Anonymous Zether: Technical Report

Benjamin E. Diamond
J.P. Morgan

Abstract

We describe a protocol for the *anonymous Zether* payment system proposed by Bünz, Agrawal, Zamani, and Boneh [BAZB]. We first discuss "Σ-Bullets"—which adapt *Bulletproofs* to the setting of ElGamal ciphertexts—and address shortcomings in the approach of [BAZB].

We further study the *anonymous* extension discussed in [BAZB, §D]. Extending Groth and Kohlweiss's *one out of many* proofs [GK15], we prove knowledge of a secret permutation (of a certain form) of ElGamal ciphertexts with heterogeneous keys. We finally give an efficient implementation based on the number-theoretic transform.

Introduction

Zether is a cryptographic protocol for confidential payment, described in a manuscript of Bünz, Agrawal, Zamani and Boneh [BAZB]. In contrast to well-known protocols like Monero and Zcash, Zether is "account-based", and features constant long-term space overhead per participant (though see also Quisquis [FMMO]); Zether also lacks a trusted setup.

The manuscript [BAZB] focuses on *basic Zether*, in which account balances and transfer amounts are concealed, but participants' identities are not. Its central cryptographic technique, called "Σ-Bullets", seeks to adapt Bünz, Bootle, Boneh, Poelstra, Wuille and Maxwell's *Bulletproofs* [BBB$^+$] (originally designed for Pedersen commitments) to the setting of ElGamal ciphertexts. We argue in Section 2 below that the Σ-Bullets protocol of [BAZB] has security flaws, and describe concrete attacks on it (targeting both soundness and zero-knowledge). We also propose a secure amendment.

In Section 3, we turn to *anonymous* payment, as proposed in [BAZB, §D]. The appendix [BAZB, §D] suggests a relation, but no proof protocol; our work proposes one. We extend *one out of many proofs*, introduced by Groth and Kohlweiss [GK15], so as to prove knowledge not just of a secret index but of a secret *permutation* (of a certain form) of ElGamal ciphertexts; we further allow ciphertexts encrypted under heterogeneous keys. Our protocol conducts a variant of basic Zether on these *permuted* (re-encrypted) ciphertexts. Our security proofs show, among other things, that knowledge of discrete logarithm relations on these permuted ciphertexts carries over to the originals (a technique which may be of independent interest).

Our protocol is efficient, with $\mathcal{O}(\log N)$-sized proofs and $\mathcal{O}(N \log N)$ runtime complexity for both the prover and the verifier (on an "anonymity set" of size N). This runtime, in particular, exploits a novel adaption of the number-theoretic transform to elliptic curve points. We describe an implementation of our system, and report on its performance.

1 Review of Basic and Anonymous Zether

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

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

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

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

1.1 Basic Zether

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

Finally, the prover should prove knowledge of:

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

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

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

1.2 Anonymous Zether

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

Finally, the prover should prove knowledge of:

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

- b^* and b^* in $\{0,\ldots,\mathsf{MAX}\}$ for which $C_{\mathsf{idx}_0} = g^{b^*} \cdot D$ and $C_{Ln,\mathsf{idx}_0} = g^{b'} \cdot C_{Rn,\mathsf{idx}_0}$ (overflow and overdraft protection).

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

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

The requirement that $\mathsf{idx}_0 \not\equiv \mathsf{idx}_1 \mod 2$ decreases privacy, but not by much. Indeed, this requirement decreases the cardinality of the set of possible pairs $(\mathsf{idx}_0, \mathsf{idx}_1) \in \{0,\ldots,N-1\}^2$ from N^2 to $\frac{N^2}{2}$; this latter cardinality of course still grows quadratically in N.

2 Bulletproofs and ElGamal Ciphertexts

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

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

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

2.1 Attacks on [BAZB]

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

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

2.2 A refined "Σ-Bullets"

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

We informally describe our amended version; a detailed protocol (which also includes anonymity) can be found in Section 5 below. Alongside the initial commitments A and S, the prover should publish additional ciphertexts (C', D') and (C'_{Ln}, C'_{Rn}) which encrypt h^{γ^*} and $h^{\gamma'}$ respectively (for randomly chosen scalars γ^* and γ'). Upon receiving x, the prover should add to τ_x the constant term $z^2 \cdot \gamma^* + z^3 \cdot \gamma'$. During the sigma protocols, in addition to proving knowledge of sk for which $g^{\mathsf{sk}} = y$, the prover should *also* prove knowledge of b^*, b', γ^*, and γ' for which:

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

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

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

3 Anonymous Payments

3.1 One-out-of-many proofs and vector rotations

We now discuss our approach to anonymity. We begin with *one out of many proofs*, introduced by Groth and Kohlweiss [GK15] and extended by Bootle, Cerulli, Chaidos, Ghadafi, Groth, and Petit [BCC+15]. These techniques serve to prove knowledge—given a fixed list of commitments—of a secret element among this list and an opening for this element to the message 0.

In fact, these techniques admit more flexibility than is explicitly described in [GK15] and [BCC+15]. An *intermediate* step of these protocols entails the construction of a "re-encryption" of the secret list element (same message but new randomness); in the final step, the prover simply reveals this re-encryption's randomness (the verifier may check explicitly whether it opens to 0). Instead of revealing the re-encryption's randomness, however, the prover may use it in further protocols. This idea will be key in our work.

The first challenge is that [GK15] and [BCC+15] require homomorphic commitment schemes, like Pedersen commitments or ElGamal encryptions under *the same* key. Using minor adjustments, we extend these protocols so as to support lists of ElGamal encryptions belonging to *arbitrary* public keys.

The most significant challenge of [GK15] and [BCC+15] is that only *one* re-encryption is delivered, whereas we want something more like a permutation. We denote by idx_0 and idx_1 the sender's and receiver's indices in what follows. Informally, we want to first run [BCC+15] twice, so as to deliver to the verifier re-encryptions of (C_{idx_0}, D) and (C_{idx_1}, D) (while concealing these elements' indices in the original list); using fairly simple extensions, we may also deliver a re-encryption of $(C_{Ln,\mathsf{idx}_0}, C_{Rn,\mathsf{idx}_0})$ and "re-encryptions" (i.e., exponentiations) of y_{idx_0} and y_{idx_1}, where the secrets idx_0 and idx_1 are provably chosen consistently throughout. These re-encrypted values in hand, the prover and verifier may conduct basic Zether on them. (That the protocol remains sound even when conducted on *re-encryptions* is non-trivial, but true, as we show below.)

