003. 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.

[FMMO19] Prastudy Fauzi, Sarah Meiklejohn, Rebekah Mercer, and Claudio Orlandi. Quisquis: A new design for anonymous cryptocurrencies. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019*, volume 11921 of *Lecture Notes in Computer Science*. Springer International Publishing, 2019.

[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.

[Kur02] Kaoru Kurosawa. Multi-recipient public-key encryption with shortened ciphertext. In David Naccache and Pascal Paillier, editors, *Public Key Cryptography*, volume 2274 of *Lecture Notes in Computer Science*, pages 48–63, 2002.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

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

[RS13] Dorit Ron and Adi Shamir. Quantitative analysis of the full Bitcoin transaction graph. In Ahmad-Reza Sadeghi, editor, *Financial Cryptography and Data Security*, pages 6–24, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[SCG+14] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from Bitcoin. In *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, pages 459–474, 2014. Full version.

[SMM16] Noether Shen, Adam Mackenzie, and Monero Core Team. Ring confidential transactions. Research Bulletin MRL-0005, Monero Research Lab, February 2016.

[TAL97] Richard Tolimieri, Myoung An, and Chao Lu. *Algorithms for Discrete Fourier Transform and Convolution*. Springer New York, second edition, 1997.

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

A Full Anonymous Zether Protocol

We provide a detailed proof protocol for the Anonymous Zether relation (2). 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 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 [BAZB20], [BBB+18], [GK15] or [BCC+15].

Protocol Anonymous Zether

$\mathcal{P}$ **computes...**
  $\alpha, \rho \leftarrow_\$ \mathbb{F}_q$ ▷ begin Bulletproof [BBB+18, §4]
  $\mathbf{a}_L \in \{0,1\}^{2\cdot n}$ s.t. $\langle \mathbf{a}_{L[:n]}, \mathbf{2}^n \rangle = b^*, \langle \mathbf{a}_{L[n:]}, \mathbf{2}^n \rangle = b'$
  $\mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^{2\cdot n}$
  $\mathbf{A} = h^\alpha \mathbf{g}^{\mathbf{a}_L} \mathbf{h}^{\mathbf{a}_R}$
  $\mathbf{s}_L, \mathbf{s}_R \leftarrow_\$ \mathbb{F}_q^{2\cdot n}$
  $\mathbf{S} = h^\rho \mathbf{g}^{\mathbf{s}_L} \mathbf{h}^{\mathbf{s}_R}$
  $r_A, r_B, r_C, r_D, r_E, r_F \leftarrow_\$ \mathbb{F}_q$ ▷ begin many-out-of-many proof
  for all $\iota \in \{0,1\}$, $k \in \{0, \ldots, m-1\}$ **do**
    sample $a_{\iota,k} \leftarrow_\$ \mathbb{F}_q$
    set $b_{\iota,k} = (l_\iota)_k$, i.e., the k^{th} (little-endian) bit of l_ι
  end for
  $A = \mathsf{Com}(a_{0,0}, \ldots, a_{1,m-1}; r_A)$
  $B = \mathsf{Com}(b_{0,0}, \ldots, b_{1,m-1}; r_B)$
  $C = \mathsf{Com}((a_{\iota,k}(1 - 2b_{\iota,k}))_{\iota,k=0}^{1,m-1}; r_C)$
  $D = \mathsf{Com}(-a_{0,0}^2, \ldots, -a_{1,m-1}^2; r_D)$
  $E = \mathsf{Com}\left((a_{0,0} \cdot a_{1,0}, a_{0,0} \cdot a_{1,0}); r_E\right)$
  $F = \mathsf{Com}\left((a_{b_{0,0},0}, -a_{b_{1,0},0}); r_F\right)$ ▷ the bits $b_{0,0}$ and $b_{1,0}$ are used as indices here.
end $\mathcal{P}$
$\mathcal{P} \rightarrow \mathcal{V} : \mathbf{A}, \mathbf{S}, A, B, C, D, E, F$
$\mathcal{V} : v \leftarrow_\$ \mathbb{F}_q$
$\mathcal{V} \rightarrow \mathcal{P} : v$
$\mathcal{P}$ **computes...** ▷ in what follows, we denote by i_k the k^{th} bit of i.
  for all $\iota \in \{0,1\}$ **do**
    for all $k \in \{0, \ldots, m-1\}$ **do**
      set $F_{\iota,k,1}(W) := b_{\iota,k} \cdot W + a_{\iota,k}$
      set $F_{\iota,k,0}(W) := W - F_{\iota,k,1}(W)$
    end for
    for all $i \in \{0, \ldots, N-1\}$ **do**
      set $P_{\iota,i}(W) := \sum_{k=0}^{m} P_{\iota,i,k} \cdot W^k := \prod_{k=0}^{m-1} F_{\iota,k,i_k}(W)$
    end for
  end for
  $(\phi_k, \chi_k, \psi_k, \omega_k)_{k=0}^{m-1} \leftarrow_\$ \mathbb{F}_q$
  set $(\xi_0, \xi_1, \xi_2, \xi_3, \ldots, \xi_{N-1}) = \left(1, 1, v, v^2, \ldots, v^{N-2}\right)$
  for all $k \in \{0, \ldots, m-1\}$ **do**
    $\widetilde{C_{Ln,k}} = \mathsf{MultiExp}\left((C_{Ln,i})_{i=0}^{N-1}, (P_{0,i,k})_{i=0}^{N-1}\right) \cdot (y_{l_0})^{\phi_k}$
    $\widetilde{C_{Rn,k}} = \mathsf{MultiExp}\left((C_{Rn,i})_{i=0}^{N-1}, (P_{0,i,k})_{i=0}^{N-1}\right) \cdot g^{\phi_k}$
    $\widetilde{C_{0,k}} = \mathsf{MultiExp}\left((C_i)_{i=0}^{N-1}, (P_{0,i,k})_{i=0}^{N-1}\right) (y_{l_0})^{\chi_k}$

