Efficient mixing of arbitrary ballots with everlasting privacy: How to verifiably mix the PPATC scheme

Kristian Gjøsteen[1], Thomas Haines[1], and Morten Rotvold Solberg[1]

Norwegian University of Science and Technology, Trondheim, Norway
{kristian.gjosteen,thomas.haines,mosolb}@ntnu.no

Abstract. The long term privacy of voting systems is of increasing concern as quantum computers come closer to reality. Everlasting privacy schemes offer the best way to manage these risks at present. While homomorphic tallying schemes with everlasting privacy are well developed, most national elections, using electronic voting, use mixnets. Currently the best candidate encryption scheme for making these kinds of elections everlastingly private is PPATC, but it has not been shown to work with any mixnet of comparable efficiency to the current ElGamal mixnets. In this work we give a paper proof, and a machine checked proof, that the variant of Wikström's mixnet commonly in use is safe for use with the PPATC encryption scheme.

Keywords: Everlasting Privacy · E-Voting · Verifiable Shuffles · Coq.

1 Introduction

Traditional paper-based and electronic voting has many good properties, but also limitations. A voter is not able to verify that her vote was counted as she cast it, and confidentiality of the votes relies heavily on trust in the election officials and procedures. In addition there are problems regarding for example counting errors and accessibility. Verifiable electronic voting systems can solve some of these issues. In particular, cryptographic techniques can be used to provide public verifiability of election results and raise each individual voter's confidence in the privacy and integrity of her vote.

To achieve verifiable elections, encrypted votes are often published on a public bulletin board, along with sophisticated cryptographic proofs that allow an individual voter to verify that their ballot was not only listed on the bulletin board, but also included correctly in the tally.

The votes are encrypted to provide confidentiality, which is usually considered essential for a fair vote. Confidentiality requires votes to remain private not only during the time of the election, but for all foreseeable future. However, due to computers and algorithms getting faster and the potential introduction of quantum computers, there is no way to safely predict how long it may take before a ciphertext encrypted today is broken. Thus, the property of *everlasting privacy* has been introduced.

Everlasting privacy is a property of electronic voting schemes where the information released to the public perfectly (or information-theoretically) hides how each voter voted, up to the outcome of the election. This means that regardless of developments in practical computing power and algorithm design, individual votes cannot be recovered from the public record.

Everlasting privacy is a subtle concept. In all systems that are practical for large-scale voting, functional requirements mean that the voter will have to encrypt their ballot and transmit this encryption to some infrastructure. The subtlety is that this ciphertext is not part of the public record. This essentially assumes that the potential powerful *future* attacker did not record the network traffic, and is only working with the public record of the election. This is in many cases a reasonable assumption. We emphasize that it is only privacy against these potential future attackers that relies on this assumption. Computationally secure cryptography still protects against adversaries with greater network access. So schemes that provide everlasting privacy are no less secure than conventional cryptographic voting schemes, but they have greater security against future adversaries that work only from the public record.

There are various candidate constructions which achieve everlasting privacy while maintaining verifiability. Most of the schemes are inspired by Cramer *et al.*'s "Multi-Authority Secret-Ballot Elections with Linear Work" [3] and Moran and Naor's "Split-ballot voting: Everlasting privacy with distributed trust" [12]. In both cases perfectly hiding commitments are combined with zero knowledge proofs to provide verifiability without leaking any information. In this work we will focus on schemes in the style of [12] which are able to handle arbitrary ballots rather than the homomorphic tally supported by [3].

Before the authorities perform the tally, the encrypted ballots need to be decrypted. To ensure that privacy is preserved, the link between each ballot and the voter who submitted the ballot is destroyed by running the ballots through a mix net. Mix nets were first introduced by Chaum [2] and consist of a finite sequence of authorities (mixers), each of which permutes (shuffles) and hides the relationship between its inputs and its outputs. This style of schemes is less developed than the homomorphic schemes, but have greater practical implications since mixnet style schemes have been used in many of countries who have voted electronically (Australia, Estonia, Norway, and Switzerland), and where homomorphic counting is often hard to do.

The general idea in these schemes is to have a publicly verifiable part dealing only with commitments to ballots. We achieve everlasting privacy by using perfectly hiding commitments. However, somehow the ballots must be recovered by the infrastructure, and this is done in a private part, typically working on encrypted openings for the commitments. In this way, we get everlasting privacy. Note that we only get computational integrity.

There are two encryption schemes which are commonly suggested for use in this context, both involve first committing to the message and then encrypting the opening to the commitment. The schemes fit into a wider everlasting privacy scheme with the perfectly hiding commitments being publicly shuffled and then opened providing both verifiability and everlasting privacy; the encrypted openings are shuffled by the authorities and then publicly posted. The first is the MN encryption scheme from Moran and Naor [12] which is built on Paillier encryption [13] and Pedersen commitments [14]. The second is the PPATC encryption from Cuvelier *et al.* [4] which uses ElGamal and Abe *et al.*'s [1] commitment scheme. Since the latter encryption scheme can be instantiated on prime order elliptic curves, rather than the semi-prime RSA groups of the former, it is significantly faster.

Simple and efficient zero-knowledge proofs for correct encryption and decryption of both encryption schemes are known. An efficient mixnet for the MN encryption scheme was proven by Haines and Gritti [11], but at present the most efficient known mixnet for PPATC uses the general version of Terelius-Wikström proof of shuffle [15] which proves statements over the integers using Fujisaki-Okamoto commitments [6], based on an RSA modulus, which hampers the efficiency of the mixnet. The reason is that every operation must happen modulo the RSA modulus, which means that basic arithmetic is very slow. We will use pairing groups, but we arrange it so that most of the group arithmetic happens in a group where arithmetic is much faster, which means that Fujisaki-Okamoto commitments will be slow compared to most of our arithmetic. In practice everyone using the Terelius-Wikström proof of shuffle uses an optimised variant which avoids the use of Fujisaki-Okamoto commitments. It is folklore that the optimised variant of Terelius-Wikström works for wide class of encryption schemes but the precise variant for each encryption scheme should be proven.

1.1 Contribution

We prove a variation of the optimised Terelius-Wikström shuffle [15] for the PPATC encryption scheme [4]. This is essentially the optimised variation which is widely used, and which avoids the use of Fujisaki-Okamoto commitments. In addition we show how the Fiat-Shamir transform can be applied so that the public proofs of correct shuffling can be trivially derived from the private proofs of correct shuffling, nearly doubling the speed of mixing.

We provide a machine-checked proof using the interactive theorem prover Coq. The machine-checked proof relies on recent work which shows that any encryption scheme with certain properties works with the optimised Terelius-Wikström shuffle. For completeness and human understanding, we also give a straight-forward traditional paper proof.

2 Notation and Tools

We denote by $\mathbb{G}_1$ and $\mathbb{G}_2$ cyclic groups of large prime order q, and by $\mathbb{Z}_q$ the field of integers modulo q. Let A^n be the set of vectors of length n, with elements from the set A. We denote vectors in bold, e.g. $\mathbf{a}$. We denote by a_i the ith element of the vector $\mathbf{a}$. Sometimes, we will work with vectors that have tuples as elements. In such cases, we also denote by a_i the ith element of $\mathbf{a}$, and by $a_{i,j}$ the jth element of the tuple a_i. Multiplication of tuples is elementwise multiplication, that is, $\mathbf{ab}$ is the tuple where the ith element is $a_i b_i$. We denote by $A^{n \times n}$ the set of $n \times n$-matrices with elements from the set A. Matrices will be denoted using capital letters, e.g. M. We denote by M_i the ith column of M, by $M_{i,*}$ the ith row of M, and by $M_{i,j}$ the element in row i and column j. A binary relation for a set S of statements and a set W of witnesses is a subset of $S \times W$ and is denoted by $\mathscr{R}$.