All this says nothing, however, about the remaining equalities $y_i^r = C_i$ for $i \notin \{\mathsf{idx}_0, \mathsf{idx}_1\}$, which present a serious difficulty. A naïve approach would essentially conduct [BCC+15] N times, adopting certain additional modifications to ensure that the secret index is chosen distinctly each time. These modifications—at least as we are able to see—are incompatible with the logarithmic optimizations of [GK15], however, so that this approach would demand $\mathcal{O}(N^2)$ communication ($\mathcal{O}(N \log N)$ communication would *perhaps* be possible, though non-trivial).

We reduce the communication to $\mathcal{O}(\log N)$ using the following trick, which necessitates that we further deconstruct the protocols [GK15] and [BCC+15]. A key step in these protocols is the definition

of a vector $(p_i(X))_{i=0}^{N-1}$ of polynomials, and the delivery to the verifier of their *evaluations* $(p_i(x))_{i=0}^{N-1}$ at a challenge x. Because $p_i(X)$ is of "high degree" just when i equals the secret index idx_0, the verifier can use $(p_i(x))_{i=0}^{N-1}$, together with some pre-challenge "correction terms", to pick out a re-encryption of the desired list element. Our observation is essentially that this vector may be reused and "rotated" in order to systematically pick out the *other* terms. Thus, the protocol need only be conducted once—or twice, rather—to handle (C_{idx_0}, D) and (C_{idx_1}, D); once this is done, the *same* vectors $(p_i(x))_{i=0}^{N-1}$ and $(q_i(x))_{i=0}^{N-1}$ (let's say) may be rotated by the verifier to obtain the remaining terms (in some still-unknown order). In fact, in each step we rotate both $(p_i(x))_{i=0}^{N-1}$ and $(p_i(x))_{i=0}^{N-1}$ by *two positions*; in precisely the case that N is even and idx_0 and idx_1 have opposite parity, this approach then yields each element of the original list exactly once. That the parities of the secret indices idx_0 and idx_1 are indeed opposite may be proven using variants of ideas which are already present in [GK15] and [BCC+15].

We thus construct a secret permutation $\kappa\colon \{0,\ldots,N-1\} \to \{0,\ldots,N-1\}$ of a special form. We begin with secret elements idx_0 and idx_1 of opposite parities; for arbitrary i, we then have that:

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

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

A further way to understand these permutations involves permutation *matrices*. Essentially, we facilitate the construction of the top two rows of an unknown matrix; by performing two-step rotations, we obtain the remaining $N-2$ rows. The ultimate matrix is a permutation matrix if and only if the top two rows attain the value 1 at indices of opposite parity. This is described in the figures below:

$$\left[\begin{array}{l} \underbrace{0,\ldots\ldots\ldots\ldots,1,\ldots\ldots\ldots\ldots,0}_{\text{1 only at index } \mathsf{idx}_0} \\ \underbrace{0,\ldots\ldots,1,\ldots\ldots\ldots\ldots\ldots,0}_{\text{1 only at index } \mathsf{idx}_1} \\ 0,\ldots\ldots\ldots\ldots\ldots\ldots,1,\ldots,0 \\ 0,\ldots\ldots\ldots\ldots\ldots,1,\ldots\ldots\ldots,0 \\ \qquad\qquad\vdots \\ 0,\ldots\ldots\ldots,1,\ldots\ldots\ldots\ldots\ldots,0 \\ 0,\ldots,1,\ldots\ldots\ldots\ldots\ldots\ldots,0 \end{array}\right]$$

Figure 1: "Prover's view".

$$\left[\begin{array}{llll} \multicolumn{4}{l}{\text{———————}\ (p_i(x))_{i=0}^{N-1}\ \text{———————}} \\ \multicolumn{4}{l}{\text{———————}\ (q_i(x))_{i=0}^{N-1}\ \text{———————}} \\ \text{———} & \text{———————} & (p_i(x))_{i=0}^{N-1} & \text{——} \\ \text{———} & \text{———————} & (q_i(x))_{i=0}^{N-1} & \text{——} \\ & & \vdots & \\ \text{——} & (p_i(x))_{i=0}^{N-1} & \text{———————} & \text{———} \\ \text{——} & (q_i(x))_{i=0}^{N-1} & \text{———————} & \text{———} \end{array}\right]$$

Figure 2: "Verifier's view".

Intuitively speaking, the prover and verifier don't *need* the freedom of using an arbitrary permutation. As far as the prover and verifier are concerned, two permutations are "essentially the same" so long as they agree at the points 0 and 1. The prover and verifier may, therefore, agree in advance upon a scheme which, given prescribed (and secret!) values only at 0 and 1, constructs from these a full permutation.

3.2 Circular convolutions and the number-theoretic transform

The main *computational* bottleneck of the above scheme concerns the matrix of Figure 2 above. Indeed, in practice the verifier must "multiply" the vectors $(C_i)_{i=0}^{N-1}$ and $(y_i)_{i=0}^{N-1}$ of curve points by this matrix of scalars (we view the elliptic curve point group $\mathbb{G}$ as a module over $\mathbb{F}_q$, or rather as a vector space). The prover too has to perform a similar matrix multiplication, in the process of deriving the correction terms. In fact, implemented *naïvely*, the above scheme requires $\mathcal{O}(N^2)$ computation for both the prover and the verifier.

Our second key observation is that the matrix of Fig. 2 is a "circulant" matrix, or rather the interleaving of the even-indexed rows of two such matrices. The multiplication of $(C_i)_{i=0}^{N-1}$ (say) by this

matrix, then, is exactly an interleaving of (the even indices of) two discrete circular convolutions—namely of $(C_i)_{i=0}^{N-1}$ by the field-element vectors $(p_i(x))_{i=0}^{N-1}$ and $(q_i(x))_{i=0}^{N-1}$, respectively. This matrix multiplication can be evaluated in $\mathcal{O}(N \log N)$ time using the number-theoretic transform (or rather two of them), provided that N is a power of 2 and $\mathbb{F}_q$'s 2-adic roots of unity number at least N. Fortuitously, the curve used in Ethereum precompiles has a prime order q for which the 2-power-torsion of $(\mathbb{F}_q)^*$ has order 2^{28}—more than sufficient for our purposes. Indeed, this curve was selected with FFTs in mind.

These ideas originated with Pollard [Pol71], and are surveyed in [Nus82, §8]. In both settings, only the convolution of two *field-element* vectors is discussed; in our setting, a vector of *module* elements (i.e., curve points) is 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 Fourier transform; the techniques indeed carry through (even when "field multiplication" can't be performed on the signal). Despite our having made a cursory search, this observation appears absent from the literature; in any case, it complements well-known techniques.