$\widetilde{D_k} = g^{\chi_k}$
$\widetilde{y_{0,k}} = \mathsf{MultiExp}\left((y_i)_{i=0}^{N-1}, (P_{0,i,k})_{i=0}^{N-1}\right)(y_{l_0})^{\psi_k}$
$\widetilde{g_k} = g^{\psi_k}$
$\widetilde{C_{X,k}} = \left(\prod_{\iota,j=0}^{1,\frac{N}{2}-1} g^{b^* \cdot \left(-P_{\iota,l_0-2\cdot j,k}+P_{\iota,l_1-2\cdot j,k}\right)}\right)^{\xi_{2\cdot j+\iota}} \cdot D^{\omega_k}$
$\widetilde{y_{X,k}} = g^{\omega_k}$
end for
end $\mathcal{P}$
$\mathcal{P} \to \mathcal{V} : \left(\widetilde{C_{Ln,k}}, \widetilde{C_{Rn,k}}, \widetilde{C_{0,k}}, \widetilde{D_k}, \widetilde{y_{0,k}}, \widetilde{g_k}, \widetilde{C_{X,k}}, \widetilde{y_{X,k}}\right)_{k=0}^{m-1}$
$\mathcal{V} : w \leftarrow\!\!\$\, \mathbb{F}_q$
$\mathcal{V} \to \mathcal{P} : w$
$\mathcal{P}$ **computes...**
for all $\iota \in \{0,1\}$, $k \in \{0,\dots,m-1\}$ **do**
set $f_{\iota,k} := F_{\iota,k,1}(w)$
end for
$z_A = r_B \cdot w + r_A$
$z_C = r_C \cdot w + r_D$
$z_E = r_F \cdot w + r_E$
end $\mathcal{P}$
$\mathcal{P} \to \mathcal{V} : (f_{\iota,k})_{\iota,k=0}^{1,m-1}, z_A, z_C, z_E$
$\mathcal{V} : y, z \leftarrow\!\!\$\, \mathbb{F}_q$
$\mathcal{V} \to \mathcal{P} : y, z$
$\mathcal{P}$ **:**
$l(X) = (\mathbf{a}_L - z \cdot \mathbf{1}^n) + \mathbf{s}_L \cdot X$
$r(X) = \mathbf{y}^n \circ (\mathbf{a}_R + z \cdot \mathbf{1}^n + \mathbf{s}_R \cdot X) + z^2 \cdot (\mathbf{2}^n \parallel \mathbf{0}^n) + z^3 \cdot (\mathbf{0}^n \parallel \mathbf{2}^n)$
$t(X) = \langle l(X), r(X) \rangle = t_0 + t_1 \cdot X + t_2 \cdot X^2$ ▷ l and r are elements of $\mathbb{F}_q^{2\cdot n}[X]$; $t \in \mathbb{F}_q[X]$
$\tau_1, \tau_2 \leftarrow\!\!\$\, \mathbb{F}_q$
$T_i = g^{t_i} h^{\tau_i}$ **for** $i \in \{1,2\}$
end $\mathcal{P}$
$\mathcal{P} \to \mathcal{V} : T_1, T_2$
$\mathcal{V} : x \leftarrow\!\!\$\, \mathbb{F}_q$
$\mathcal{V} \to \mathcal{P} : x$
$\mathcal{P}$ **computes...**
$\mathbf{l} = l(x) = \mathbf{a}_L - z \cdot \mathbf{1}^{2\cdot n} + \mathbf{s}_L \cdot x$
$\mathbf{r} = r(x) = \mathbf{y}^{2\cdot n} \circ (\mathbf{a}_R + z \cdot \mathbf{1}^{2\cdot n} + \mathbf{s}_R \cdot x) + z^2 \cdot (\mathbf{2}^n \parallel \mathbf{0}^n) + z^3 \cdot (\mathbf{0}^n \parallel \mathbf{2}^n)$
$\hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle$ ▷ $\mathbf{l}$ and $\mathbf{r}$ are elements of $\mathbb{F}_q^{2\cdot n}$; $\hat{t} \in \mathbb{F}_q$
$\tau_x = \tau_2 \cdot x^2 + \tau_1 \cdot x$
$\mu = \alpha + \rho \cdot x$
$\overline{C_{Rn}} = (C_{Rn,l_0})^{w^m} \cdot \left(\prod_{k=0}^{m-1} g^{-\phi_k \cdot w^k}\right)$ ▷ prover "anticipates" certain re-encryptions
$\overline{D} = D^{w^m} \cdot g^{-\sum_{k=0}^{m-1} \chi_k \cdot w^k}$
$\overline{y_0} = (y_{l_0})^{w^m} \cdot \left(\prod_{k=0}^{m-1} y_{l_0}^{-\psi_k \cdot w^k}\right)$
$\overline{g} = g^{w^m - \sum_{k=0}^{m-1} \psi_k \cdot w^k}$
$\overline{y_{X,k}} = \prod_{\iota,j=0}^{1,\frac{N}{2}-1} \mathsf{MultiExp}\left((y_i)_{i=0}^{N-1}, \mathsf{Shift}\left((P_{\iota,i}(w))_{i=0}^{N-1}, 2\cdot j\right)\right)^{\xi_{2\cdot j+\iota}} \cdot \left(\prod_{k=0}^{m-1} g^{-\omega_k \cdot w^k}\right)$
$k_{\mathsf{sk}}, k_r, k_b, k_\tau \leftarrow\!\!\$\, \mathbb{F}_q$ ▷ begin Σ-protocol proving
$A_y = \overline{g}^{k_{\mathsf{sk}}}$
$A_D = g^{k_r}$
$A_b = g^{k_b} \cdot \left(\overline{D}^{-z^2} \cdot \overline{C_{Rn}}^{z^3}\right)^{k_{\mathsf{sk}}}$
$A_X = \overline{y_X}^{k_r}$ ▷ in some sense, A_X replaces $A_{\overline{y}}$
$A_t = g^{-k_b} h^{k_\tau}$

$A_u = g_{\mathsf{epoch}}^{k_{\mathsf{sk}}}$
end $\mathcal{P}$
$\mathcal{P} \to \mathcal{V} : \hat{t}, \mu, A_y, A_D, A_b, A_X, A_t, A_u$
$\mathcal{V} : c \leftarrow_{\$} \mathbb{F}_q$
$\mathcal{V} \to \mathcal{P} : c$
$\mathcal{P}$ **computes...**
$s_{\mathsf{sk}} = k_{\mathsf{sk}} + c \cdot \mathsf{sk}$
$s_r = k_r + c \cdot r$
$s_b = k_b + c \cdot w^m \cdot (b^* z^2 + b' z^3)$
$s_\tau = k_\tau + c \cdot w^m \cdot \tau_x$
end $\mathcal{P}$
$\mathcal{P} \to \mathcal{V} : s_{\mathsf{sk}}, s_r, s_b, s_\tau$
$\mathcal{V}$ **requires...**
for all $\iota \in \{0, 1\}$, $k \in \{0, \ldots, m-1\}$ **do**
set $f_{\iota,k,1} = f_{\iota,k}$
set $f_{\iota,k,0} = w - f_{\iota,k}$
for all $i \in \{0, \ldots, N-1\}$ **do**
set $p_{\iota,i} = \prod_{k=0}^{m-1} f_{\iota,k,k_i}$
end for
end for
$B^w A \stackrel{?}{=} \mathsf{Com}\left(f_{0,0}, \ldots, f_{1,m-1}; z_A\right)$
$C^w D \stackrel{?}{=} \mathsf{Com}\left(\left(f_{\iota,k}(w - f_{\iota,k})\right)_{\iota,k=0}^{1,m-1}; z_C\right)$
$F^w E \stackrel{?}{=} \mathsf{Com}\left((f_{0,0} \cdot f_{1,0}, (w - f_{0,0})(w - f_{1,0})); z_E\right)$ ▷ opposite parity check, see Subsection 6.5
$\overline{C_{Ln}} = \mathsf{MultiExp}\left((C_{Ln,i})_{i=0}^{N-1}, (p_{0,k})_{i=0}^{N-1}\right) \cdot \prod_{k=0}^{m-1} \widetilde{C_{Ln,k}}^{-w^k}$ ▷ begin comp. of re-encryptions
$\overline{C_{Rn}} = \mathsf{MultiExp}\left((C_{Rn,i})_{i=0}^{N-1}, (p_{0,k})_{i=0}^{N-1}\right) \cdot \prod_{k=0}^{m-1} \widetilde{C_{Rn,k}}^{-w^k}$
$\overline{C_0} = \mathsf{MultiExp}\left((C_i)_{i=0}^{N-1}, (p_{0,k})_{i=0}^{N-1}\right) \cdot \prod_{k=0}^{m-1} \widetilde{C_{0,k}}^{-w^k}$
$\overline{D} = D^{w^m} \cdot \prod_{k=0}^{m-1} \widetilde{D_k}^{-w^k}$
$\overline{y_0} = \mathsf{MultiExp}\left((y_i)_{i=0}^{N-1}, (p_{0,k})_{i=0}^{N-1}\right) \cdot \prod_{k=0}^{m-1} \widetilde{y_{0,k}}^{-w^k}$
$\overline{g} = g^{w^m} \cdot \prod_{k=0}^{m-1} \widetilde{g_k}^{-w^k}$
set $(\xi_0, \xi_1, \xi_2, \xi_3, \ldots, \xi_{N-1}) = \left(1, 1, v, v^2, \ldots, v^{N-2}\right)$
$\overline{C_X} = \prod_{\iota,j=0}^{1,\frac{N}{2}-1} \mathsf{MultiExp}\left((C_i)_{i=0}^{N-1}, \mathsf{Shift}\left((p_{\iota,i})_{i=0}^{N-1}, 2 \cdot j\right)\right)^{\xi_{2 \cdot j + \iota}} \cdot \prod_{k=0}^{m-1} \widetilde{C_{X,k}}^{-w^k}$
$\overline{y_X} = \prod_{\iota,j=0}^{1,\frac{N}{2}-1} \mathsf{MultiExp}\left((y_i)_{i=0}^{N-1}, \mathsf{Shift}\left((p_{\iota,i})_{i=0}^{N-1}, 2 \cdot j\right)\right)^{\xi_{2 \cdot j + \iota}} \cdot \prod_{k=0}^{m-1} \widetilde{y_{X,k}}^{-w^k}$
$A_y \stackrel{?}{=} \overline{g}^{s_{\mathsf{sk}}} \cdot \overline{y_0}^{-c}$ ▷ begin Σ-protocol verification
$A_D \stackrel{?}{=} g^{s_r} \cdot D^{-c}$
$g^{-s_b} \cdot A_b \stackrel{?}{=} \left(\overline{D}^{-z^2} \cdot \overline{C_{Rn}}^{z^3}\right)^{s_{\mathsf{sk}}} \cdot \left(\overline{C_0}^{-z^2} \cdot \overline{C_{Ln}}^{z^3}\right)^{-c}$
$A_X \stackrel{?}{=} \overline{y_X}^{s_r} \cdot \overline{C_X}^{-c}$
$\delta(y, z) = (z - z^2) \cdot \left\langle \mathbf{1}^{2 \cdot n}, \mathbf{y}^{2 \cdot n} \right\rangle - \left(z^3 \cdot \langle \mathbf{1}^n, \mathbf{2}^n \rangle + z^4 \cdot \langle \mathbf{1}^n, \mathbf{2}^n \rangle\right)$
$g^{w^m \cdot c \cdot \hat{t}} \cdot h^{s_\tau} \stackrel{?}{=} g^{w^m \cdot c \cdot \delta(y,z)} \cdot g^{s_b} \cdot A_t \cdot \left(T_1^x \cdot T_2^{x^2}\right)^{w^m \cdot c}$
$A_u \stackrel{?}{=} g_{\mathsf{epoch}}^{s_{\mathsf{sk}}} \cdot u^{-c}$
end $\mathcal{V}$
$\mathbf{h}' = \left(h_0, h_1^{y^{-1}}, h_2^{y^{-2}}, \ldots, h_{2 \cdot n-1}^{y^{-2 \cdot n+1}}\right)$ ▷ complete inner product argument
$P = \mathbf{A} \cdot \mathbf{S}^x \cdot \mathbf{g}^{-z} \cdot \mathbf{h}'^{z \cdot \mathbf{y}^{2 \cdot n} + z^2 \cdot (\mathbf{2}^n \| \mathbf{0}^n) + z^3 \cdot (\mathbf{0}^n \| \mathbf{2}^n)}$
$\mathcal{P}$ and $\mathcal{V}$ engage in Protocol 1 of [BBB+18] on inputs $(\mathbf{g}, \mathbf{h}', Ph^{-\mu}, \hat{t}; \mathbf{l}, \mathbf{r})$