Matrix Commitments. We now describe how to commit to a matrix using a variation of Pedersen commitments [15]. We denote by $\mathsf{Com}_{\gamma,\gamma_1}(m,t)$ the Pedersen commitment of $m \in \mathbb{Z}_q$ with randomness $t \in \mathbb{Z}_q$, i.e. $\mathsf{Com}_{\gamma,\gamma_1}(m,t) = \gamma^t \gamma_1^m$ for group generators γ and γ_1. To commit to a vector $\mathbf{v} \in \mathbb{Z}_q^n$, we compute $u = \mathsf{Com}_{\gamma,\gamma_1,\cdots,\gamma_n}(\mathbf{v},t) = \gamma^t \prod_{i=1}^n \gamma_i^{v_i}$, where t is chosen at random from $\mathbb{Z}_q$, and the γs are random group generators. If the commitment parameters are omitted, it is implicit that they are $\gamma, \gamma_1, \cdots, \gamma_n$. An $n \times n$ matrix M is committed to column-wise. For a matrix $M \in \mathbb{Z}_q^{n \times n}$ and a vector $\mathbf{t}$ chosen at random from $\mathbb{Z}_q^n$, we compute the commitment $\mathbf{u}$ of M as

$$\mathbf{u} = \mathsf{Com}(M,\mathbf{t}) = \left(\gamma^{t_1} \varPi_{i=1}^n \gamma_i^{M_{i,1}}, \ldots, \gamma^{t_n} \varPi_{i=1}^n \gamma_i^{M_{i,n}}\right).$$

Abe Commitments. We now describe a perfectly hiding commitment scheme due to Abe *et al.* [1], that is used in a a construction of the PPATC encryption scheme that we describe further down. Let $\varLambda_{sxdh} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h)$ be a description of bilinear groups, where g is a generator of $\mathbb{G}_1$, h is a generator of $\mathbb{G}_2$ and e is an efficient and non-degenerate bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. We assume that the DDH problem is hard in both $\mathbb{G}_1$ and $\mathbb{G}_2$. In our notation, an Abe commitment to a message $m \in \mathbb{G}_1$ is the tuple $(h^{r_1} h_1^{r_2}, m g_1^{r_2})$, where r_1 and r_2 are random elements in $\mathbb{Z}_q$ and g_1 and h_1 are random elements of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. An Abe commitment to m can be thought of as an ElGamal encryption of m where the first coordinate is hidden in a Pedersen commitment. An opening is of the form $(g_1^{r_1}, m)$ which is valid if $e(g, h^{r_1} h_1^{r_2}) = e(g_1^{r_1}, h) e(m g_1^{r_2}/m, h_1)$.

Polynomial Identity Testing. We will make use of the Schwartz-Zippel lemma to analyze the soundness of our protocol. The lemma gives an efficient method for testing whether a polynomial is equal to zero.

Lemma 1 (Schwartz-Zippel). *Let $f \in \mathbb{Z}_q[X_1, ..., X_n]$ be a non-zero polynomial of total degree $d \geq 0$ over $\mathbb{Z}_q$. Let $S \subseteq \mathbb{Z}_q$ and let $x_1, ..., x_n$ be chosen uniformly at random from S. Then* $\Pr[f(x_1, ..., x_n) = 0] \leq d/|S|$.

3 Commitment Consistent Encryption

We now describe *commitment consistent encryption* (CCE), as defined by Cuvelier *et al.* [4]. The idea is that for any ciphertext, one can derive a commitment to that ciphertext, and the secret key can be used to obtain an opening to that commitment. Furthermore, applied in a voting protocol, the idea is that the voters compute a CC encryption of their ballot, and the authorities derive a commitment to the ciphertext and post this commitment on a public bulletin board. If the commitments are perfectly hiding, they can be used to provide a perfectly private audit trail, which allows anyone to verify the correctness of the count, but does not contain any information about who submitted which ballots.

Definition 1 (CC Encryption [4]). *A commitment consistent encryption scheme Π is a tuple of six efficient algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{DeriveCom}, \mathsf{Open}, \mathsf{Verify})$, defined as follows:*

- $\mathsf{Gen}(1^\lambda)$*: on input a security parameter λ, output a triple (pp, pk, sk) of public parameters, public key and secret key. The public parameter pp is implicitly given as input to the rest of the algorithms.*
- $\mathsf{Enc}_{pk}(m)$*: output a ciphertext c, which is an encryption of a message m in the plaintext space $\mathcal{M}$ (defined by pp) using public key pk.*
- $\mathsf{Dec}_{sk}(c)$*: for a ciphertext c in the ciphertext space $\mathcal{C}$ (defined by pp), output a message m using secret key sk.*
- $\mathsf{DeriveCom}_{pk}(c)$*: From a ciphertext c, output a commitment d using pk.*
- $\mathsf{Open}_{sk}(c)$*: from a ciphertext c, output an auxiliary value a, that can be considered as part of an opening for a commitment d.*
- $\mathsf{Verify}_{pk}(d, m, a)$*: On input a message m and a commitment d wrt. public key pk, and auxiliary value a, output a bit. The algorithm checks that the opening (m, a) is valid wrt. d and pk.*

Correctness. We expect that any commitment consistent encryption scheme satisfies the following correctness property: For any triple (pp, pk, sk) output by Gen, any message $m \in \mathcal{M}$ and any $c = \mathsf{Enc}_{pk}(m)$, it holds with overwhelming probability in the security parameter that $\mathsf{Dec}_{sk}(c) = m$ and $\mathsf{Verify}_{pk}\big(\mathsf{DeriveCom}_{pk}(c), \mathsf{Dec}_{sk}(c), \mathsf{Open}_{sk}(c)\big) = 1$.

The above definition does not guarantee that it is infeasible to create honest-looking CCE ciphertexts that are in fact not consistent. To address this issue, Cuvelier *et al.* [4] define the concept of *validity augmentation* (VA) for CCE schemes. A validity augmentation adds three new algorithms $\mathsf{Expand}, \mathsf{Valid}$ and Strip to the scheme. The Expand algorithm augments the public key for use in the other algorithms. The Valid algorithm takes as input an augmented ciphertext c^{va} along with some proofs of validity. It then checks whether it is possible to derive a commitment and an encryption of an opening to that commitment. The Strip algorithm removes the proofs of validity.

Definition 2 (Validity Augmentation [4]). *A scheme $\Pi^{\mathsf{va}} = (\mathsf{VA.Gen}, \mathsf{VA.Enc}, \mathsf{VA.Dec}, \mathsf{VA.DeriveCom}, \mathsf{VA.Open}, \mathsf{VA.Verify}, \mathsf{Expand}, \mathsf{Strip}, \mathsf{Valid})$ is a validity augmentation of the CCE scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{DeriveCom}, \mathsf{Open}, \mathsf{Verify})$ if the following conditions are satisfied:*

- Augmentation: *$\mathsf{VA.Gen}$ runs Gen to obtain (pp, pk, sk) and outputs an updated triple $(pp^{\mathsf{va}}, pk^{\mathsf{va}}, sk^{\mathsf{va}}) = (pp, \mathsf{Expand}(pk), sk)$.*

- Validity: $\mathit{Valid}_{pk^{\mathsf{va}}}(c^{\mathsf{va}}) = 1$ *for all honestly generated public keys and ciphertexts. In addition, for any PPT adversary $\mathcal{A}$, the following probability is negligible in λ:*

$$\Pr[\mathit{Valid}_{pk^{\mathsf{va}}}(c^{\mathsf{va}}) = 1 \land \neg \mathit{Verify}_{pk}(\mathit{Strip}_{pk^{\mathsf{va}}}(c^{\mathsf{va}})) = 1 \\ \mid c \leftarrow \mathcal{A}(pp^{\mathsf{va}}, pk^{\mathsf{va}}); (pp^{\mathsf{va}}, pk^{\mathsf{va}}, sk^{\mathsf{va}}) \leftarrow \mathit{VA.Gen}]$$