4 Security Proofs

In this section, we sketch proofs of soundness and zero-knowledge.

4.1 Soundness

Following the soundness proofs [GK15, Thm. 2] and [BCC+15, §B.1], we first argue that, for a prover who succeeds against two values of w, each response $(f_{k,i,j})_{k,i,j=0}^{1,m-1,1}$ necessarily takes the form $f_{k,i,j} = q_{k,i,j} \cdot w + p_{k,i,j}$ for well-defined bits $q_{k,i,j} \in \{0,1\}$ for which $q_{k,i,0} + q_{k,i,1} = 1$ (for each k, i). We henceforth denote, for each $k \in \{0,1\}$, by idx_k that element $i \in \{0, \ldots, N-1\}$ whose (little-endian) binary representation is given by $(q_{k,i,1})_{i=0}^{m-1}$.

We argue in addition that:

Claim 1. $\mathsf{idx}_0 \not\equiv \mathsf{idx}_1 \mod 2$.

Proof. Again following [BCC+15, §B.1], we extract elements $(x_j, y_j)_{j=0}^{1}$ which open X and Y, and for which, by the verification equation 104, $y_j \cdot w + x_j = f_{0,0,j} \cdot f_{1,0,j}$ (for each $j \in \{0,1\}$). For each j, this right-hand side is a polynomial in w whose degree-2 term is exactly $q_{0,0,j} \cdot q_{1,0,j}$. As in the proof of [BCC+15, §B.1], we assert from the equality of these polynomials at three values of w that this quadratic coefficient is 0 (for each j). We observe now that $q_{0,0,1} \cdot q_{1,0,1}$ represents the logical conjunction of idx_0 and idx_1's (respective) least-significant bits, whereas $q_{0,0,0} \cdot q_{1,0,0}$ represents the conjunction of their logical negations; that both conjunctions are zero expresses exactly that the least-significant bits are distinct. □

Claim 2. *The prover possesses* sk *for which* $g^{\mathsf{sk}} = y_{\mathsf{idx}_0}$*, as well as elements* $b^*, b' \in \{0, \ldots, \mathsf{MAX}\}$ *for which* $g^{b^*} \cdot D^{\mathsf{sk}} = C_{\mathsf{idx}_0}$ *and* $g^{b'} \cdot (C_{Rn,\mathsf{idx}_0})^{\mathsf{sk}} = C_{Ln,\mathsf{idx}_0}$.

Proof. We first fix d, w, y, z, and x. For each such choice, the extractor, by running the standard Schnorr extractor on two choices of c, may extract a value $\widehat{\mathsf{sk}}$ (not necessarily equal to sk) for which $\overline{g}^{\widehat{\mathsf{sk}}} = \overline{y_0}$. Similarly, we consider equation 119, reproduced below:

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

where $c_{\mathsf{commit}} = \left((\overline{D} \cdot D')^{z^2} \cdot (\overline{C_{Rn}} \cdot C'_{Rn})^{z^3}\right)^{s_{\mathsf{sk}}} \cdot \left((\overline{C_0} \cdot C')^{z^2} \cdot (\overline{C_{Ln}} \cdot C'_{Ln})^{z^3}\right)^{-c}$. By subtracting two copies of this equation, the Schnorr extractor simultaneously obtains an equality:

$$g^{w^m \cdot \hat{t}} \cdot h^{w^m \cdot \tau_x} \stackrel{?}{=} g^{w^m \cdot \delta(y,z)} \cdot {V_1}^{z^2} \cdot {V_2}^{z^3} \cdot \left(T_1^x \cdot T_2^{x^2}\right)^{w^m},$$

where, for abbreviation, we write $V_1 = (\overline{D}^{-\widehat{\mathsf{sk}}} \cdot \overline{C_0}) \cdot (D'^{-\widehat{\mathsf{sk}}} \cdot C')$ and $V_2 = (\overline{C_{Rn}}^{-\widehat{\mathsf{sk}}} \cdot \overline{C_{Ln}}) \cdot (C'^{-\widehat{\mathsf{sk}}}_{Rn} \cdot C'_{Ln})$.

We now adapt the Bulletproofs extractor. Exactly as in [BBB+, §C], by running the inner product extractor against two values of x, the extractor may obtain openings α, ρ, $\mathbf{a}_L$, $\mathbf{a}_R$, $\mathbf{s}_L$, and $\mathbf{s}_R$ of A and S; for each *particular* choice of x, the extractor obtains expressions for $\mathbf{l}$ and $\mathbf{r}$ of the form of 71 and 72.

Meanwhile, by obtaining values $\hat{t}$ and τ_x as above for three distinct values of x, the extractor may obtain openings t_1, t_2, τ_1, and τ_2 of T_1 and T_2, as well as openings b, γ for which $\left(g^b h^\gamma\right)^{w^m} = {V_1}^{z^2} \cdot {V_2}^{z^3}$. (We remark that V_1 and V_2 do not depend on z, y, x, or c.) As in [BBB+, §C], by obtaining such openings b, γ for two distinct values of z, the extractor may calculate individual openings (b^*, γ^*) and (b', γ') for which $\left(g^{b^*} h^{\gamma^*}\right)^{w^m} = V_1$ and $\left(g^{b'} h^{\gamma'}\right)^{w^m} = V_2$. We thus establish, for 2 distinct challenges z and $2 \cdot n$ distinct challenges y, the equality (at 3 values x) of the polynomials $t(X) = w^m \cdot \left(\delta(y,z) + z^2 \cdot b^* + z^3 \cdot b' + t_1 \cdot X + t_2 \cdot X^2\right)$ and $p(X) = w^m \cdot \langle l(X), r(X) \rangle$. Assuming now that w is nonzero, we conclude as in [BBB+, §C] b^* and b' reside in $\{0, \ldots, \mathsf{MAX}\}$.