B Overdraft Safety: Proof

Proof of Theorem 6.4. We fix an adversary $\mathcal{A}$ targeting $\mathsf{Overdraft}_{\mathcal{A},\Pi}(\lambda)$; assuming that Com is binding, we yield an adversary $\mathcal{A}'$ which wins $\mathsf{DLog}_{\mathcal{A}',\mathcal{G}}(\lambda)$ with polynomially related probability. (By specializing to Pedersen commitments, we incorporate the binding property of Com "for free" in the hypothesis of the Theorem.)

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

1. Generate parameters $\sigma \leftarrow \mathsf{Setup}\left(1^\lambda\right)$ for which $\mathbb{G}$, q, and g are as given by the experiment input. Give σ to $\mathcal{A}$.
2. Given the list $(b_i)_{i=0}^{N-1}$, generate a keypair $(y_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\lambda)$ for each $i \in \{0, \ldots, N-1\}$. For a randomly chosen index $l \in \{0, \ldots, N-1\}$, overwrite $y_l := h$. Encrypt $\mathsf{acc}[y_i] := \mathsf{Enc}_{y_i}(b_i)$ for each i and initialize $\mathcal{O}_{\mathsf{SC}}$ with acc. Finally, give the *modified* list $S = (y_i)_{i=0}^{N-1}$ to $\mathcal{A}$.
3. Respond to each of $\mathcal{A}$'s random oracle queries with a random element of $\mathbb{F}_q$.
4. For each oracle query $\mathsf{Transact}(s, \overline{y}, R, b^*)$ for which $s \neq l$, simply compute $\mathsf{tx} := \left((C_i, y_i)_{i=0}^{N-1}, D, \pi\right) \leftarrow \mathsf{Trans}(\mathsf{acc}, \mathsf{sk}_s, \overline{y}, R, b^*)$ as specified by the Anonymous Zether protocol. If instead $s = l$, construct π by replacing each Schnorr protocol involving sk by a simulation. More specifically, randomly generate s_{sk} and c. Finally, set $A_y := \overline{g}^{s_{\mathsf{sk}}} \cdot \overline{y_0}^{-c}$ and $A_t := g^{w^m \cdot c \cdot (\hat{t} - \delta(y,z))} \cdot h^{w^m \cdot c \cdot \tau_x} \cdot c_{\mathsf{commit}} \cdot \left(T_1^x \cdot T_2^{x^2}\right)^{-w^m \cdot c}$. Compute all other elements as specified by the protocol.
5. For each oracle query $\mathsf{Insert}(\mathsf{tx})$, forward tx to $\mathcal{O}_{\mathsf{SC}}$.
6. For each oracle query $\mathsf{Corrupt}(i)$ for which $i \neq l$, return sk_i. If $i = l$, abort.
7. When $\mathcal{A}$ outputs $\mathsf{tx}^* = \left((C_i, y_i)_{i=0}^{N-1}, D, \pi^*\right)$, check if either of the conditions of step 6. of $\mathsf{Overdraft}_{\mathcal{A},\Pi}(\lambda)$ hold. If not, then abort. If either does, then rewind $\mathcal{A}$ so as to obtain an $(N-1, 2m+1, 2 \cdot n, 4, 3, 2)$-tree of proofs π^* (by freshly simulating its random oracle queries for the challenges (v, w, y, z, x, c), respectively). If any of the resulting leaves π^* fails to be valid, or if any collisions in the simulated challenges occur, then abort.
8. By running the extractor of Lemma 4.2, obtain openings $b_{0,0}, \ldots, b_{1,m-1}$, $a_{0,0}, \ldots, a_{1,m-1}$ of the initial commitments B and A for which $b_{\iota,k} \in \{0, 1\}$. If for *any* w it holds that $f_{\iota,k} \neq b_{\iota,k} \cdot w + a_{\iota,k}$ for some ι, k, abort (having obtained a violation of the binding property of the commitment $B^w A$). Otherwise, determine whether the element y^* (say) whose index in $R^* := (y_i)_{i=0}^{N-1}$ is given (in binary) by $b_{0,0}, \ldots, b_{0,m-1}$ equals y_l. If it doesn't, abort.
9. Given these conditions, use the exact same procedure as in Theorem 5.3, step 8. (on the first list of bits, $b_{0,0}, \ldots, b_{0,m-1}$) to obtain, with probably 1, sk for which $g^{\mathsf{sk}} = y_l$. Return sk.

We first introduce a modified experiment $\mathsf{Overdraft}'_{\mathcal{A},\Pi}$, which differs from the standard $\mathsf{Overdraft}_{\mathcal{A},\Pi}$ *only* in that the experimenter, having received $\mathsf{tx}^* = \left((C_i, y_i)_{i=0}^{N-1}, D, \pi^*\right)$, rewinds $\mathcal{A}$ so as to obtain a $(N-1, 2m+1, 2 \cdot n, 4, 3, 2)$-tree of proofs π^* (freshly simulating its random oracle queries), and imposes the winning condition 6. of $\mathsf{Overdraft}_{\mathcal{A},\Pi}$ on all leaves (aborting also if any challenge collisions occur in the tree). Exactly as in [KL15, Thm. 12.11] and in Theorem 5.3, after applying Jensen's inequality iteratively, we see that for some negligible function negl, $\Pr[\mathsf{Overdraft}'_{\mathcal{A},\Pi}(\lambda) = 1] \geq \Pr[\mathsf{Overdraft}_{\mathcal{A},\Pi}(\lambda) = 1]^{(N-1)\cdot(2m+1)\cdot 2 \cdot n \cdot 24} - \mathsf{negl}(\lambda)$.

Now, let $\mathsf{Overdraft}''_{\mathcal{A},\Pi}$ differ further from $\mathsf{Overdraft}'_{\mathcal{A},\Pi}$ only in that, after requiring the validity of all leaves π^*, the experimenter imposes upon them *instead* of 6. an (apparently stronger) winning condition, whereby no binding violation in the $(b_{\iota,k}, a_{\iota,k})_{\iota,k=0}^{1,m-1}$ occurs (as in step 8. above) and in addition $y^* \in S \backslash C$. (In effect $\mathcal{A}$ must forge the ring signature of Fig. 5 on an honest user's behalf.) We argue that the joint event in which $\mathcal{A}$ wins $\mathsf{Overdraft}'_{\mathcal{A},\Pi}$ *and* fails to satisfy the winning condition of $\mathsf{Overdraft}''_{\mathcal{A},\Pi}$ occurs in at most negligibly many executions of $\mathsf{Overdraft}'_{\mathcal{A},\Pi}$ (say negl'). In fact, this follows exactly from (the binding property of Com and) the $(N-1, 2m+1, 2 \cdot n, 4, 3, 2)$-soundness of the interactive protocol of

Appendix A, whose proof we further defer to a lemma below. Indeed, if all π^* are valid *and* $\mathcal{A}$'s statement $((C_i, y_i)_{i=0}^{N-1}, D)$ has a witness $(\mathsf{sk}, b^*, b', r, l_0, l_1)$ in the sense of (2) (and no $B^w A$ binding violation occurs), then the winning conditions of $\mathsf{Overdraft}'_{\mathcal{A},\Pi}$ and $\mathsf{Overdraft}''_{\mathcal{A},\Pi}$ are equivalent; in particular, in this case 6.(ii) implies 6.(i), which in turn implies $y^* \in S \backslash C$.