- Consistency: *The values $\mathit{Strip}_{pk^{\mathsf{va}}}(\mathit{VA.enc}_{pk^{\mathsf{va}}}(m))$ and $\mathit{Enc}_{pk}(m)$ are equally distributed for all $m \in \mathcal{M}$, i.e. it is possible to strip a validity augmented ciphertext into a "normal" one. In addition, it holds, for all ciphertexts and keys, that $\mathit{VA.Dec}_{sk^{\mathsf{va}}}(c^{\mathsf{va}}) = \mathit{Dec}_{sk}(\mathit{Strip}_{pk^{\mathsf{va}}}(c^{\mathsf{va}}))$, that $\mathit{VA.Open}_{sk^{\mathsf{va}}}(c^{\mathsf{va}}) = \mathit{Open}_{sk}(\mathit{Strip}_{pk^{\mathsf{va}}}(c^{\mathsf{va}}))$ and that $\mathit{VA.Verify}_{pk^{\mathsf{va}}}(c^{\mathsf{va}}) = \mathit{Verify}_{pk}(\mathit{Strip}_{pk^{\mathsf{va}}}(c^{\mathsf{va}}))$. In other words, the decryption, opening and verification for Π^{va} is consistent with those of Π.*

3.1 The PPATC Encryption System

We now describe an augmented CCE system called PPATC (Perfectly Private Audit Trail with Complex ballots). The system is defined as follows [4]:

- $\mathsf{VA.Gen}(1^\lambda)$: Generate $\Lambda_{sxdh} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h)$ and random generators $g_1 = g^{x_1}, g_2 = g^{x_2} \in \mathbb{G}_1$ and $h_1 \in \mathbb{G}_2$. Now, $(pp, pk, sk) = ((\Lambda_{sxdh}, h_1), (g_1, g_2), (x_1, x_2))$. The augmented key $pk^{\mathsf{va}} = \mathsf{Expand}(pk)$ is computed by adding a description of a hash function $\mathcal{H}$ with range $\mathbb{Z}_q$ to the public key, resulting in the triple $(pp^{\mathsf{va}} = pp, pk^{\mathsf{va}}, sk^{\mathsf{va}} = sk)$.
- $\mathsf{VA.Enc}_{pk^{\mathsf{va}}}(m; r)$: Compute the CCE ciphertext $c = \mathsf{Enc}_{pk}(m; r)$ where $c = (c_1, c_2, c_3, d_1, d_2) = (g^{r_2}, g^{r_3}, g_1^{r_1} g_2^{r_3}, h^{r_1} h_1^{r_2}, m g_1^{r_2})$ and $r = (r_1, r_2, r_3) \in \mathbb{Z}_q^3$. Then compute the following validity proof. Select $s_1, s_2, s_3 \overset{r}{\leftarrow} \mathbb{Z}_q$ and compute $c' = (c'_1, c'_2, c'_3, d'_1) = (g^{s_2}, g^{s_3}, g_1^{s_1} g_2^{s_3}, h^{s_1} h_1^{s_2})$. Compute $\nu_{cc} = \mathcal{H}(pp^{\mathsf{va}}, pk^{\mathsf{va}}, c, c')$, $f_1 = s_1 + \nu_{cc} r_1$, $f_2 = s_2 + \nu_{cc} r_2$ and $f_3 = s_3 + \nu_{cc} r_3$. Let $\sigma_{cc} = (\nu_{cc}, f_1, f_2, f_3)$. The ciphertext is $c^{\mathsf{va}} = (c, \sigma_{cc})$.
- $\mathsf{VA.Dec}_{sk^{\mathsf{va}}}(c^{\mathsf{va}})$: Parse c^{va} as $(c_1, c_2, c_3, d_1, d_2, \sigma_{cc})$ and return $d_2 / c_1^{x_1}$.
- $\mathsf{VA.DeriveCom}_{pk^{\mathsf{va}}}(c^{\mathsf{va}})$: Parse c^{va} as $(c_1, c_2, c_3, d_1, d_2, \sigma_{cc})$ and return (d_1, d_2).
- $\mathsf{VA.Open}_{sk^{\mathsf{va}}}(c^{\mathsf{va}})$: Parse c^{va} as $(c_1, c_2, c_3, d_1, d_2, \sigma_{cc})$ and return $a = c_3 / c_2^{x_2}$.
- $\mathsf{VA.Verify}_{pk^{\mathsf{va}}}(d_1, d_2, m, a)$: Return 1 if $e(g, d_1) = e(a, h) e(d_2/m, h_1)$ and 0 otherwise.
- $\mathsf{Valid}_{pk^{\mathsf{va}}}(c^{\mathsf{va}})$: Parse c^{va} as $(c_1, c_2, c_3, d_1, d_2, \nu_{cc}, f_1, f_2, f_3)$ and check if all elements of c^{va} are properly encoded. Compute $c'_1 = g^{f_2} / c_1^{\nu_{cc}}$, $c'_2 = g^{f_3} / c_2^{\nu_{cc}}$, $c'_3 = g_1^{f_1} g_2^{f_3} / c_3^{\nu_{cc}}$ and $d'_1 = h^{f_1} h_1^{f_2} / d_1^{\nu_{cc}}$. Return 1 only if

$$\nu_{cc} = \mathcal{H}(pp^{\mathsf{va}}, pk^{\mathsf{va}}, c_1, c_2, c_3, d_1, d_2, c'_1, c'_2, c'_3, d'_1, d'_2).$$

- $\mathsf{Strip}_{pk^{\mathsf{va}}}(c^{\mathsf{va}})$: Parse c^{va} as $(c_1, c_2, c_3, d_1, d_2, \sigma_{cc})$ and return the CCE ciphertext $c = (c_1, c_2, c_3, d_1, d_2)$ and the commitment $d = (d_1, d_2)$.

A CCE ciphertext $c = \mathsf{Enc}_{pk}(m; r) = (g^{r_2}, g^{r_3}, g_1^{r_1} g_2^{r_3}, h^{r_1} h_1^{r_1}, m g_1^{r_2})$ can be *re-encrypted*, by multiplying c with the encryption of 1 using randomness $r' = (r'_1, r'_2, r'_3) \in \mathbb{Z}_q^3$. Thus, a ciphertext c', where

$$c' = c \cdot \mathsf{Enc}_{pk}(1; r') = (g^{r_2+r'_2}, g^{r_3+r'_3}, g_1^{r_1+r'_1} g_2^{r_3+r'_3}, h^{r_1+r'_1} h_1^{r_2+r'_2}, m g_1^{r_2+r'_2}),$$

can be thought of as an encryption of m using randomness $r + r'$.

4 Shuffling Commitment Consistent Ciphertexts

In this section, we first describe how the PPATC can be used as a building block in a voting system. We then give concrete protocols for shuffling PPATC ciphertexts and their derived commitments, before describing how to apply the Fiat-Shamir heuristic to make the shuffles non-interactive.

4.1 Using the PPATC scheme in a Voting System

A validity augmented CCE scheme can be applied in an election as follows [4]. First, a setup phase takes place, where the election authorities generate encryption and decryption keys, as well as two bulletin boards PB and SB. The public board PB will contain the public audit trail, while SB will contain encrypted votes, be kept secret by the authorities and will be used to compute the tally. To produce a ballot, each voter encrypts her vote using the PPATC scheme, and sends the resulting ciphertext to the authorities. The ciphertext is stored on SB and the derived commitment is stored on PB.