Finally, the Schnorr extractor also obtains quantities v^*, v', ν^*, and ν' for which:

$$g^{v^*} = \overline{D}^{-\widehat{\mathsf{sk}}} \cdot \overline{C_0}, \quad g^{v'} = \overline{C_{Rn}}^{-\widehat{\mathsf{sk}}} \cdot \overline{C_{Ln}}, \quad h^{\nu^*} = {D'}^{-\widehat{\mathsf{sk}}} \cdot C', \quad h^{\nu'} = {C'_{Rn}}^{-\widehat{\mathsf{sk}}} \cdot C'_{Ln}.$$

By comparing the openings (b^*, γ^*) and (b', γ') of V_1 and V_2 with the representations above, we derive from the uniqueness of Pedersen representations the *individual* equalities:

$$g^{w^m \cdot b^*} = \overline{D}^{-\widehat{\mathsf{sk}}} \cdot \overline{C_0},$$
$$g^{w^m \cdot b'} = \overline{C_{Rn}}^{-\widehat{\mathsf{sk}}} \cdot \overline{C_{Ln}}.$$

We thus obtain, for *each* choice w, an element $\widehat{\mathsf{sk}}$, and equalities as above. We finally run the one-out-of-many extractor. As in the proofs [GK15, Thm. 3] and [BCC+15, §B.2], after running the prover for two choices of w, the extractor may construct, for *each particular* w, representations:

$$(\overline{y_0}, \overline{g}) = \left(y_{\mathsf{idx}_0}^{w^m} \cdot \prod_{i=0}^{m-1} \widehat{y_{0,i}}^{-w^i}, g^{w^m} \cdot \prod_{i=0}^{m-1} \widetilde{g_i}^{-w^i} \right),$$
$$\left(\overline{C_0}, \overline{D}\right) = \left(C_{\mathsf{idx}_0}^{w^m} \cdot \prod_{i=0}^{m-1} \widehat{C_{0,i}}^{-w^i}, D^{w^m} \cdot \prod_{i=0}^{m-1} \widetilde{D_i}^{-w^i} \right),$$
$$\left(\overline{C_{Ln}}, \overline{C_{Rn}}\right) = \left(C_{Ln,\mathsf{idx}_0}^{w^m} \cdot \prod_{i=0}^{m-1} \widehat{C_{Ln,i}}^{-w^i}, C_{Rn,\mathsf{idx}_0}^{w^m} \cdot \prod_{i=0}^{m-1} \widehat{C_{Rn,i}}^{-w^i} \right),$$

where $\left(\widehat{y_{0,i}}, \widehat{C_{0,i}}, \widehat{C_{Ln,i}}, \widehat{C_{Rn,i}}\right)_{i=0}^{m-1}$ are easily computed by the extractor and depend only on data sent by the prover before receiving w.

Choosing an arbitrary such w and obtaining a corresponding $\widehat{\mathsf{sk}}$ as above, we deduce the refined equalities:

$$y_{\mathsf{idx}_0}^{w^m} \cdot \prod_{i=0}^{m-1} \widehat{y_{0,i}}^{-w^i} = \left(g^{w^m} \cdot \prod_{i=0}^{m-1} \widetilde{g_i}^{-w^i} \right)^{\widehat{\mathsf{sk}}},$$
$$\left(g^{-b^*} \cdot C_{\mathsf{idx}_0} \right)^{w^m} \cdot \prod_{i=0}^{m-1} \widehat{C_{0,i}}^{-w^i} = \left(D^{w^m} \cdot \prod_{i=0}^{m-1} \widetilde{D_i}^{-w^i} \right)^{\widehat{\mathsf{sk}}},$$
$$\left(g^{-b'} \cdot C_{Ln,\mathsf{idx}_0} \right)^{w^m} \cdot \prod_{i=0}^{m-1} \widehat{C_{Ln,i}}^{-w^i} = \left(C_{Rn,\mathsf{idx}_0}^{w^m} \cdot \prod_{i=0}^{m-1} \widehat{C_{Rn,i}}^{-w^i} \right)^{\widehat{\mathsf{sk}}}.$$

Taking discrete logarithms with respect to g, our situation is that of three algebraic equations, with

unknown coefficients, over $\mathbb{F}_q^2$, which the point $(w, \widehat{\mathsf{sk}})$ simultaneously satisfies:

$$\log(y_{\mathsf{idx}_0}) \cdot w^m - \sum_{i=0}^{m-1} \log(\widehat{y_{0,i}}) \cdot w^i = \left(1 \cdot w^m - \sum_{i=0}^{m-1} \log(\widetilde{g_i}) \cdot w^i\right) \cdot \widehat{\mathsf{sk}}, \tag{3}$$

$$\log(g^{-b^*} \cdot C_{\mathsf{idx}_0}) \cdot w^m - \sum_{i=0}^{m-1} \log(\widehat{C_{0,i}}) \cdot w^i = \left(\log(D) \cdot w^m - \sum_{i=0}^{m-1} \log(\widetilde{D_i}) \cdot w^i\right) \cdot \widehat{\mathsf{sk}}, \tag{4}$$

$$\log(g^{-b'} \cdot C_{Ln,\mathsf{idx}_0}) \cdot w^m - \sum_{i=0}^{m-1} \log(\widehat{C_{Ln,i}}) \cdot w^i = \left(\log(C_{Rn,\mathsf{idx}_0}) \cdot w^m - \sum_{i=0}^{m-1} \log(\widehat{C_{Rn,i}}) \cdot w^i\right) \cdot \widehat{\mathsf{sk}}. \tag{5}$$

Indeed, the extractor may generate arbitrary pairs $(w^{(i)}, \widehat{\mathsf{sk}}^{(i)})$ satisfying these equations. This setting is thus exactly that of the "interpolation" of an unknown *rational* function given black-box access to it; we refer to the section *Cauchy interpolation* of the text of von zur Gathen and Gerhard [vzGG13, §5.8]. In fact, we reside in 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). For thoroughness, we outline the details.

We first generate $2m + 1$ pairs $(w^{(i)}, \widehat{\mathsf{sk}}^{(i)}), i \in \{0, \ldots, 2 \cdot m\}$; we view each equation above as an instance of [vzGG13, (21)] (using the parameters $n = 2m + 1$, $k = m + 1$). We denote by r and t the (unknown) polynomials, in the indeterminate W, which appear in (3)'s left- and right-hand sides. We immediately construct polynomials $r_j, t_j \in \mathbb{F}_q[W]$ as in the statement of [vzGG13, Cor. 5.18]; adapting the proof of [vzGG13, Thm. 5.16, (ii)] (we note that its condition still holds), we see that there exists some nonzero $\alpha \in \mathbb{F}_q[W]$ for which

$$(r, t) = (\alpha \cdot r_j, \alpha \cdot t_j). \tag{6}$$

Because t is monic, α's leading coefficient must equal τ^{-1} (as in [vzGG13, §5.18, (ii)], we denote by $\tau = \mathrm{lc}(t_j)$ the leading coefficient of t_j). We see that $\log(y_{\mathsf{idx}_0})$ can be concretely determined as $\tau^{-1} \cdot \mathrm{lc}(r_j)$. (We note that the extractor in general does not know α.)