We finally argue that $\mathcal{A}$'s view in its simulation by $\mathcal{A}'$ differs from its view in $\mathsf{Overdraft}''_{\mathcal{A},\Pi}$ *only* if $\mathcal{A}'$ aborts (i.e., upon receiving a call $\mathsf{Corrupt}(l)$, or if $y^* \neq y_l$) or if $\mathcal{A}'$'s response to some $\mathsf{Transact}(l, \overline{y}, R, b^*)$ query features a Schnorr simulation whose implicit random oracle response clashes with a prior evaluation. The latter event happens in negligibly many executions of $\mathcal{A}'$ (say negl''); among executions for which it does not occur, $\mathcal{A}$ necessarily queries $\mathsf{Corrupt}(l)$ (for a random $l \in \{0, \ldots, N-1\}$, chosen in advance) in at most $\frac{1}{N}$ among executions for which the winning condition of $\mathsf{Overdraft}''_{\mathcal{A},\Pi}$ holds.

Putting these facts together, we see that:

$$\begin{aligned}
\Pr[\mathsf{DLog}_{\mathcal{A}',\mathcal{G}}(\lambda) = 1] &\geq \Pr[\mathsf{Overdraft}''_{\mathcal{A},\Pi}(\lambda) = 1 \wedge y^* = y_l] - \mathsf{negl}''(\lambda) \\
&\geq \frac{1}{N} \cdot \Pr[\mathsf{Overdraft}''_{\mathcal{A},\Pi}(\lambda) = 1] - \mathsf{negl}''(\lambda) \\
&\geq \frac{1}{N} \cdot \big(\Pr[\mathsf{Overdraft}'_{\mathcal{A},\Pi}(\lambda) = 1] - \mathsf{negl}'(\lambda)\big) - \mathsf{negl}''(\lambda) \\
&\geq \frac{1}{N} \cdot \Big(\Pr[\mathsf{Overdraft}_{\mathcal{A},\Pi}(\lambda) = 1]^{(N-1)\cdot(2m+1)\cdot 2\cdot n\cdot 24} - \mathsf{negl}(\lambda) - \mathsf{negl}'(\lambda)\Big) - \mathsf{negl}''(\lambda).
\end{aligned}$$

This completes the proof. $\square$

It thus remains only to show:

Lemma B.1. *If the discrete logarithm problem is hard with respect to $\mathcal{G}$, then the interactive protocol of Appendix A for (2) is $(N-1, 2m+1, 2\cdot n, 4, 3, 2)$-special sound.*

Proof. We describe an extractor $\mathcal{X}$ which, given an arbitrary statement $((y_i, C_i, C_{Ln,i}, C_{Rn,i})_{i=0}^{N-1}, D)$ and an $(N-1, 2m+1, 2\cdot n, 4, 3, 2)$-tree of accepting transcripts, returns either a witness $(\mathsf{sk}, b^*, b', r, l_0, l_1)$ as in (2) or a binding violation. The latter event can be shown to be negligibly probable under the assumption that Com is binding, by a direct reduction. (By specializing Com to the Pedersen scheme, we obtain Com's binding property from the discrete logarithm hypothesis alone.)

$\mathcal{X}$ first performs a Σ-Bullets extraction, as in [BAZB20, §G], modified to suit our "re-encrypted" setting. For notational convenience, we assume in what follows that each challenge $w \neq 0$.

For each particular assignment of values to the challenges w, y, z, x (and for arbitrarily chosen v), by using the standard Schnorr extractor on the two leaves c, $\mathcal{X}$ obtains from the verification equation $A_y = \overline{g}^{s_{\mathsf{sk}}} \cdot \overline{y_0}^{-c}$ a quantity $\widehat{\mathsf{sk}}$ (*a priori* unequal to sk) for which $\overline{g}^{\widehat{\mathsf{sk}}} = \overline{y_0}$. Similarly, from the two copies of the Σ-Bullets verification equation 121, $\mathcal{X}$ obtains a quantity b (which we tacitly divide by w^m) and an equality:

$$g^{w^m \cdot b} = \left(\overline{D}^{-\widehat{\mathsf{sk}}} \cdot \overline{C_0}\right)^{-z^2} \cdot \left(\overline{C_{Rn}}^{-\widehat{\mathsf{sk}}} \cdot \overline{C_{Ln}}\right)^{z^3} \tag{3}$$

for $\widehat{\mathsf{sk}}$ as above. Finally, from the two copies of the Bulletproofs verification equation 124 (and again dividing throughout by w^m), $\mathcal{X}$ obtains a quantity τ_x, as well as an expression:

$$g^{\hat{t}} \cdot h^{\tau_x} = g^{\delta(y,z)} \cdot g^b \cdot T_1^x \cdot T_2^{x^2}. \tag{4}$$

$\mathcal{X}$ now performs a Bulletproofs extraction, essentially as in [BBB+18, §C]. For completeness, we carry through the details. For each assignment of values to the challenges w, y, z (and v as before), $\mathcal{X}$ runs for each value x the inner product extractor, so as to obtain vectors $\mathbf{l}$ and $\mathbf{r}$ for which $P = h^\mu \cdot \mathbf{g}^{\mathbf{l}} \cdot (\mathbf{h}')^{\mathbf{r}}$ and $\hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle$. (Technically, our tree of transcripts should be augmented so as to incorporate, for each x, a $\log(n)$-depth tree containing $O(n^2)$ further transcripts, upon which the inner product extractor may operate. For notational convenience, we suppress this matter.) Using the resulting vectors for two distinct challenges x—and the representations $P = \mathbf{A} \cdot \mathbf{S}^x \cdot \mathbf{g}^{-z} \cdot \mathbf{h}'^{z \cdot \mathbf{y}^{2\cdot n} + z^2 \cdot (\mathbf{2}^n \| \mathbf{0}^n) + z^3 \cdot (\mathbf{0}^n \| \mathbf{2}^n)}$—$\mathcal{X}$ obtains openings

$\alpha, \mathbf{a}_L, \mathbf{a}_R$ and $\rho, \mathbf{s}_L, \mathbf{s}_R$ of A and S (respectively). If, for any x, equalities of the form of lines 71 and 72 do *not* hold, $\mathcal{X}$ returns the corresponding binding violation of P.

Otherwise, we observe that necessarily $\hat{t} = t(X)$ for each x, where $t(X) := \langle l(X), r(X) \rangle$ (and $l(X)$ and $r(X)$ are defined as in 61 and 62 with respect to the above openings). By constructing a Vandermonde matrix in the challenges x and multiplying its inverse by the vector of values $(\hat{t}, \tau_x)$, $\mathcal{X}$ obtains, in view of (4), openings for $g^{b+\delta(y,z)}$ as well as for T_1 and T_2. Barring a second binding violation, we conclude in fact that $b + \delta(y,z) = t_0$ for each x. Indeed, both sides of this equality are (in this case) uniquely characterized as the first component of that vector whose image equals $\hat{t}$ under each "evaluation-at-x" linear functional.

We conclude in particular from the definitions of $t(X)$ and $\delta(y,z)$ that:

$$b = t_0 - \delta(y,z) = \left\langle \mathbf{a}_L, \mathbf{y}^{2 \cdot n} \circ \mathbf{a}_R \right\rangle + \left\langle \mathbf{a}_L - \mathbf{1}^{2 \cdot n} - \mathbf{a}_R, \mathbf{y}^{2 \cdot n} \right\rangle \cdot z + \left\langle \mathbf{a}_{L[:n]}, \mathbf{2}^n \right\rangle \cdot z^2 + \left\langle \mathbf{a}_{L[n:]}, \mathbf{2}^n \right\rangle \cdot z^3$$

for each w, y and z. Exponentiating both sides of this equality by g (and also multiplying them by w^m), we derive the following alternative expression for the right-hand side of (3):

$$g^{w^m \cdot \left\langle \mathbf{a}_L, \mathbf{y}^{2 \cdot n} \circ \mathbf{a}_R \right\rangle} \cdot \left(g^{w^m \cdot \left\langle \mathbf{a}_L - \mathbf{1}^{2 \cdot n} - \mathbf{a}_R, \mathbf{y}^{2 \cdot n} \right\rangle} \right)^z \cdot \left(g^{w^m \cdot \left\langle \mathbf{a}_{L[:n]}, \mathbf{2}^n \right\rangle} \right)^{z^2} \cdot \left(g^{w^m \cdot \left\langle \mathbf{a}_{L[n:]}, \mathbf{2}^n \right\rangle} \right)^{z^3}.$$

We conclude immediately (i.e., after inverting a 4×4 Vandermonde matrix in the challenges z and applying it to both representations) that $\left\langle \mathbf{a}_L, \mathbf{y}^{2 \cdot n} \circ \mathbf{a}_R \right\rangle = 0$ and $\left\langle \mathbf{a}_L - \mathbf{1}^{2 \cdot n} - \mathbf{a}_R, \mathbf{y}^{2 \cdot n} \right\rangle = 0$, as well as that $g^{-w^m \cdot b^*} = \overline{D}^{-\widehat{\mathsf{sk}}} \cdot \overline{C_0}$ and $g^{w^m \cdot b'} = \overline{C_{Rn}}^{-\widehat{\mathsf{sk}}} \cdot \overline{C_{Ln}}$, where we denote $b^* := \left\langle \mathbf{a}_{L[:n]}, \mathbf{2}^n \right\rangle$ and $b' := \left\langle \mathbf{a}_{L[n:]}, \mathbf{2}^n \right\rangle$. From the first two equalities at all $2 \cdot n$ challenges y, we conclude that $\mathbf{a}_L \in \{0,1\}^{2 \cdot n}$. It follows immediately that b^* and b' reside in $\{0, \ldots, \mathsf{MAX}\}$ (from the construction of these latter values).