To preserve privacy, the link between voter and vote must be destroyed, the list of ciphertexts on SB is *shuffled*. A shuffle of a list $\mathbf{v}$ of ciphertexts is a new list $\mathbf{v}'$, such that for all $i = 1, \ldots, n$, $v'_i = v_{\pi(i)} \cdot \mathsf{Enc}_{pk}(1; r_{\pi(i)})$, where $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ is a randomly chosen permutation. Thus, the two lists $\mathbf{v}$ and $\mathbf{v}'$ contain encryptions of the same plaintexts in permuted order. To also provide verifiability, we keep track of the concordance between the ciphertexts on SB and the corresponding commitments on PB. To achieve this, the list of commitments on PB is also shuffled, using the same permutation as for SB.

The lists are shuffled several times, by a series of *mix servers*. It is necessary that each mix server provides a *proof of shuffle*, to prove that he follows the protocol, and that the lists of ciphertexts in fact decrypt to the same plaintexts. For our shuffle protocols we will use the optimised version of the Terelius-Wikström shuffle presented by Haenni *et al.* [8], where a proof of shuffle consists of proving knowledge of the permutation π and the random vector $\mathbf{r}$ used to re-encrypt the ciphertexts. Thus, the tally procedure will proceed as follows:

1. *Stripping*: Algorithms Valid and Strip are run on the ciphertexts stored on SB to obtain a vector $\mathbf{v}$ of n CCE ciphertexts and a vector $\mathbf{d}$ of the corresponding commitments.

2. *Performing the shuffles*: Each mix server selects a random permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$, also defining a permutation matrix M, and computes a commitment $\mathbf{u}$ on that permutation matrix, along with a proof of knowledge of the permutation. The mix server then selects a random vector $\mathbf{r} = ((r_{1,1}, r_{1,2}, r_{1,3}), \cdots, (r_{n,1}, r_{n,2}, r_{n,3}))$ and computes a new vector $\mathbf{v}'$ where $v'_i = v_{\pi(i)} \cdot \mathsf{Enc}_{pk}(1; r_{\pi(i)})$, and $r_{\pi(i)} = (r_{\pi(i),1}, r_{\pi(i),2}, r_{\pi(i),3})$. Let the last two components of each ciphertext v'_i form a vector $\mathbf{d}'$. This vector is posted on PB. Finally, the mix server computes two commitment consistent proofs of shuffle, showing that $\mathbf{v}'$ is a shuffle of $\mathbf{v}$ and $\mathbf{d}'$ is a shuffle of $\mathbf{d}$, with respect to the permutation π.
3. *Decryption of openings:* The authorities verify the proofs and perform a threshold decryption of the ciphertexts in $\mathbf{v}'$. In addition, they run the algorithm Open on these ciphertexts to obtain the auxiliary values for the commitments. The plaintexts and the auxiliary values are posted on PB.

4.2 Proof of Shuffle on the Private Board

We begin with the shuffle on the private board, i.e. the shuffle of the CCE ciphertexts. In the following, let $\mathscr{R}_{com}$ be a relation between the commitment parameters $\gamma, \gamma_1, ..., \gamma_n \in \mathbb{G}_1$, $\mathbf{m}, \mathbf{m}' \in \mathbb{Z}_q^n$ and $t, t' \in \mathbb{Z}_q$ which holds if and only if $\mathsf{Com}_{\gamma,\gamma_1,\ldots,\gamma_n}(\mathbf{m}, t) = \mathsf{Com}_{\gamma,\gamma_1,\ldots,\gamma_n}(\mathbf{m}', t')$ and $\mathbf{m} \neq \mathbf{m}'$. Let $\mathscr{R}_\pi$ be the relation between the commitment parameters $\gamma, \gamma_1, ..., \gamma_n$, a commitment $\mathbf{u} \in \mathbb{G}_1^n$, a permutation matrix $M \in \mathbb{Z}_q^{n\times n}$ and a randomness vector $\mathbf{t} \in \mathbb{Z}_q^n$ which holds only if $\mathbf{u} = \mathsf{Com}_{\gamma,\gamma_1,\ldots\gamma_n}(M, \mathbf{t})$. Let $\mathscr{R}^{shuf}_{ReEnc}(pk, (v_1, ..., v_n), (v'_1, ..., v'_n))(\pi, (r_1, ..., r_n))$, where π is a permutation of the set $\{1, ..., n\}$, be the relation which holds if and only if $v'_i = v_{\pi(i)} \cdot \mathsf{Enc}_{pk}(1; r_{\pi(i)})$ for all $i \in \{1, ..., n\}$.

Theorem 1. *Protocol 1 is a perfectly complete, 4-round special soundness, special honest-verifier zero-knowledge proof of knowledge of the relation* $\mathscr{R}_{com} \vee (\mathscr{R}_\pi \wedge \mathscr{R}^{shuf}_{ReEnc})$.

It is infeasible under the discrete log assumption to find a witness for $\mathscr{R}_{com}$, so Theorem 1 implies a proof of knowledge for $(\mathscr{R}_\pi \wedge \mathscr{R}^{shuf}_{ReEnc})$. To prove the theorem, we now demonstrate the completeness of the protocol, as well as the special soundness extractor and the special honest-verifier zero-knowledge simulator. For brevity we omit parts of the paper proof. The full proof can be found in the full version of our paper [7].

Special Soundness. We will follow the structure of Terelius & Wikström [15] and split the extractor in two parts. In the first part, the basic extractor, we show that for two accepting transcripts with the same $\mathbf{w}$ but different β, we can extract witnesses for certain sub-statements. In the second part, the extended extractor, we show that we can extract a witness to the main statement, given witnesses which hold for these sub-statements, for n different $\mathbf{w}$.

Protocol 1 Interactive ZK-Proof of Shuffle on Private Board

Common Input: A public key pk, a matrix commitment $\mathbf{u}$, commitment parameters $\gamma, \gamma_1, ..., \gamma_n$ and ciphertext vectors $\mathbf{v}, \mathbf{v}' \in (\mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_1)^n$.

Private Input: Permutation matrix $M \in \mathbb{Z}_q^{n \times n}$ and randomness $\mathbf{t} \in \mathbb{Z}_q^n$ such that $\mathbf{u} = \mathsf{Com}(M, \mathbf{t})$. Randomness $\mathbf{r} \in (\mathbb{Z}_q \times \mathbb{Z}_q \times \mathbb{Z}_q)^n$ such that $v'_i = v_{\pi(i)} \cdot \mathsf{Enc}_{pk}(1; r_{\pi(i)})$ for $i = 1, ..., n$.

1: $\mathcal{V}$ chooses a random $\mathbf{w} \in \mathbb{Z}_q^n$ and sends $\mathbf{w}$ to $\mathcal{P}$.

2: $\mathcal{P}$ computes $\mathbf{w}' = (w'_1, ..., w'_n) = M\mathbf{w}$, and randomly chooses $\hat{\mathbf{t}} = (\hat{t}_1, \ldots, \hat{t}_n),\ \hat{\mathbf{z}} = (\hat{z}_1, ..., \hat{z}_n),\ \mathbf{z}' = (z'_1, ..., z'_n) \in \mathbb{Z}_q^n, z_1, z_2, z_3 \in \mathbb{Z}_q$ and $\tilde{\mathbf{z}} = (\tilde{z}_1, \tilde{z}_2, \tilde{z}_3) \in \mathbb{Z}_q^3$. $\mathcal{P}$ defines

$$\bar{t} = \langle \mathbf{1}, \mathbf{t} \rangle,\ \tilde{t} = \langle \mathbf{t}, \mathbf{w} \rangle,\ \hat{t} = \hat{t}_n + \sum_{i=1}^{n-1} \left(\hat{t}_i \prod_{j=i+1}^{n} w'_j \right) \text{ and}$$

$$\mathbf{r}' = \left(\sum_{i=1}^{n} r_{i,1} w_i, \sum_{i=1}^{n} r_{i,2} w_i, \sum_{i=1}^{n} r_{i,3} w_i \right),$$

and sends the following elements to $\mathcal{V}$ (for $i = 1, \ldots, n$):

$$\hat{u}_0 = \gamma_1 \qquad \hat{u}_i = \gamma^{\hat{t}_i} (\hat{u}_{i-1})^{w'_i} \qquad a_1 = \gamma^{z_1} \qquad a_2 = \gamma^{z_2}$$

$$a_3 = \gamma^{z_3} \mathit{\Pi}_{i=1}^{n} \gamma_i^{z'_i} \qquad a_4 = \mathsf{Enc}_{pk}(1; \tilde{\mathbf{z}}) \mathit{\Pi}_{i=1}^{n} (v'_i)^{z'_i} \qquad \hat{a}_i = \gamma^{\hat{z}_i} (\hat{u}_{i-1})^{z'_i}.$$

3: $\mathcal{V}$ chooses a random challenge $\beta \in \mathbb{Z}_q$ and sends β to $\mathcal{P}$.