In fact, (6) holds for *arbitrary* polynomials r, t of appropriate degree for which $r(w^{(i)}) = t(w^{(i)}) \cdot \widehat{\mathsf{sk}}^{(i)}$ for each i (though in general with different α). For example, in the case of (4), we see that α's leading coefficient must equal $\log(D) \cdot \tau^{-1}$; we conclude that $\log(g^{-b^*} \cdot C_{\mathsf{idx}_0}) = \log(D) \cdot \tau^{-1} \cdot \mathrm{lc}(r_j) = \log(D) \cdot \log(y_{\mathsf{idx}_0})$. Similarly, from (5) we see that that $\log(g^{-b'} \cdot C_{Ln,\mathsf{idx}_0}) = \log(C_{Rn,\mathsf{idx}_0}) \cdot \log(y_{\mathsf{idx}_0})$. These facts together suffice to establish the Claim. □

Claim 3. *The prover possesses r for which $g^r = D$. Moreover, $(y_{\mathsf{idx}_0} \cdot y_{\mathsf{idx}_1})^r = C_{\mathsf{idx}_0} \cdot C_{\mathsf{idx}_1}$, and, for each $i \notin \{\mathsf{idx}_0, \mathsf{idx}_1\}$, $y_i^r = C_i$.*

Proof. As in the proof of Claim 2, by fixing d and w and running the Schnorr extractor on two choices of c, the extractor obtains a value r for which, simultaneously, $D = g^r$ and $\overline{C_X} = \overline{y_X}^r$. On the other hand, after running the prover on two choices of w, the extractor may generate, for any *particular* w, an expression:

$$\left(\overline{C_X}, \overline{y_X}\right) = \left(\left(\prod_{k,j=0}^{1,\frac{N}{2}-1} C_{\mathsf{idx}_k+2\cdot j}^{\mathbf{d}_{k+2\cdot j}}\right)^{w^m} \cdot \prod_{i=0}^{m-1} \widehat{C_{X,i}}^{-w^i}, \left(\prod_{k,j=0}^{1,\frac{N}{2}-1} y_{\mathsf{idx}_k+2\cdot j}^{\mathbf{d}_{k+2\cdot j}}\right)^{w^m} \cdot \prod_{i=0}^{m-1} \widehat{y_{X,i}}^{-w^i}\right),$$

where $\mathbf{d} = \left(1, 1, d, d^2, \ldots, d^{N-2}\right)$ is as in the protocol, and the elements $\left(\widehat{C_{X,i}}, \widehat{y_{X,i}}\right)_{i=0}^{m-1}$ (easily computed by the extractor) depend only on data sent by the prover before receiving w.

Combining the above facts, we derive the refined equality:

$$\left(\prod_{k,j=0}^{1,\frac{N}{2}-1} C_{\mathsf{idx}_k+2\cdot j}^{\mathbf{d}_{k+2\cdot j}}\right)^{w^m} \cdot \prod_{i=0}^{m-1} \widehat{C_{X,i}}^{-w^i} = \left(\left(\prod_{k,j=0}^{1,\frac{N}{2}-1} y_{\mathsf{idx}_k+2\cdot j}^{\mathbf{d}_{k+2\cdot j}}\right)^{w^m} \cdot \prod_{i=0}^{m-1} \widehat{y_{X,i}}^{-w^i}\right)^r.$$

In fact, for fixed d the extractor may establish this equalities for $m+1$ distinct values of w (observe that r doesn't depend on w, or even d). The two sides of this equation thus represent polynomials of degree m in w, which moreover agree at $m+1$ distinct values; as in [GK15, Thm. 3], we conclude after inverting a Vandermonde matrix that their coefficients are identical. We see in particular that $\left(\prod_{k,j=0}^{1,\frac{N}{2}-1} y_{\mathsf{idx}_k+2\cdot j}^{\mathbf{d}_{k+2\cdot j}}\right)^r = \prod_{k,j=0}^{1,\frac{N}{2}-1} C_{\mathsf{idx}_k+2\cdot j}^{\mathbf{d}_{k+2\cdot j}}$.

We turn to this latter equality, whose dependence on d resides *only* in the exponent (the values idx_k are well-defined as soon as Q is released). Indeed, by obtaining this equality for $N-1$ distinct values of d, and inverting a second Vandermonde matrix, the extractor establishes the individual equalities

$$(y_{\mathsf{idx}_0} \cdot y_{\mathsf{idx}_1})^r = C_{\mathsf{idx}_0} \cdot C_{\mathsf{idx}_1},$$
$$\{(y_{\mathsf{idx}_k+2\cdot j})^r = C_{\mathsf{idx}_k+2\cdot j}\}_{k,j=0,1}^{1,\frac{N}{2}-1}.$$

We finally argue, in virtue of Claim 1, that the set $\{C_{\mathsf{idx}_k+2\cdot j}, y_{\mathsf{idx}_k+2\cdot j}\}_{k,j=0,1}^{1,\frac{N}{2}-1}$ is in bijection with $\{(C_i, y_i)\}_{i\notin\{\mathsf{idx}_0,\mathsf{idx}_1\}}$. These facts together imply the statement of the Claim. □

4.2 Zero Knowledge

We describe a simulator which outputs accepting transcripts; we argue in steps that these transcripts are indistinguishable from actual ones.

Claim 4. *The elements $(f_{k,i})_{k,i=0}^{1,m-1}$, P, Q, U, V, X, Y, z_P, z_U, z_X, and w can be simulated perfectly indistinguishably.*

Proof. As in the one-out-of-many simulator [BCC+15, §B.1], our simulator randomly chooses $(f_{k,i} = f_{k,i,1})_{k,i}^{1,m-1}$, Q, U, Y, z_P, z_U, z_X, and w; it then sets $(f_{k,i,0} = w - f_{k,i,1})_{k,i=0}^{1,m-1}$, as well as:

$$P = Q^{-w} \cdot \mathrm{Com}\left(f_{0,0,0}, \ldots, f_{1,m-1,1}; z_P\right),$$
$$V = U^{-w} \cdot \mathrm{Com}\left((f_{k,i,j}(w - f_{k,i,j}))_{k,i,j=0}^{1,m-1,1}; z_U\right),$$
$$X = Y^{-w} \cdot \mathrm{Com}\left((f_{0,0,j} \cdot f_{1,0,j})_{j=0}^{1}; z_X\right).$$