$\mathcal{X}$ continues as follows. By running the extractor of Lemma 4.2 on three values w, $\mathcal{X}$ obtains openings $b_{0,0}, \ldots, b_{1,m-1}$, $a_{0,0}, \ldots, a_{1,m-1}$ of the initial commitments B and A for which $b_{\iota,k} \in \{0,1\}$. If for *any* w it holds that the response $f_{\iota,k} \neq b_{\iota,k} \cdot w + a_{\iota,k}$ for some ι, k, $\mathcal{X}$ returns the corresponding binding violation of $B^w A$. Otherwise, it uses the bits $b_{\iota,0}, \ldots, b_{\iota,m-1}$ to determine the witness l_ι (for each $\iota \in \{0,1\}$). It then uses the openings $b_{\iota,k}$ and $a_{\iota,k}$ to reconstruct the polynomials $P_{\iota,k}(W)$ (for each ι, k), and in particular representations, valid for each w, of the form:

$$\begin{aligned}
(\overline{y_0}, \overline{g}) &= \left(y_{l_0}^{w^m} \cdot \prod_{k=0}^{m-1} \widehat{y_{0,k}}^{-w^k}, g^{w^m} \cdot \prod_{k=0}^{m-1} \widetilde{g_k}^{-w^k} \right), \\
(\overline{C_0}, \overline{D}) &= \left(C_{l_0}^{w^m} \cdot \prod_{k=0}^{m-1} \widehat{C_{0,k}}^{-w^k}, D^{w^m} \cdot \prod_{k=0}^{m-1} \widetilde{D_k}^{-w^k} \right), \\
(\overline{C_{Ln}}, \overline{C_{Rn}}) &= \left(C_{Ln,l_0}^{w^m} \cdot \prod_{k=0}^{m-1} \widehat{C_{Ln,k}}^{-w^k}, C_{Rn,l_0}^{w^m} \cdot \prod_{k=0}^{m-1} \widehat{C_{Rn,k}}^{-w^k} \right),
\end{aligned}$$

for easily computable elements $\left(\widehat{y_{0,k}}, \widehat{C_{0,k}}, \widehat{C_{Ln,k}}, \widehat{C_{Rn,k}} \right)_{k=0}^{m-1}$ which don't depend on w.

From the equalities $g^{-w^m \cdot b^*} = \overline{D}^{-\widehat{\mathsf{sk}}} \cdot \overline{C_0}$, $g^{w^m \cdot b'} = \overline{C_{Rn}}^{-\widehat{\mathsf{sk}}} \cdot \overline{C_{Ln}}$, and $\overline{g}^{\widehat{\mathsf{sk}}} = \overline{y_0}$, $\mathcal{X}$ obtains in turn the relationships

$$\begin{aligned}
y_{l_0}^{w^m} \cdot \prod_{k=0}^{m-1} \widehat{y_{0,k}}^{-w^k} &= \left(g^{w^m} \cdot \prod_{k=0}^{m-1} \widetilde{g_k}^{-w^k} \right)^{\widehat{\mathsf{sk}}}, \\
\left(g^{b^*} \cdot C_{l_0} \right)^{w^m} \cdot \prod_{k=0}^{m-1} \widehat{C_{0,k}}^{-w^k} &= \left(D^{w^m} \cdot \prod_{k=0}^{m-1} \widetilde{D_k}^{-w^k} \right)^{\widehat{\mathsf{sk}}}, \\
\left(g^{-b'} \cdot C_{Ln,l_0} \right)^{w^m} \cdot \prod_{k=0}^{m-1} \widehat{C_{Ln,k}}^{-w^k} &= \left(C_{Rn,l_0}^{w^m} \cdot \prod_{k=0}^{m-1} \widehat{C_{Rn,k}}^{-w^k} \right)^{\widehat{\mathsf{sk}}}.
\end{aligned}$$

The discrete logarithms with respect to g of these equalities give three algebraic equations, with unknown coefficients, which each point $(w, \widehat{\mathsf{sk}})$ *simultaneously* satisfies:

$$\log(y_{l_0}) \cdot w^m - \sum_{k=0}^{m-1} \log(\widehat{y_{0,k}}) \cdot w^k = \left(1 \cdot w^m - \sum_{k=0}^{m-1} \log(\widetilde{g_k}) \cdot w^k\right) \cdot \widehat{\mathsf{sk}}, \tag{5}$$

$$\log(g^{b^*} \cdot C_{l_0}) \cdot w^m - \sum_{k=0}^{m-1} \log(\widehat{C_{0,k}}) \cdot w^k = \left(\log(D) \cdot w^m - \sum_{k=0}^{m-1} \log(\widetilde{D_k}) \cdot w^k\right) \cdot \widehat{\mathsf{sk}}, \tag{6}$$

$$\log(g^{-b'} \cdot C_{Ln,l_0}) \cdot w^m - \sum_{k=0}^{m-1} \log(\widehat{C_{Ln,k}}) \cdot w^k = \left(\log(C_{Rn,l_0}) \cdot w^m - \sum_{k=0}^{m-1} \log(\widehat{C_{Rn,k}}) \cdot w^k\right) \cdot \widehat{\mathsf{sk}}. \tag{7}$$

$\mathcal{X}$ views the $2m+1$ satisfying points $(w, \widehat{\mathsf{sk}})$ of (5) as an instance of [vzGG13, (21)] (using the parameters $n = 2m+1$, $k = m+1$), and in particular denotes by r and t the (unknown) polynomials (in the indeterminate W) which appear in (5)'s left- and right-hand sides. It constructs polynomials $r_j, t_j \in \mathbb{F}_q[W]$ as in the statement of [vzGG13, Cor. 5.18]; finally, it sets $\mathsf{sk} = \log(y_l)$ as $(\mathrm{lc}(t_j))^{-1} \cdot \mathrm{lc}(r_j)$.

We argue here exactly as in the proof of Theorem 5.3. Indeed, just as before, we claim from [vzGG13, Thm. 5.16, (ii)] that there exists some nonzero $\alpha \in \mathbb{F}_q[W]$ for which

$$(r, t) = (\alpha \cdot r_j, \alpha \cdot t_j), \tag{8}$$

and consequently that $\log(y_{l_0})$ can be expressed as $\tau^{-1} \cdot \mathrm{lc}(r_j)$ (where $\tau = \mathrm{lc}(t_j)$). In fact, we further observe that (8) holds for *any* polynomials r, t of appropriate degree which satisfy [vzGG13, (21)] (though in general with different α). In particular, from (6) we deduce the existence of some α as in (8), whose leading coefficient moreover must equal $\log(D) \cdot \tau^{-1}$; we conclude that $\log(g^{b^*} \cdot C_{l_0}) = \log(D) \cdot \tau^{-1} \cdot \mathrm{lc}(r_j) = \log(D) \cdot \log(y_{l_0})$. Similarly, from (7) we see that that $\log(g^{-b'} \cdot C_{Ln,l_0}) = \log(C_{Rn,l_0}) \cdot \log(y_{l_0})$. These latter facts imply (respectively) that $C_{l_0} = g^{-b^*} D^{\mathsf{sk}}$ and $C_{Ln,l_0} = g^{b'} C_{Rn,l_0}^{\mathsf{sk}}$, as required by (2).