4: For $i \in \{1, \ldots, n\}$, $\mathcal{P}$ responds with

$$b_1 = z_1 + \beta \cdot \bar{t} \qquad b_2 = z_2 + \beta \cdot \hat{t} \qquad b_3 = z_3 + \beta \cdot \tilde{t}$$

$$\tilde{\mathbf{b}} = \tilde{\mathbf{z}} - \beta \cdot \mathbf{r}' \qquad \hat{b}_i = \hat{z}_i + \beta \cdot \hat{t}_i \qquad b'_i = z'_i + \beta \cdot w'_i.$$

5: $\mathcal{V}$ accepts if and only if, for $i \in \{1, \ldots, n\}$

$$a_1 = (\mathit{\Pi}_{i=1}^{n} u_i / \mathit{\Pi}_{i=1}^{n} \gamma_i)^{-\beta} \cdot \gamma^{b_1} \quad a_2 = (\hat{u}_n / \gamma_1^{\mathit{\Pi}_{i=1}^{n} w_i})^{-\beta} \cdot \gamma^{b_2}$$

$$a_3 = (\mathit{\Pi}_{i=1}^{n} u_i^{w_i})^{-\beta} \cdot \gamma^{b_3} \cdot \mathit{\Pi}_{i=1}^{n} \gamma_i^{b'_i} \qquad a_4 = (\mathit{\Pi}_{i=1}^{n} v_i^{w_i})^{-\beta} \cdot \mathsf{Enc}_{pk}(1; \tilde{\mathbf{b}}) \cdot \mathit{\Pi}_{i=1}^{n} (v'_i)^{b'_i}$$

$$\hat{a}_i = (\hat{u}_i)^{-\beta} \cdot \gamma^{\hat{b}_i} \cdot (\hat{u}_{i-1})^{b'_i}$$

Basic extractor. Given two accepting transcripts

$$(\mathbf{w}, \hat{\mathbf{u}}, a_1, a_2, a_3, a_4, \hat{\mathbf{a}}, \beta, b_1, b_2, b_3, \tilde{\mathbf{b}}, \hat{\mathbf{b}}, \mathbf{b}')$$
$$(\mathbf{w}, \hat{\mathbf{u}}, a_1, a_2, a_3, a_4, \hat{\mathbf{a}}, \beta^*, b_1^*, b_2^*, b_3^*, \tilde{\mathbf{b}^*}, \hat{\mathbf{b}}^*, \mathbf{b}'^*)$$

where $\beta \neq \beta^*$, the basic extractor computes

$$\bar{t} = (b_1 - b_1^*)/(\beta - \beta^*) \qquad \hat{t} = (b_2 - b_2^*)/(\beta - \beta^*) \qquad \tilde{t} = (b_3 - b_3^*)/(\beta - \beta^*)$$
$$\hat{\mathbf{t}}' = (\hat{\mathbf{b}} - \hat{\mathbf{b}}^*)/(\beta - \beta^*) \qquad \mathbf{w}' = (\mathbf{b}' - \mathbf{b}'^*)/(\beta - \beta^*) \qquad \mathbf{r}' = (\tilde{\mathbf{b}} - \tilde{\mathbf{b}^*})/(\beta - \beta^*)$$

We will prove that

$$\mathit{\Pi}_{i=1}^n u_i = \mathsf{Com}(\mathbf{1}, \bar{t}), \quad \mathit{\Pi}_{i=1}^n u_i^{w_i} = \mathsf{Com}(\mathbf{w}', \tilde{t}), \quad \mathit{\Pi}_{i=1}^n v_i^{w_i} = \mathit{\Pi}_{i=1}^n (v_i')^{w_i'} \cdot \mathsf{Enc}_{pk}(1; -\mathbf{r}'),$$
$$\hat{u}_i = \mathsf{Com}_{\gamma, \hat{u}_{i-1}}(w_i', \hat{t}_i') \quad \text{and} \quad \hat{u}_n = \mathsf{Com}_{\gamma, \gamma_1}\left(\mathit{\Pi}_{i=1}^n w_i, \hat{t}\right).$$

The proof consists of algebraic manipulations:

$$\mathit{\Pi}_{i=1}^n u_i = \left(\frac{(\mathit{\Pi}_{i=1}^n u_i)^\beta \cdot a_1}{(\mathit{\Pi}_{i=1}^n u_i)^{\beta^*} \cdot a_1} \right)^{\frac{1}{\beta - \beta^*}} = \gamma^{\frac{b_1 - b_1^*}{\beta - \beta^*}} \cdot \mathit{\Pi}_{i=1}^n \gamma_i = \mathsf{Com}(\mathbf{1}, \bar{t}).$$

$$\mathit{\Pi}_{i=1}^n u_i^{w_i} = \left(\frac{(\mathit{\Pi}_{i=1}^n u_i^{w_i})^\beta \cdot a_3}{(\mathit{\Pi}_{i=1}^n u_i^{w_i})^{\beta^*} \cdot a_3} \right)^{\frac{1}{\beta - \beta^*}} = \gamma^{\frac{b_3 - b_3^*}{\beta - \beta^*}} \cdot \mathit{\Pi}_{i=1}^n \gamma_i^{\frac{b_i' - b_i'^*}{\beta - \beta^*}} = \mathsf{Com}(\mathbf{w}', \tilde{t}).$$

$$\begin{aligned} \mathit{\Pi}_{i=1}^n v_i^{w_i} &= \left(\frac{(\mathit{\Pi}_{i=1}^n v_i^{w_i})^\beta \cdot a_4}{(\mathit{\Pi}_{i=1}^n v_i^{w_i})^{\beta^*} \cdot a_4} \right)^{\frac{1}{\beta - \beta^*}} = \mathit{\Pi}_{i=1}^n (v_i')^{\frac{b_i' - b_i'^*}{\beta - \beta^*}} \cdot \mathsf{Enc}_{pk}\left(1; \frac{\tilde{\mathbf{b}} - \tilde{\mathbf{b}^*}}{\beta - \beta^*}\right) \\ &= \mathit{\Pi}_{i=1}^n (v_i')^{w_i'} \cdot \mathsf{Enc}_{pk}(1; -\mathbf{r}'). \end{aligned}$$

$$\hat{u}_i = \gamma^{\frac{\hat{b}_i - \hat{b}_i^*}{\beta - \beta^*}} \cdot (\hat{u}_{i-1})^{\frac{b_i' - b_i'^*}{\beta - \beta^*}} = \gamma^{\hat{t}_i'} \cdot (\hat{u}_{i-1})^{w_i'} = \mathsf{Com}_{\gamma, \hat{u}_{i-1}}(w_i', \hat{t}_i').$$

$$\hat{u}_n = \gamma^{\frac{b_2 - b_2^*}{\beta - \beta^*}} \cdot \gamma_1^{\mathit{\Pi}_{i=1}^n w_i} = \gamma^{\hat{t}} \cdot \gamma_1^{\mathit{\Pi}_{i=1}^n w_i} = \mathsf{Com}_{\gamma, \gamma_1}(\mathit{\Pi}_{i=1}^n w_i; \hat{t}).$$

Thus, all the equations are satisfied.