We argue just as in [BCC+15, §B.1] that this simulation is perfect. □

Claim 5. *The elements $\hat{t}$, τ_x, μ, A, S, T_1, T_2, $\mathbf{l}$, $\mathbf{r}$, z, y, and x can be simulated perfectly indistinguishably.*

Proof. We adapt the Bulletproofs simulator [BBB+, §C]. Our simulator randomly selects $\hat{t}$, τ_x, μ, S, T_1, T_2, $\mathbf{l}$, $\mathbf{r}$, z, y, and x, and sets:

$$A = h^{\mu} \cdot \mathbf{g}^{\mathbf{l}} \cdot \mathbf{h'}^{\mathbf{r}} \cdot \left(S^x \cdot \mathbf{g}^{-z} \cdot \mathbf{h'}^{z\cdot\mathbf{y}^{2\cdot n}} \cdot \mathbf{h'}^{z^2\cdot(\mathbf{2}^n\|\mathbf{0}^n)+z^3\cdot(\mathbf{0}^n\|\mathbf{2}^n)}\right)^{-1}.$$

The simulator also runs the inner product argument with the verifier, using the simulated witnesses $\mathbf{l}$ and $\mathbf{r}$. As in [BBB+, §C], we argue that the elements generated in this way are distributed identically to those in actual transcripts. □

Claim 6. *The elements (C', D') and (C'_{Ln}, C'_{Rn}), $A_{C'}$, and $A_{C'_{Ln}}$, and finally s_{ν^*}, $s_{\nu'}$, and c can be simulated computationally indistinguishably.*

Proof. The simulator may generate (C', D') and (C'_{Ln}, C'_{Rn}) randomly; by the Diffie–Hellman assumption, their analogues in real proofs (ElGamal ciphertexts under y_{idx_0}, with independent randomnesses) are indistinguishable from random elements of $\mathbb{G}^2$. The simulator may also generate s_{ν^*}, $s_{\nu'}$, and c randomly. Finally, as in the Schnorr simulator, these quantities uniquely determine the remaining values $A_{C'}$, and $A_{C'_{Ln}}$. □

Claim 7. *The elements s_{sk}, s_r, s_{v^*}, $s_{v'}$, and c, as well as $\left(\widetilde{C_{Ln,i}}, \widetilde{C_{Rn,i}}, \widetilde{C_{0,i}}, \widetilde{y_{0,i}}, \widetilde{C_{X,i}}, \widetilde{y_{X,i}}, \widetilde{D_i}, \widetilde{g_i}\right)_{i=0}^{m-1}$, and finally A_y, A_D, A_u, A_X, A_t, $A_{\overline{C_0}}$, and $A_{\overline{C_{Ln}}}$, can be simulated computationally indistinguishably.*

Proof. The simulator randomly generates $\left(\widetilde{C_{Ln,i}}, \widetilde{C_{Rn,i}}, \widetilde{C_{0,i}}, \widetilde{y_{0,i}}, \widetilde{C_{X,i}}, \widetilde{y_{X,i}}, \widetilde{D_i}, \widetilde{g_i}\right)_{i=0}^{m-1}$. These values, together with those of w and $(f_{k,i})_{k,i=0}^{1,m-1}$, uniquely determine the intermediate values $\overline{C_{Ln}}$, $\overline{C_{Rn}}$, $\overline{C_0}$, $\overline{y_0}$, $\overline{C_X}$, and $\overline{y_X}$. Finally, the simulator randomly generates s_{sk}, s_r, s_{v^*}, $s_{v'}$, and c (we assume that c is generated consistently with respect to the above). These, together with those generated above and the verification equations, uniquely determine the remaining quantities A_y, A_D, A_u, A_X, A_t, $A_{\overline{C_0}}$, and $A_{\overline{C_{Ln}}}$.

We argue that transcripts generated in this way are indistinguishable from actual transcripts. For each $i \in \{0, \ldots, m-1\}$, we have the expressions:

$$\left(\widetilde{C_{0,i}}, \widetilde{D_i}\right) = \left(\mathsf{MultiExp}\left((C_j)_{j=0}^{N-1}, (r_{0,j,i})_{j=0}^{N-1}\right)\left(y_{\mathsf{idx}_0}^r\right)^{\sigma_{0,i}}, D^{\sigma_{0,i}}\right),$$
$$\left(\widetilde{y_{0,i}}, \widetilde{g_i}\right) = \left(\mathsf{MultiExp}\left((y_j)_{j=0}^{N-1}, (r_{0,j,i})_{j=0}^{N-1}\right)\left(y_{\mathsf{idx}_0}\right)^{\sigma_{0,i}}, g^{\sigma_{0,i}}\right).$$

These pairs are ElGamal encryptions under y_{idx_0}; indeed, the tuples $\left(\widetilde{C_{0,i}}, \widetilde{D_i}, \widetilde{y_{0,i}}, \widetilde{g_i}\right)$ (each of which depends bijectively on the uniform secret $\sigma_{0,i}$) implicitly define independent, uniformly random Diffie–Hellman quadruples (each with secrets r and sk). We conclude that no adversary can distinguish them from random group elements.

We further have the expressions:

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

These indeed give ElGamal "encryptions" under the "public key" $g^r = D$, with further independent randomnesses $\sigma_{X,i}$. We conclude that no adversary can distinguish them from random elements.

We finally consider the pairs

$$\left(\widetilde{C_{Ln,i}}, \widetilde{C_{Rn,i}}\right) = \left(\mathsf{MultiExp}\left((C_{Ln,j})_{j=0}^{N-1}, (r_{0,j,i})_{j=0}^{N-1}\right) \cdot y_{\mathsf{idx}_0}^{\pi_i}, \mathsf{MultiExp}\left((C_{Rn,j})_{j=0}^{N-1}, (r_{0,j,i})_{j=0}^{N-1}\right) \cdot g^{\pi_i}\right).$$

For each i, this pair varies throughout certain *translates* (in $\mathbb{G}^2$) of an ElGamal encryption under y_{idx_0}, with randomness given by the uniform, independent secret π_i; we conclude that these pairs are indistinguishable from random elements. □

5 Protocol Specification

We now specify in detail a zero-knowledge proof protocol for the statement $\mathsf{st}_{\mathsf{AnonTransfer}}$ (1) above, following the *Bulletproofs* paper of Bünz, Bootle, Boneh, Poelstra, Wuille and Maxwell [BBB+] where applicable. We denote by n that integer for which $\mathsf{MAX} = 2^n - 1$ and by m that integer for which $N = 2^m$. All vector indices are understood to be taken modulo the size of the vector (in case of overflow). We define and make use of the following functions:

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

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

Protocol Anonymous Zether

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

$\alpha, \rho \leftarrow_\$ \mathbb{Z}_q$ ▷ Begin Bulletproof [BBB+, §4]

$\mathbf{a}_L \in \{0,1\}^{2 \cdot n}$ s.t. $\langle \mathbf{a}_{L[:n]}, \mathbf{2}^n \rangle = b^*, \langle \mathbf{a}_{L[n:]}, \mathbf{2}^n \rangle = b'$

$\mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^{2 \cdot n}$

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

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

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

$r_P, r_Q, r_U, r_V, r_X, r_Y \leftarrow_\$ \mathbb{Z}_q$ ▷ Begin one-out-of-many proof [BCC+15]

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

sample $p_{k,i,1} \leftarrow_\$ \mathbb{Z}_q$; set $p_{k,i,0} = -p_{k,i,1}$

set $q_{k,i,1} = (\mathsf{idx}_k)_i$, i.e., the i^{th} (little-endian) bit of idx_k; set $q_{k,i,0} = \neg q_{k,i,1}$

end for

$P = \text{Com}(p_{0,0,0}, \ldots, p_{1,m-1,1}; r_P)$

$Q = \text{Com}(q_{0,0,0}, \ldots, q_{1,m-1,1}; r_Q)$

$U = \text{Com}((p_{k,i,j}(1 - 2q_{k,i,j}))_{k,i,j=0}^{1,m-1,1}; r_U)$

$V = \text{Com}(-p_{0,0,0}^2, \ldots, -p_{0,m-1,1}^2; r_V)$

$X = \text{Com}\left((p_{0,0,j} \cdot p_{1,0,j})_{j=0}^{1}, r_X\right)$

$Y = \text{Com}\left((p_{0,0,j} \cdot q_{1,0,j} + p_{1,0,j} \cdot q_{0,0,j})_{j=0}^{1}, r_Y\right)$

end $\mathcal{P}$

$\mathcal{P} \to \mathcal{V} : A, S, P, Q, U, V, X, Y$

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

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

$\mathcal{P}$ **sets...** ▷ in what follows, we denote by $(j)_i$ the i^{th} bit of j.

set $\left(r_{k,j}(W) = \prod_{i=0}^{m-1}(q_{k,i,(j)_i} \cdot W + p_{k,i,(j)_i}) = \sum_{i=0}^{m} r_{k,j,i} \cdot W^i\right)_{k,j=0}^{1,N-1}$

$(\pi_i)_{i=0}^{m-1}, (\sigma_{0,i})_{i=0}^{m-1}, (\sigma_{1,i})_{i=0}^{m-1}, (\sigma_{X,i})_{i=0}^{m-1} \leftarrow_\$ \mathbb{Z}_q$

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

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

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

$\widetilde{C_{0,i}} = \mathsf{MultiExp}\left((C_j)_{j=0}^{N-1}, (r_{0,j,i})_{j=0}^{N-1}\right) \left(y_{\mathsf{idx}_0}^r\right)^{\sigma_{0,i}}$

$\widetilde{y_{0,i}} = \mathsf{MultiExp}\left((y_j)_{j=0}^{N-1}, (r_{0,j,i})_{j=0}^{N-1}\right) \left(y_{\mathsf{idx}_0}\right)^{\sigma_{0,i}}$

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

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

$\widetilde{y_{X,i}} = g^{\sigma_{X,i}}$

$\widetilde{D_i} = D^{\sigma_{0,i}}$

$\widetilde{g_i} = g^{\sigma_{0,i}}$

end for

end $\mathcal{P}$

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

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

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

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

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

set $f_{k,i} = q_{k,i,1} \cdot w + p_{k,i,1}$

end for

$z_P = r_Q \cdot w + r_P$

$z_U = r_U \cdot w + r_V$

$z_X = r_Y \cdot w + r_X$

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

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

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

$\overline{D} = D^{w^m - \sum_{i=0}^{m-1} \sigma_{0,i} \cdot w^i}$

$\overline{g} = g^{w^m - \sum_{i=0}^{m-1} \sigma_{0,i} \cdot w^i}$

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

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

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

end $\mathcal{P}$

$\mathcal{P} \to \mathcal{V} : (f_{k,i})_{k,i=0}^{1,m-1}, z_P, z_U, z_X, C', D', C'_{Ln}, C'_{Rn}$