We continue with the operation of $\mathcal{X}$. For each assignment of values to the challenges v, w, $\mathcal{X}$ uses the equality $A_D = g^{s_r} \cdot D^{-c}$ for two values c (and arbitrary y, z, x) to obtain an element r for which $g^r = D$. Exactly as in the proof of Theorem 4.6, it constructs using the $P_{0,k}(W)$ and $P_{1,k}(W)$ an expression:

$$(\overline{C_X}, \overline{y_X}) = \left(\left(\prod_{\iota,j=0}^{1,\frac{N}{2}-1} C_{l_\iota+2\cdot j}^{\xi_{2\cdot j+\iota}} \right)^{w^m} \cdot \prod_{k=0}^{m-1} \widehat{C_{X,k}}^{-w^k}, \left(\prod_{\iota,j=0}^{1,\frac{N}{2}-1} y_{l_\iota+2\cdot j}^{\xi_{2\cdot j+\iota}} \right)^{w^m} \cdot \prod_{k=0}^{m-1} \widehat{y_{X,k}}^{-w^k} \right),$$

where $(\xi_0, \xi_1, \xi_2, \xi_3, \ldots, \xi_{N-1}) = \left(1, 1, v, v^2, \ldots, v^{N-2}\right)$, and the elements $\left(\widehat{C_{X,k}}, \widehat{y_{X,k}}\right)_{k=0}^{m-1}$ don't depend on w. From the verification equation $A_X = \overline{y_X}^{s_r} \cdot \overline{C_X}^{-c}$, we observe moreover that $\overline{C_X} = \overline{y_X}^r$, and hence that:

$$\left(\prod_{\iota,j=0}^{1,\frac{N}{2}-1} C_{l_\iota+2\cdot j}^{\xi_{2\cdot j+\iota}} \right)^{w^m} \cdot \prod_{k=0}^{m-1} \widehat{C_{X,k}}^{-w^k} = \left(\left(\prod_{\iota,j=0}^{1,\frac{N}{2}-1} y_{l_\iota+2\cdot j}^{\xi_{2\cdot j+\iota}} \right)^{w^m} \cdot \prod_{k=0}^{m-1} \widehat{y_{X,k}}^{-w^k} \right)^r.$$

For each v, $\mathcal{X}$ uses copies of this equation for $m+1$ values w (i.e., by inverting a Vandermonde matrix in w and reading off its bottom row) to derive the equality $\left(\prod_{\iota,j=0}^{1,\frac{N}{2}-1} C_{l_\iota+2\cdot j}^{\xi_{2\cdot j+\iota}}\right)^r = \prod_{\iota,j=0}^{1,\frac{N}{2}-1} y_{l_\iota+2\cdot j}^{\xi_{2\cdot j+\iota}}$. Finally, using copies of this equation for $N-1$ values v (and inverting a final Vandermonde matrix, in v), $\mathcal{X}$ obtains the individual equalities $(y_{l_0} \cdot y_{l_1})^r = C_{l_0} \cdot C_{l_1}$ and $\{(y_{l_\iota+2\cdot j})^r = C_{l_\iota+2\cdot j}\}_{\iota,j=0,1}^{1,\frac{N}{2}-1}$.

From the equality $F^w E = \mathsf{Com}\left((f_{0,0} \cdot f_{1,0}, (w - f_{0,0})(w - f_{1,0})); z_E\right)$ at three values w, we argue that the (*a priori* quadratic) polynomials $F_{0,0,1}(W) \cdot F_{1,0,1}(W)$ and $(W - F_{0,0,1}(W)) \cdot (W - F_{1,0,1}(W))$ are both in fact linear in W, and hence that their leading coefficients—namely, $b_{0,0} \cdot b_{1,0}$ and $(1 - b_{0,0}) \cdot (1 - b_{1,0})$, respectively—are both 0. This latter fact encodes exactly that the least-significant bits $b_{0,0}$ and $b_{1,0}$, respectively, of l_0 and l_1 satisfy $b_{0,0} \wedge b_{1,0}$ and $\neg b_{0,0} \wedge \neg b_{1,0}$, or in other words $b_{0,0} \oplus b_{1,0} = 1$, and hence that $l_0 \not\equiv l_1 \mod 2$. This in turn, together with the equalities $\{(y_{l_\iota+2\cdot j})^r = C_{l_\iota+2\cdot j}\}_{\iota,j=0,1}^{1,\frac{N}{2}-1}$ obtained above (and a re-indexing), implies finally the equalities $\{y_i^r = C_i\}_{i\notin\{l_0,l_1\}}$ required by (2). This completes the extraction process. $\square$

Remark B.2. This soundness property, alongside its role in the proof of Theorem 6.4, implies in addition the *a priori* untrue fact that $\mathsf{Crypt}_{\mathcal{A},\Pi}$ is *well-defined*, in that $\mathcal{A}$'s intermediate $\mathsf{Insert}(\mathsf{tx})$ calls can't mangle honest (or even corrupt) accounts $y_i \in S$, and hence that each $\mathsf{Transact}(\cdot,\cdot,\cdot,\cdot)$ query is guaranteed to yield a valid transaction. More precisely, any $\mathcal{A}$ for which these properties fail in non-negligibly many executions of $\mathsf{Crypt}_{\mathcal{A},\Pi}$ can be converted into an adversary $\mathcal{A}'$ who successfully attacks $\mathsf{Binding}_{\mathcal{A}',\mathsf{Com}}$ with respect to the Pedersen scheme. We leave the details of this reduction to the reader.

C Ledger Indistinguishability: Proof

Proof of Theorem 6.5. For any fixed adversary $\mathcal{A}$ attacking $\mathsf{L\text{-}IND}_{\mathcal{A},\Pi}$, we define an algorithm $\mathcal{A}'$ for $\mathsf{DDH}_{\mathcal{A}',\mathcal{G}}(\lambda)$. Our construction is analogous to that of Theorem 5.6.

$\mathcal{A}'$ is defined as follows. It is given the inputs $\mathbb{G}$, q, g, h_1, h_2, and h'.

1. Generate parameters $\sigma \leftarrow \mathsf{Setup}\left(1^\lambda\right)$ for which $\mathbb{G}$, q, and g are as given by the experiment input. Give σ to $\mathcal{A}$.

2. Given the list $(b_i)_{i=0}^{N-1}$, generate a keypair $(y_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\lambda)$ for each $i \in \{0,\ldots,N-1\}$. For a randomly chosen index $l \in \{0,\ldots,N-1\}$, overwrite $y_l := h_1$. Encrypt $\mathsf{acc}[y_i] := \mathsf{Enc}_{y_i}(b_i)$ for each i and initialize $\mathcal{O}_{\mathsf{SC}}$ with acc. Finally, give the modified list $S = (y_i)_{i=0}^{N-1}$ to $\mathcal{A}$.

3. Respond to each of $\mathcal{A}$'s random oracle queries with a random element of $\mathbb{F}_q$.

4. For each oracle query $\mathsf{Transact}(s,\overline{y},R,b^*)$ for which $s \neq l$, simply compute $\mathsf{tx} := \left((C_i,y_i)_{i=0}^{N-1}, D, \pi\right) \leftarrow \mathsf{Trans}(\mathsf{acc},\mathsf{sk}_s,\overline{y},R,b^*)$ as specified by the Anonymous Zether protocol. If instead $s = l$, construct π by replacing each Schnorr protocol involving sk by a simulation. More specifically, randomly generate c and s_{sk}, and set $A_y := \overline{g}^{s_{\mathsf{sk}}} \cdot \overline{y_0}^{-c}$, as well as $A_b := g^{s_b} \cdot \left(\overline{D}^{-z^2} \cdot \overline{C_{Rn}}^{z^3}\right)^{s_{\mathsf{sk}}} \cdot \left(\overline{C_0}^{-z^2} \cdot \overline{C_{Ln}}^{z^3}\right)^{-c}$. Compute all other elements as specified by the protocol.

5. For each oracle query $\mathsf{Insert}(\mathsf{tx})$, forward tx to $\mathcal{O}_{\mathsf{SC}}$.

6. When $\mathcal{A}$ outputs $s_0, s_1, r_0, r_1, b_0^*, b_1^*$, and R^* (and if the conditions of step 5. hold), choose a uniform bit $b \in \{0,1\}$. If $s_b \neq l$, abort and return a random bit.

7. If on the other hand $s_b = l$, proceed as follows. Having obtained from $\mathcal{A}$ secret keys sk_i^* for each $y_i^* \in R^*$, assume without loss of generality that $R^* \subset S$, and in fact that $R^* = S$. (This assumption just simplifies notation.) Assign $D := h_2$, and construct $((C_i,y_i)_{i=0}^{N-1}, D)$ with the aid of the sk_i (that is, set $C_i := (h_2)^{\mathsf{sk}_i}$, multiplying in addition by $g^{-b_b^*}$ if $i = s_b$ and by $g^{b_b^*}$ if $g^{\mathsf{sk}_i} = \overline{y}_b$). During the construction of the proof π, perform the replacements:

$$\left(\widetilde{y_{0,k}}, \widetilde{g_k}\right) := \left(\prod_{i=0}^{N-1} y_i^{P_{0,i,k}} \cdot (h')^{\psi_k}, (h_2)^{\psi_k}\right),$$

$$\left(\widetilde{C_{0,k}}, \widetilde{D_k}\right) := \left(\prod_{i=0}^{N-1} C_i^{P_{0,i,k}} (h')^{\chi_k}, (h_2)^{\chi_k}\right),$$

$$\left(\widetilde{C_{Ln,k}}, \widetilde{C_{Rn,k}}\right) := \left(\prod_{i=0}^{N-1} C_{Ln,i}^{P_{0,i,k}} \cdot (h')^{\phi_k}, \prod_{i=0}^{N-1} C_{Rn,i}^{P_{0,i,k}} \cdot (h_2)^{\phi_k}\right),$$

   for each $k \in \{0,\ldots,m-1\}$. Finally, the Schnorr protocols involving s_{sk}, as above.