Extended Extractor. The extended extractor takes, for one statement, n different witnesses extracted by the basic extractor, and produces a witness for the main statement. Let $\bar{\mathbf{t}}, \hat{\mathbf{t}}, \tilde{\mathbf{t}} \in \mathbb{Z}_q^n$, $\mathbf{r}' \in (\mathbb{Z}_q \times \mathbb{Z}_q \times \mathbb{Z}_q)^n$ and $\hat{T}', W' \in \mathbb{Z}_q^{n \times n}$ be the collective output from the n runs of the basic extractor, extracted from challenges $W \in \mathbb{Z}_q^{n \times n}$. Let W_j be the jth column of W, i.e. the challenge vector from the jth run of the basic extractor. The challenge vectors are sampled from a uniform distribution, but since the cheating prover may not succeed with uniform probability for all challenge vectors, the final distribution of challenge vectors is non-uniform. However, since the adversary has a significant probability of success, any set of challenge vectors with a significant success probability must be much larger than the set of non-invertible matrices. It follows that the columns of W will be linearly independent with overwhelming probability.

Thus, W will, with overwhelming probability, have an inverse. We call this inverse A. For such matrix A, we have that WA_k is the kth standard unit vector in $\mathbb{Z}_q^n$, where A_k is the kth column of A. We see that

$$\begin{aligned} u_k &= \mathit{\Pi}_{i=1}^n u_i^{WA_k} = \mathit{\Pi}_{i=1}^n \left(\mathit{\Pi}_{j=1}^n u_i^{W_{i,j} A_{j,k}} \right) = \mathit{\Pi}_{j=1}^n \mathsf{Com}(W_j', \tilde{t}_j)^{A_{j,k}} \\ &= \mathit{\Pi}_{j=1}^n \mathsf{Com}(W_j' A_{j,k}, \tilde{t}_j A_{j,k}) = \mathsf{Com}(W' A_k, \langle \tilde{\mathbf{t}}, A_k \rangle). \end{aligned}$$

Thus, we can open $\mathbf{u}$ to a matrix M, where $M_k = W'A_k$ is committed to using randomness $\langle \tilde{\mathbf{t}}, A_k \rangle$. We expect M to be a permutation matrix. If it is not, we can find a witness breaking the binding property of the commitment scheme. We extract this witness in two different ways, depending on whether or not $M\mathbf{1} = \mathbf{1}$.

If $M\mathbf{1} \neq \mathbf{1}$, let $\mathbf{w}'' = M\mathbf{1}$. We note that $\mathbf{w}'' \neq \mathbf{1}$ and that $\mathsf{Com}(\mathbf{1}, \bar{t}_j) = \mathit{\Pi}_{i=1}^n u_i = \mathsf{Com}(\mathbf{w}'', \tilde{\mathbf{t}}A)$, meaning that we have found a witness violating the binding property of the commitment scheme.

Now assume that $M\mathbf{1} = \mathbf{1}$. Terelius & Wikström [15] prove that M is a permutation matrix if and only if $M\mathbf{1} = \mathbf{1}$ and $\mathit{\Pi}_{i=1}^n \langle M_i, \mathbf{x} \rangle = \mathit{\Pi}_{i=1}^n x_i$ for a vector $\mathbf{x} \in \mathbb{Z}_q^n$ of independent elements. This fact, along with the Schwartz-Zippel lemma and the assumptions that $M\mathbf{1} = \mathbf{1}$ and that M is not a permutation matrix, implies that there exists, with overwhelming probability, some $j \in \{1, ..., n\}$ such that $\mathit{\Pi}_{i=1}^n \langle M_{i,*}, W_j \rangle - \mathit{\Pi}_{i=1}^n W_{i,j} \neq 0$. As this is true with overwhelming probability, we assume that it is true and rewind if not. Now let $\mathbf{w}'' = MW_j$. Note that $\mathit{\Pi}_{i=1}^n W_{i,j}' = \mathit{\Pi}_{i=1}^n W_{i,j}$ and that $\mathit{\Pi}_{i=1}^n W_{i,j} \neq \mathit{\Pi}_{i=1}^n w_i''$. The equality follows from the base statements, and the inequality follows from the Schwartz-Zippel lemma and the definition of $\mathbf{w}''$. Together, these facts imply that $\mathbf{w}'' \neq W_j'$.

We also see that $\mathsf{Com}(W_j', \tilde{t}_j) = \mathit{\Pi}_{i=1}^n u_i^{W_{i,j}} = \mathsf{Com}(\mathbf{w}'', \langle \tilde{\mathbf{t}}A, W_j \rangle)$. Since $\mathbf{w}'' \neq W_j'$, this means that we have found a witness violating the binding property of the commitment scheme. We conclude that either M is a permutation matrix, or the binding property of the commitment scheme does not hold. We conclude further that we either violate the binding property of the commitment scheme, or we have that $\mathbf{w}'' = W_j$, meaning that $W_j' = MW_j$, for all $j \in \{0, ..., n\}$.

4.3 Proof of Shuffle on the Public Board

A verifiable shuffle for the public board is given in Protocol 2. Note that it is very similar to the shuffle in Protocol 1. The difference is that on the public board, the shuffle is performed on the two last components of each ciphertext, rather than on the full ciphertext. Let $\mathscr{R}_\pi$ and $\mathscr{R}_{com}$ be as in Section 4.2. Let $\mathscr{R}_{ReRand}^{shuf}(pk, \mathbf{d}, \mathbf{d}')(\pi, \mathbf{r}')$, where π is a permutation on $\{1, \ldots, n\}$, be the relation which holds if $d_i' = \mathsf{ReRand}(d_{\pi(i)}; r_{\pi(i)}')$ for all $i \in \{1, \ldots, n\}$, where $\mathsf{ReRand}(d_i; r_i') = (h^{r_{i,1}+r_{i,1}'} h_1^{r_{i,2}+r_{i,2}'}, m g_1^{r_{i,2}+r_{i,2}'})$ for $d_i = (h^{r_{i,1}} h_1^{r_{i,2}}, m g_1^{r_{i,2}})$ and random $\mathbf{r}, \mathbf{r}' \in (\mathbb{Z}_q \times \mathbb{Z}_q)^n$.

Theorem 2. *Protocol 2 is a perfectly complete, 4-round special soundness, special honest-verifier zero-knowledge proof of knowledge of the relation* $\mathscr{R}_{com} \vee (\mathscr{R}_{\pi} \wedge \mathscr{R}^{shuf}_{ReRand})$.

The proof is very similar to the proof of Theorem 1 and will be omitted.

4.4 Applying the Fiat-Shamir Heuristic

We now describe how we can make the shuffle non-interactive, by applying the Fiat-Shamir heuristic [5]. The main idea is to replace the challenges sent by the verifier (in step 1 and 3) by a call to some hash function, making the challenges look random. This is straight-forward, but we do not want to run the argument twice, once for the public board and once for the private board. We want to have only one computation. It is easy to see that the interactive public board argument can be extracted from the interactive private board argument, but applying Fiat-Shamir is not straight-forward now, since different knowledge is available in the two cases.

The idea is to use a nested hash function for the private board argument, and then provide the inner hash value as part of the public board argument. This allows us to extract the public board argument from the private board argument by replacing the knowledge that is not present on the public board by their hash value. In order to ensure that no knowledge leaks, we actually commit to the hash of the private values, so that we can prove that the hash value does not contain any information about the private values. This is safe, since commitments are binding.