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

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

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

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

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

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

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

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

end $\mathcal{P}$

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

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

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

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

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

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

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

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

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

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

$A_D = g^{k_r}$

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

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

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

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

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

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

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

end $\mathcal{P}$

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

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

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

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

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

$s_r = k_r + c \cdot r$

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

93: $s_{v'} = k_{v'} + c \cdot w^m \cdot b'$
94: $s_{\nu^*} = k_{\nu^*} + c \cdot w^m \cdot \gamma^*$
95: $s_{\nu'} = k_{\nu'} + c \cdot w^m \cdot \gamma'$
96: **end** $\mathcal{P}$
97: $\mathcal{P} \to \mathcal{V} : s_{\mathsf{sk}}, s_r, s_{v^*}, s_{v'}, s_{\nu^*}, s_{\nu'}$
98: $\mathcal{V}$ **:**
99: **for all** $k \in \{0,1\}$, $i \in \{0,\dots,m-1\}$ **do**
100: set $f_{k,i,1} = f_{k,i}$, $f_{k,i,0} = w - f_{k,i}$
101: **end for**
102: $Q^w P \stackrel{?}{=} \mathrm{Com}\left(f_{0,0,0},\dots,f_{1,m-1,1}; z_P\right)$
103: $U^w V \stackrel{?}{=} \mathrm{Com}\left((f_{k,i,j}(w - f_{k,i,j}))_{k,i,j=0}^{1,m-1,1}; z_U\right)$
104: $Y^w X \stackrel{?}{=} \mathrm{Com}\left((f_{0,0,j} \cdot f_{1,0,j})_{j=0}^{1}; z_X\right)$ ▷ Opposite parity check
105: set $\left(r_{k,j}(w) = \prod_{i=0}^{m-1} f_{k,i,(j)_i}\right)_{k,j=0}^{1,N-1}$ ▷ Begin computation of re-encryptions
106: $\overline{C_{Ln}} = \mathsf{MultiExp}\left((C_{Ln,j})_{j=0}^{N-1}, (r_{0,j}(w))_{j=0}^{N-1}\right) \cdot \prod_{i=0}^{m-1} \widetilde{C_{Ln,i}}^{-w^i}$
107: $\overline{C_{Rn}} = \mathsf{MultiExp}\left((C_{Rn,j})_{j=0}^{N-1}, (r_{0,j}(w))_{j=0}^{N-1}\right) \cdot \prod_{i=0}^{m-1} \widetilde{C_{Rn,i}}^{-w^i}$
108: $\overline{C_0} = \mathsf{MultiExp}\left((C_j)_{j=0}^{N-1}, (r_{0,j}(w))_{j=0}^{N-1}\right) \cdot \prod_{i=0}^{m-1} \widetilde{C_{0,i}}^{-w^i}$
109: $\overline{y_0} = \mathsf{MultiExp}\left((y_j)_{j=0}^{N-1}, (r_{0,j}(w))_{j=0}^{N-1}\right) \cdot \prod_{i=0}^{m-1} \widetilde{C_{Rn,i}}^{-w^i}$
110: set $\mathbf{d} = (\mathbf{d}_0, \mathbf{d}_1, \dots, \mathbf{d}_{N-1}) = \left(1, 1, d, d^2, \dots, d^{N-2}\right)$
111: $\overline{C_X} = \prod_{k,j=0}^{1,\frac{N}{2}-1} \mathsf{MultiExp}\left((C_i)_{i=0}^{N-1}, \mathsf{Shift}\left((r_{k,i}(w))_{i=0}^{N-1}, 2 \cdot j\right)\right)^{\mathbf{d}_{2\cdot j+k}} \cdot \prod_{i=0}^{m-1} \widetilde{C_{X,i}}^{-w^i}$
112: $\overline{y_X} = \prod_{k,j=0}^{1,\frac{N}{2}-1} \mathsf{MultiExp}\left((C_i)_{i=0}^{N-1}, \mathsf{Shift}\left((r_{k,i}(w))_{i=0}^{N-1}, 2 \cdot j\right)\right)^{\mathbf{d}_{2\cdot j+k}} \cdot \prod_{i=0}^{m-1} \widetilde{y_{X,i}}^{-w^i}$
113: $A_y \stackrel{?}{=} \overline{g}^{s_{\mathsf{sk}}} \cdot (\overline{y_0})^{-c}$ ▷ Begin sigma protocol verification
114: $A_D \stackrel{?}{=} g^{s_r} \cdot D^{-c}$
115: $A_u \stackrel{?}{=} g_{\mathsf{epoch}}^{s_{\mathsf{sk}}} \cdot u^{-c}$
116: $A_X \stackrel{?}{=} \overline{y_X}^{s_r} \cdot \overline{C_X}^{-c}$
117: $\delta(y,z) = (z - z^2) \cdot \left\langle \mathbf{1}^{2\cdot n}, \mathbf{y}^{2\cdot n} \right\rangle - \left(z^3 \cdot \langle \mathbf{1}^n, \mathbf{2}^n \rangle + z^4 \cdot \langle \mathbf{1}^n, \mathbf{2}^n \rangle\right)$
118: $c_{\mathsf{commit}} = \left((\overline{D} \cdot D')^{z^2} \cdot (\overline{C_{Rn}} \cdot C'_{Rn})^{z^3}\right)^{s_{\mathsf{sk}}} \cdot \left((\overline{C_0} \cdot C')^{z^2} \cdot (\overline{C_{Ln}} \cdot C'_{Ln})^{z^3}\right)^{-c}$
119: $g^{w^m \cdot c \cdot \hat{t}} \cdot h^{w^m \cdot c \cdot \tau_x} \stackrel{?}{=} g^{w^m \cdot c \cdot \delta(y,z)} \cdot A_t \cdot {c_{\mathsf{commit}}}^{-1} \cdot \left(T_1^x \cdot T_2^{x^2}\right)^{w^m \cdot c}$
120: $A_{\overline{C_0}} \stackrel{?}{=} g^{s_{v^*}} \cdot \overline{D}^{s_{\mathsf{sk}}} \cdot \overline{C_0}^{-c}$
121: $A_{\overline{C_{Ln}}} \stackrel{?}{=} g^{s_{v'}} \cdot \overline{C_{Rn}}^{s_{\mathsf{sk}}} \cdot \overline{C_{Ln}}^{-c}$
122: $A_{C'} \stackrel{?}{=} h^{s_{\nu^*}} \cdot D'^{s_{\mathsf{sk}}} \cdot C'^{-c}$
123: $A_{C'_{Ln}} \stackrel{?}{=} h^{s_{\nu'}} \cdot C'^{s_{\mathsf{sk}}}_{Rn} \cdot C'^{-c}_{Ln}$
124: **end** $\mathcal{V}$
125: $\mathbf{h}' = \left(h_1, h_2^{(y^{-1})}, h_3^{(y^{-2})}, \dots, h_{2\cdot n}^{(y^{-2\cdot n+1})}\right)$ ▷ Complete inner product argument
126: $Z = A \cdot S^x \cdot \mathbf{g}^{-z} \cdot \mathbf{h}'^{z \cdot \mathbf{y}^{2\cdot n}} \cdot \mathbf{h}'^{z^2 \cdot (\mathbf{2}^n \| \mathbf{0}^n) + z^3 \cdot (\mathbf{0}^n \| \mathbf{2}^n)}$
127: $\mathcal{P}$ and $\mathcal{V}$ engage in Protocol 1 of [BBB$^+$] on inputs $(\mathbf{g}, \mathbf{h}', Zh^{-\mu}, \hat{t}; \mathbf{l}, \mathbf{r})$

6 Performance

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

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

	Prov. Time (ms)	Verif. Time (ms)	Prf. Size (bytes)	Gas Used
Burn	907	83	1,312	2,393,134
Transfer (2)	1,861	120	2,720	5,084,502
Transfer (4)	2,005	142	3,296	6,169,442
Transfer (8)	2,252	172	3,872	8,382,722
Transfer (16)	2,844	247	4,448	13,253,724
Transfer (32)	4,200	394	5,024	24,072,647
Transfer (64)	7,264	705	5,600	47,884,056
Transfer (N)	$\mathcal{O}(N \log N)$	$\mathcal{O}(N \log N)$	$\mathcal{O}(\log N)$	$\mathcal{O}(N \log N)$

References

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

[BBB+] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. 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.

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

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

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

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

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