8. When $\mathcal{A}$ outputs b', return whether $b' \stackrel{?}{=} b$.

For notational clarity, we introduce a modified experiment $\mathsf{L\text{-}IND}'_{\mathcal{A},\Pi}$, designed to remove $\mathcal{A}'$'s tendency to abort from the analysis. $\mathsf{L\text{-}IND}'_{\mathcal{A},\Pi}$ differs from $\mathsf{L\text{-}IND}_{\mathcal{A},\Pi}$ *only* in the construction strategy of π. In particular, the experimenter, upon receiving $s_0, s_1, \overline{y}_0, \overline{y}_1, b_0^*, b_1^*$, and R, checking the conditions of 5., and generating $b \in \{0, 1\}$, generates a further random bit $b'' \in \{0, 1\}$. If $b'' = 1$, the experimenter proceeds exactly as in $\mathsf{L\text{-}IND}_{\mathcal{A},\Pi}$. Otherwise, the experimenter, after generating $D := h_2$ and h' randomly, proceeds exactly as in step 7. above (i.e., using these elements, as opposed to the DDH challenge elements).

We first analyze $\mathcal{A}$'s advantage in $\mathsf{L\text{-}IND}'_{\mathcal{A},\Pi}$. If $b'' = 1$, this advantage exactly equals that of $\mathcal{A}$ in $\mathsf{L\text{-}IND}_{\mathcal{A},\Pi}$, by construction of the former experiment. We claim that if $b'' = 0$, $\mathcal{A}$'s probability of guessing $b' = b$ at most negligibly exceeds $\frac{1}{2}$. To make this explicit, we define a further experiment $\mathsf{L\text{-}IND}''_{\mathcal{A},\Pi}$ representing the case $b'' = 0$ of $\mathsf{L\text{-}IND}'_{\mathcal{A},\Pi}$; that is, the experimenter *always* assigns h_2 and h' randomly, and proceeds as in step 7. We claim that $\Pr[\mathsf{L\text{-}IND}''_{\mathcal{A},\Pi}(\lambda) = 1] - \frac{1}{2} \leq \mathsf{negl}(\lambda)$ for some negligible function negl, a fact whose proof we defer to a lemma below.

Assuming this result for now, we observe finally that $\mathcal{A}$'s view in its simulation by $\mathcal{A}'$ exactly matches its view in $\mathsf{L\text{-}IND}'_{\mathcal{A},\Pi}$, provided that $\mathcal{A}'$ doesn't abort (i.e., if $s_b = l$) and no random oracle inconsistencies occur during $\mathcal{A}'$'s $\mathsf{Transact}(\cdot, \cdot, \cdot, \cdot)$ simulations. The latter event happens in negligibly many executions of $\mathcal{A}'$ (say negl''); among the remaining executions, $\mathcal{A}'$ tendency to abort impacts exactly $\frac{1}{N}$ of those execution paths in which $\mathcal{A}$ "wins" (i.e., satisfies the winning condition of $\mathsf{L\text{-}IND}'_{\mathcal{A},\Pi}$).

Putting these facts together, we conclude that:

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

This completes the proof. □

It thus remains to prove the lemma below. To this end, we invoke the *multi-recipient, randomness-reusing* encryption experiment of [BBS03, Def. 4.1], and in particular its specialization to the multi-recipient, randomness-reusing El Gamal scheme $\overline{\Pi} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. For convenience, we reproduce this specialization below. We give also to the adversary an LR-oracle for the "both sides El Gamal" experiment sketched in Example 3.8 above, under an unrelated key (see step 4. below). Explicitly:

Definition C.1 (Bellare–Boldyreva–Staddon [BBS03, Def. 4.1])**.** The *multi-recipient, randomness-reusing El Gamal experiment* $\mathsf{RR\text{-}MREG}^{N(\cdot)}_{\mathcal{A},\mathcal{G}}(\lambda)$ is defined as:

1. Parameters $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^\lambda)$ are generated and given to $\mathcal{A}$.
2. $\mathcal{A}$ outputs an integer $N \leq N(\lambda)$.
3. Keypairs $(y_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\lambda)$ for $i \in \{0, \ldots, N-1\}$ are generated, and $(y_i)_{i=0}^{N-1}$ is given to $\mathcal{A}$. A uniform bit $b \leftarrow \{0, 1\}$ is chosen.
4. An extra key y^* is also generated. $\mathcal{A}$ is given access to y^* and to an oracle $\mathsf{LR}_{b,y^*}(\cdot, \cdot)$, where $\mathsf{LR}_{b,y^*}((M_0, m_0), (M_1, m_1))$ returns $(M_b \cdot (y^*)^r, m_b \cdot g^r)$ for a fresh randomly generated element $r \leftarrow \mathbb{F}_q$.
5. At any point during the experiment, $\mathcal{A}$ outputs vectors $(m_{0,i})_{i=0}^{N-1}$ and $(m_{1,i})_{i=0}^{N-1}$, together with a vector $(m_i)_{i=N}^{N(\lambda)-1}$ and additional keypairs $(y_i, \mathsf{sk}_i)_{i=N}^{N(\lambda)-1}$. For (fresh) random $r \leftarrow \mathbb{F}_q$, $\mathcal{A}$ is given $D := g^r$, as well as, for each $i \in \{0, \ldots, N(\lambda) - 1\}$, the element $C_i := m_i \cdot D^{\mathsf{sk}_i}$ (where $m_i := m_{b,i}$).
6. $\mathcal{A}$ outputs a bit b'. The output of the experiment is defined to be 1 if and only if $b' = b$.

We say that multi-recipient, randomness-reusing El Gamal is *secure* if, for each PPT adversary $\mathcal{A}$, there exists a negligible function negl for which $\Pr[\mathsf{RR\text{-}MREG}^{N(\cdot)}_{\mathcal{A}}(\lambda) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$.

From the DDH assumption (see [KL15, Thm. 11.18]), and the "reproducibility" of El Gamal encryption [BBS03, Lem. 7.1], we conclude from [BBS03, Thm. 6.2] that multi-recipient, randomness-reusing El Gamal is secure (i.e., under the DDH assumption).

We turn to the remaining claim:

Lemma C.2. *If the DDH problem is hard with respect to $\mathcal{G}$, then* $\Pr[\mathsf{L\text{-}IND}''_{\mathcal{A},\Pi}(\lambda) = 1] \le \frac{1}{2} + \mathsf{negl}(\lambda)$*, for some negligible function* negl.

Proof. We fix an arbitrary adversary $\mathcal{A}$ attacking $\mathsf{L\text{-}IND}''_{\mathcal{A},\Pi}$; we denote by $N(\cdot)$ a polynomial upper bound on the *sum* of the sizes of $\mathcal{A}$'s initial list $(b_i)_{i=0}^{N-1}$ and of the ring R^* of step 5. (where both sizes are viewed as functions of the security parameter λ). We define an algorithm $\mathcal{A}''$ attacking $\mathsf{RR\text{-}MREG}^{N(\cdot)}_{\mathcal{A}'',\mathcal{G}}$ (i.e., its modified version given above) as follows. It is given inputs $\mathbb{G}, q, g$.