To obtain $\mathbf{w}$, we first hash the parts of the common input on the private board that is not part of the common input on the public board, i.e. the first three components of the CCE ciphertexts. We then commit to this hash, and hash the commitment along with the part of the common input that is also present on the public board. The challenge $\mathbf{w}$ is set to be this second hash value. The commitment is posted on the public board and opened on the private board.

The challenge β is obtained in a similar manner. We first hash the information on the private board that is not present on the public board, commit to this hash, post the commitment on the public board and then open the commitment on the private board. Further, the commitment is hashed along with the information on the private board that is also present on the public board. This hash is set to be the challenge value β.

5 Machine Checked Proof

Having given a paper proof of the mixnet we now turn our attention to the machine checked proof. One approach would be to codify the above paper proof in an interactive theorem prover. However, codifying such proofs is a complex process, so instead we reuse previous work. The idea is that our variant of the mixnet has a machine checked proof. The gap is that the mixnet is not proved

Protocol 2 Interactive ZK-Proof of Shuffle on Public Board

Common Input: A public key pk, a matrix commitment $\mathbf{u}$, commitment parameters $\gamma, \gamma_1, ..., \gamma_n$ and vectors $\mathbf{d}, \mathbf{d}' \in (\mathbb{G}_2 \times \mathbb{G}_1)^n$.

Private Input: Permutation matrix $M \in \mathbb{Z}_q^{n \times n}$ and randomness $\mathbf{t} \in \mathbb{Z}_q^n$ such that $\mathbf{u} = \mathsf{Com}(M, \mathbf{t})$. Randomness $\mathbf{r} \in (\mathbb{Z}_q \times \mathbb{Z}_q)^n$ such that $d'_i = \mathsf{ReRand}(d_{\pi(i)}, r_{\pi(i)})$ for $i = 1, ..., n$.

1: $\mathcal{V}$ chooses a random $\mathbf{w} \in \mathbb{Z}_q^n$ and sends $\mathbf{w}$ to $\mathcal{P}$.

2: $\mathcal{P}$ computes $\mathbf{w}' = (w'_1, ..., w'_n) = M\mathbf{w}$, and randomly chooses $\hat{\mathbf{t}} = (\hat{t}_1, \ldots, \hat{t}_n)$, $\hat{\mathbf{z}} = (\hat{z}_1, ..., \hat{z}_n)$, $\mathbf{z}' = (z'_1, ..., z'_n) \in \mathbb{Z}_q^n, z_1, z_2, z_3 \in \mathbb{Z}_q$ and $\tilde{\mathbf{z}} = (\tilde{z}_1, \tilde{z}_2) \in \mathbb{Z}_q^2$. $\mathcal{P}$ defines

$$\bar{t} = \langle \mathbf{1}, \mathbf{t} \rangle,\ \tilde{t} = \langle \mathbf{t}, \mathbf{w} \rangle,\ \hat{t} = \hat{t}_n + \sum_{i=1}^{n-1} \left(\hat{t}_i \prod_{j=i+1}^{n} w'_j \right) \text{ and}$$

$$\mathbf{r}' = \left(\sum_{i=1}^{n} r_{i,1} w_i, \sum_{i=1}^{n} r_{i,2} w_i \right),$$

and sends the following elements to $\mathcal{V}$ (for $i = 1, \ldots, n$):

$$\hat{u}_0 = \gamma_1 \qquad \hat{u}_i = \gamma^{\hat{t}_i} (\hat{u}_{i-1})^{w'_i} \qquad a_1 = \gamma^{z_1} \qquad a_2 = \gamma^{z_2}$$

$$a_3 = \gamma^{z_3} \mathit{\Pi}_{i=1}^{n} \gamma_i^{z'_i} \qquad a_4 = (h^{\tilde{z}_1} h_1^{\tilde{z}_2}, g_1^{\tilde{z}_2}) \mathit{\Pi}_{i=1}^{n} (d'_i)^{z'_i}$$

$$\hat{a}_i = \gamma^{\hat{z}_i} (\hat{u}_{i-1})^{z'_i}.$$

3: $\mathcal{V}$ chooses a random challenge $\beta \in \mathbb{Z}_q$ and sends β to $\mathcal{P}$.

4: For $i \in \{1, \ldots, n\}$, $\mathcal{P}$ responds with

$$b_1 = z_1 + \beta \cdot \bar{t} \qquad b_2 = z_2 + \beta \cdot \hat{t} \qquad b_3 = z_3 + \beta \cdot \tilde{t}$$

$$\tilde{\mathbf{b}} = \tilde{\mathbf{z}} - \beta \cdot \mathbf{r}' \qquad \hat{b}_i = \hat{z}_i + \beta \cdot \hat{t}_i \qquad b'_i = z'_i + \beta \cdot w'_i.$$

5: $\mathcal{V}$ accepts if and only if, for $i \in \{1, \ldots, n\}$

$$a_1 = (\mathit{\Pi}_{i=1}^{n} u_i / \mathit{\Pi}_{i=1}^{n} \gamma_i)^{-\beta} \cdot \gamma^{b_1} \quad a_2 = (\hat{u}_n / \gamma_1^{\mathit{\Pi}_{i=1}^{n} w_i})^{-\beta} \cdot \gamma^{b_2}$$

$$a_3 = (\mathit{\Pi}_{i=1}^{n} u_i^{w_i})^{-\beta} \cdot \gamma^{b_3} \cdot \mathit{\Pi}_{i=1}^{n} \gamma_i^{b'_i}$$

$$a_4 = (\mathit{\Pi}_{i=1}^{n} d_i^{w_i})^{-\beta} \cdot (h^{\tilde{b}_1} h_1^{\tilde{b}_2}, g_1^{\tilde{b}_2}) \cdot \mathit{\Pi}_{i=1}^{n} (d'_i)^{b'_i}$$

$$\hat{a}_i = (\hat{u}_i)^{-\beta} \cdot \gamma^{\hat{b}_i} \cdot (\hat{u}_{i-1})^{b'_i}$$

for our particular encryption scheme. But the existing proof applies to a large class of encryption schemes. We need only prove that our scheme is in this class, after which we know that the general results also applies to our concrete mixnet.

For the machine checked proof we will make use of the interactive theorem prover Coq. Our work expands upon Haines *et al.* [10]; who demonstrated how interactive theorem provers and code extraction can be used to gain much higher confidence in the outcome of elections; they achieved this by using the interactive theorem prover Coq and its code extraction facility to produce verifiers, for verifiable voting schemes, with the verifiers proven to be cryptographically correct. They also showed that it was possible to verify the correctness (completeness, soundness and zero-knowledge) of a proof of correct shuffle. Their work was subsequently expanded upon by [9] who removed a number of limitations in the original work and expanded the result. Specifically they proved that for any encryption scheme that falls within a class, which they formally defined, it can be securely mixed in the optimised variant of Wikström's mixnet. We exploit this result by proving that PPATC falls within this class and hence can be verifiably mixed by Wikström's mixnet. Note that the mixnet generated in the Coq code is equivalent to Protocol 1.

In the rest of this section we will present our work in standard notation. Interested readers can find the Coq code at `https://github.com/gerlion/secure-e-voting-with-coq`. We begin by proving that the ciphertext space is a group. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ represent the elements of the two groups of the bilinear pairing both of which are of prime order p. We let the set S of the ciphertext space equal $\mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_1$. All operations are performed pairwise and the group axioms are satisfied trivially.

We then show that the ciphertext group is isomorphic to a vector space over the field of integers modulo p. This follows directly from the fact that two groups of the same order are themselves isomorphic to vector spaces over the field of integers modulo p. We are now ready to define the encryption scheme. Beyond the groups already mentioned we denote the field of integers modulo p as $\mathbb{F}$.

Key generation space := $\mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_2 \times \mathbb{F} \times \mathbb{F}$.
Public key space := $\mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_2 \times \mathbb{G}_1 \times \mathbb{G}_1$.
Secret key space := $\mathbb{F} \times \mathbb{F}$.
Message space := $\mathbb{G}_1$.
Randomness space := $\mathbb{F} \times \mathbb{F} \times \mathbb{F}$.
Key generation := On input (g, h, h_1, x_1, x_2) from key generation space output public key $(g, h, h_1, g^{x_1}, g^{x_2})$ and secret key (x_1, x_2)
Encryption := On input public key (g, h, h_1, y_1, y_2), message m, and randomness (r_1, r_2, r_3) and output ciphertext $(g^{r_1}, g^{r_2}, y_2^{r_2} g^{r_3}, h^{r_3} h_1^{r_1}, y_1^{r_1} m)$.
Decryption := Given secret key (x_1, x_2) and ciphertext $(c_1, c_2, c_3, c_4, c_5)$ and return $c_5/c_1^{x_1}$.

To show that the encryption scheme can be correctly mixed we need to prove the three theorems stated below. We also require the vector space properties for the spaces defined above, see the Coq code for a formal definition of these properties.

```
Lemma correct : forall (kgr : KGR)(m : M)(r : Ring.F),
let (pk,sk) := keygen kgr in
dec sk (enc pk m r) = m.
```

Theorem 3. *Correctness:* $\forall kgr \in$ *Key generation space*, $m \in$ *Message space*, $r \in$ *Randomness space*, $(pk, sk) =$ *Key generation*(kgr),
Decryption$(sk$ *Encryption*$(pk\ m\ r)) = m$.

The correctness of PPATC follows directly from the correctness of ElGamal.

```
Lemma homomorphism : forall (pk : PK)(m m' : M)(r r' : Ring.F),
C.Gdot (enc pk m' r')(enc pk m r) =
enc pk (Mop m m') (Ring.Fadd r r').
```

Theorem 4. *Homomorphism:* $\forall pk \in$ *Public key space*, $m\ m' \in$ *Message space*,
$r\ r' \in$ *Randomness space*,
Encryption$(pk\ m\ r) \times$ *Encryption*$(pk\ m'\ r') =$ *Encryption*$(pk\ (m \cdot m')\ (r * r'))$

The homomorphic property of PPATC follows from the homomorphic properties of ElGamal and Abe *et al.*'s commitments.

```
Lemma encOfOnePrec : forall (pk : PK)(a : Ring.F)(b: F),
(VS.op (enc pk Mzero a) b) = enc pk Mzero (MVS.op3 a b).
```

Theorem 5. *Encryption of one preserved:* $\forall pk \in$ *Public key space*,
$r\ r' \in$ *Randomness space*, *Encryption*$(pk\ 1\ a)^b =$ *Encryption*$(pk\ 1\ (a * b))$

To see that this property holds, first consider a PPATC ciphertext encrypting zero $(g^{r_1}, g^{r_2}, y_2^{r_2} g^{r_3}, h^{r_3} h_1^{r_1}, y_1^{r_1})$. Now observe that raising it to any power a is an encryption of one with randomness $(r_1 a, r_2 a, r_3 a)$, $(g^{r_1}, g^{r_2}, y_2^{r_2} g^{r_3}, h^{r_3} h_1^{r_1}, y_1^{r_1})^a = (g^{r_1 a}, g^{r_2 a}, y_2^{r_2 a} g^{r_3 a}, h^{r_3 a} h_1^{r_1 a}, y_1^{r_1 a})$.

This suffices for a proof that the PPATC scheme can be safely mixed by the optimised variant of the Wikström's mixnet. Readers will have noted that we proved the scheme for any pair of groups with the same prime order. Technically, we didn't even require that there exists a billinear pairing between them, though this would be required to get the verifiable component of the Abe *et al.* commitments to work. The current work could be extracted into OCaml code and appropriate groups provided to check election transcripts. However, further work is ongoing in Coq to allow these groups to be instantiated within Coq.

6 Conclusion

We have given a paper proof for a variant of the optimised Wikström's mixnet for the PPATC encryption scheme. This is a useful result for anyone wanting to build an efficient e-voting scheme with everlasting privacy which can handle arbitrary ballots. In addition we provide a machine checked proof of the mixnet.

Acknowledgments. This work was supported by the Luxembourg National Research Fund (FNR) and the Research Council of Norway for the joint project SURCVS.

References

1. Abe, M., Haralambiev, K., Ohkubo, M.: Group to group commitments do not shrink. In: EUROCRYPT. LNCS, vol. 7237, pp. 301–317. Springer (2012)
2. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM **24**(2), 84–88 (1981)
3. Cramer, R., Franklin, M.K., Schoenmakers, B., Yung, M.: Multi-autority secret-ballot elections with linear work. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 1070, pp. 72–83. Springer (1996)
4. Cuvelier, E., Pereira, O., Peters, T.: Election verifiability or ballot privacy: Do we need to choose? In: Crampton, J., Jajodia, S., Mayes, K. (eds.) ESORICS 2013. LNCS, vol. 8134, pp. 481–498. Springer, Heidelberg, Germany, Egham, UK (Sep 9–13, 2013). https://doi.org/10.1007/978-3-642-40203-6_27
5. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 1987). https://doi.org/10.1007/3-540-47721-7_12
6. Fujisaki, E., Okamoto, T.: Statistical zero knowledge protocols to prove modular polynomial relations. In: CRYPTO. Lecture Notes in Computer Science, vol. 1294, pp. 16–30. Springer (1997)
7. Gjøsteen, K., Haines, T., Solberg, M.R.: Efficient mixing of arbitrary ballots with everlasting privacy: How to verifiably mix the PPATC scheme (full version). Cryptology ePrint Archive, Report 2020/1331 (2020), https://eprint.iacr.org/2020/1331
8. Haenni, R., Locher, P., Koenig, R.E., Dubuis, E.: Pseudo-code algorithms for verifiable re-encryption mix-nets. In: Brenner, M., Rohloff, K., Bonneau, J., Miller, A., Ryan, P.Y.A., Teague, V., Bracciali, A., Sala, M., Pintore, F., Jakobsson, M. (eds.) FC 2017 Workshops. LNCS, vol. 10323, pp. 370–384. Springer, Heidelberg, Germany, Sliema, Malta (Apr 7, 2017)
9. Haines, T., Goré, R., Sharma, B.: Did you mix me? Formally verifying verifiable mix nets in voting. In: 2021 IEEE Symposium on Security and Privacy, SP 2021, San Jose, CA, USA, May 23-27, 2021. IEEE (2021)
10. Haines, T., Goré, R., Tiwari, M.: Verified verifiers for verifying elections. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019. pp. 685–702. ACM (2019)
11. Haines, T., Gritti, C.: Improvements in everlasting privacy: Efficient and secure zero knowledge proofs. In: E-VOTE-ID. Lecture Notes in Computer Science, vol. 11759, pp. 116–133. Springer (2019)
12. Moran, T., Naor, M.: Split-ballot voting: Everlasting privacy with distributed trust. ACM Trans. Inf. Syst. Secur. **13**(2), 16:1–16:43 (2010)
13. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: EUROCRYPT. pp. 223–238. Springer (1999)
14. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: CRYPTO. LNCS, vol. 576, pp. 129–140. Springer (1991)
15. Terelius, B., Wikström, D.: Proofs of restricted shuffles. In: Bernstein, D.J., Lange, T. (eds.) AFRICACRYPT 10. LNCS, vol. 6055, pp. 100–113. Springer, Heidelberg, Germany, Stellenbosch, South Africa (May 3–6, 2010)