1. Construct parameters σ which are compatible with the inputs $\mathbb{G}, q, g$, and give σ to $\mathcal{A}$.
2. When $\mathcal{A}$ outputs $(b_i)_{i=0}^{N-1}$, output N. Upon receiving $S := (y_i)_{i=0}^{N-1}$, compute $\mathsf{acc}[y_i] := \mathsf{Enc}_{y_i}(b_i)$ for each y_i (i.e., using standard El Gamal) and initialize $\mathcal{O}_{\mathsf{SC}}$ with acc. Give S to $\mathcal{A}$.
3. Respond to each of $\mathcal{A}$'s random oracle queries with a random element of $\mathbb{F}_q$.
4. For each oracle query $\mathsf{Transact}(s, \overline{y}, R, b^*)$, construct tx as follows. Use the implicit "CPA oracle" of multi-recipient El Gamal to construct the statement $((C_i, y_i)_{i=0}^{N-1}, D)$. When constructing the proof π, replace all Schnorr protocols involving sk by simulations, exactly as specified in 4. above.
5. For each oracle query $\mathsf{Insert}(\mathsf{tx})$, forward tx to $\mathcal{O}_{\mathsf{SC}}$.
6. When $\mathcal{A}$ outputs $s_0, s_1, \overline{y}_0, \overline{y}_1, b_0^*, b_1^*$, and R^*, together with secret keys sk_i for those $y_i \in R^* \backslash S$ (and if the conditions of step 5. hold), proceed as follows. After extending R^* with any unused elements of S, as well as possibly generating extra keypairs (y_i, sk_i), and finally re-ordering elements, assume (without loss of generality) that R^* takes the form $(y_0, \ldots, y_{N-1}, y_N, \ldots, y_{N(\lambda)-1})$ (i.e., where y_i for $i \in \{N, \ldots, N(\lambda) - 1\}$ are adversarially generated). Initialize empty (i.e., identity-element) vectors $(m_{0,i})_{i=0}^{N-1}$, $(m_{1,i})_{i=0}^{N-1}$, and $(m_i)_{i=N}^{N(\lambda)-1}$. Set $m_{0,s_0} := g^{-b_0^*}$ and $m_{1,s_1} := g^{-b_1^*}$. If $\overline{y}_0 \in S$ and $\overline{y}_1 \in S$, set $m_{0,r_0} := g^{b_0^*}$ and $m_{1,r_1} := g^{b_1^*}$, where r_0 and r_1 are the indices in S of $\overline{y}_0$ and $\overline{y}_1$, respectively. Otherwise, set $m_r := g^{b^*}$, where $r \in \{N, \ldots, N(\lambda) - 1\}$ is the index of $\overline{y} := \overline{y}_0 = \overline{y}_1$ in R^* and $b^* := b_0^* = b_1^*$ (we use 5.(ii) here).
7. Simulate elements $\mathbf{A}, \mathbf{S}, y, z, T_1, T_2, x, \hat{t}, \mu$ using the Bulletproofs SHVZK simulator [BBB+18, §3]. Simulate the elements $A, B, C, D, E, F, v, w, (f_{\iota,k})_{\iota,k=0}^{1,m-1}$ using (an obvious extension of) the SHVZK simulator of Lemma 4.2. (Note that, as in Appendix A, two executions of Fig. 3, corresponding respectively to the indices $\iota \in \{0, 1\}$, are combined.)

   Using the simulated quantities $w, (f_{0,k})_{k=0}^{m-1}$ and the candidate indices s_0 and s_1, construct exactly as in step 3. of Theorem 5.6 above degree-m polynomials $P_{0,0,i}(X) = \sum_{k=0}^{m-1} P_{0,0,i,k} \cdot X^k$ and $P_{1,0,i}(X) = \sum_{k=0}^{m-1} P_{1,0,i,k} \cdot X^k$ (for $i \in \{0, \ldots, N(\lambda) - 1\}$). Similarly, use $(f_{1,k})_{k=0}^{m-1}$ and the recipient indices r_0 and r_1 defined above to construct $P_{0,1,i}(X)$ and $P_{1,1,i}(X)$.

   Assume without loss of generality the existence of $\log(N(\lambda))$ *further* unused honest keys in S, say $(\widetilde{y_{X,k}})_{k=0}^{m-1}$. (Tacitly, we replace $N(\lambda)$ with $N(\lambda) + \log(N(\lambda))$; to keep our notation reasonable, we avoid making this explicit.) For each $k \in \{0, \ldots, m - 1\}$, over the index corresponding to the key $\widetilde{y_{X,k}}$, write into the respective message vectors $(m_{0,i})_{i=0}^{N-1}$ and $(m_{1,i})_{i=0}^{N-1}$ the elements $\prod_{\iota,j=0}^{1,\frac{N}{2}-1} \left(g^{b_0^* \cdot \left(-P_{0,\iota,s_0-2\cdot j,k} + P_{0,\iota,r_0-2\cdot j,k}\right)}\right)^{\xi_{2\cdot j+\iota}}$ and $\prod_{\iota,j=0}^{1,\frac{N}{2}-1} \left(g^{b_1^* \cdot \left(-P_{1,\iota,s_1-2\cdot j,k} + P_{1,\iota,r_1-2\cdot j,k}\right)}\right)^{\xi_{2\cdot j+\iota}}$.
8. Output the (extended) vectors $(m_{0,i})_{i=0}^{N-1}$ and $(m_{1,i})_{i=0}^{N-1}$, along with $(m_i)_{i=N}^{N(\lambda)-1}$, and $(y_i, \mathsf{sk}_i)_{i=N}^{N(\lambda)-1}$. In this way, obtain shared-randomness encryptions $(C_i, D)_{i=0}^{N(\lambda)-1}$ under $(y_i)_{i=0}^{N(\lambda)-1}$; from the extensions defined above, obtain "encryptions" $(\widetilde{C_{X,k}}, D)_{k=0}^{m-1}$ (let's say) under the "public keys" $(\widetilde{y_{X,k}})_{k=0}^{m-1}$.
9. By submitting tuples $\left((\prod_{i=0}^{N-1} y_i^{P_{0,0,i,k}}, \mathrm{id}), (\prod_{i=0}^{N-1} y_i^{P_{1,0,i,k}}, \mathrm{id})\right)$ to $\mathsf{LR}_{y^*,b}$, obtain standard El Gamal encryptions $(\widetilde{y_{0,k}}, \widetilde{g_k})$ under y^* (i.e., for each $k \in \{0, \ldots, m-1\}$). Similarly, by submitting the tuples

$\Big((\prod_{i=0}^{N-1} C_i^{P_{0,0,i,k}}, \mathrm{id}), (\prod_{i=0}^{N-1} C_i^{P_{1,0,i,k}}, \mathrm{id})\Big)$, obtain standard encryptions $(\widetilde{C_{0,k}}, \widetilde{D_k})$ under y^*. Finally, submit tuples

$$\left(\Big(\prod_{i=0}^{N-1} C_{Ln,i}^{P_{0,0,i,k}}, \prod_{i=0}^{N-1} C_{Rn,i}^{P_{0,0,i,k}}\Big), \Big(\prod_{i=0}^{N-1} C_{Ln,i}^{P_{1,0,i,k}}, \prod_{i=0}^{N-1} C_{Rn,i}^{P_{1,0,i,k}}\Big)\right),$$

for $k \in \{0, \ldots, m-1\}$, to obtain the "both-sides encryptions" $(\widetilde{C_{Ln,k}}, \widetilde{C_{Rn,k}})$. In particular, these responses determine the intermediate quantities $\overline{y_0}, \overline{g}, \overline{C_0}, \overline{D}, \overline{C_{Ln}}, \overline{C_{Rn}}$.

Simulate all Σ-protocols using the Σ-Bullets simulator [BAZB20, §G]. In particular, simulate all Schnorr protocols involving sk, exactly as in $\mathcal{A}'$ above. Simulate those involving s_r, by randomly generating s_r, and setting $A_D := g^{s_r} \cdot D^{-c}$, as well as $A_X := \overline{y_X}^{s_r} \cdot \overline{C_X}^{-c}$. Randomly generate s_b and s_τ, and assign $A_t := g^{w^m \cdot c \cdot (\hat{t} - \delta(y,z))} \cdot h^{s_\tau} \cdot \left(T_1^x \cdot T_2^{x^2}\right)^{-w^m \cdot c} \cdot g^{-s_b}$. Submit the resulting transaction $\mathsf{tx} := ((C_i, y_i)_{i=0}^{N(\lambda)-1}, D, \pi)$ to $\mathcal{O}_{\mathsf{SC}}$.

10. When $\mathcal{A}$ outputs a bit b', return the bit b'.

To simplify our analysis, we specialize all commitments to the Pedersen scheme, which is perfectly hiding. We note now that $\mathcal{A}$'s view in its simulation by $\mathcal{A}''$ above *exactly* matches its view in the experiment $\mathsf{L\text{-}IND}''_{\mathcal{A},\Pi}$, provided that all Schnorr simulations succeed (we use the perfect SHVZK of Bulletproofs, of many-out-of-many proofs, and of Σ-Bullets). Finally, $\mathcal{A}''$ wins $\mathsf{RR\text{-}MREG}^{N(\cdot)}_{\mathcal{A}'',\mathcal{G}}$ whenever $\mathcal{A}$ "wins" $\mathsf{L\text{-}IND}''_{\mathcal{A},\Pi}$ (i.e., chooses the right bit). This observation completes the proof. $\square$