A Publicly-Verifiable Mix-net with Everlasting Privacy Towards Observers

Denise Demirel[1] and Jeroen van de Graaf[2]

[1] Department of Computer Science, Cryptography and Computer Algebra Group, Technische Universität Darmstadt, Darmstadt, Germany. E-mail: ddemirel@cdc.informatik.tu-darmstadt.de
[2] Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, CEP 31270-901, Brazil. E-mail: jvdg@dcc.ufmg.br

Abstract. In this paper we present a novel, publicly verifiable mixing scheme which has everlasting privacy towards observers: all the information published on the bulletin board by the mixes (audit information etc) reveals no information about the identity of any of the messages published. The correctness of the mixing process is statistical: even if all authorities conspire, they cannot change the contents of any message without being detected with overwhelming probability.
We accomplish this by encoding the messages submitted using so-called Pedersen commitments. Decoding (opening) these is possible because we create a parallel mix-net run by the same mixes to which the public has no access. This private mix-net uses the same permutations as the public one, but uses homomorphic encryption, which is used to send auxiliary information (messages, decommitment values) through the mix-net to allow decoding.

1 Introduction

1.1 Motivation

Mix-nets were introduced by David Chaum in 1981 [8] to allow anonymous communication within a network. The basic functionality of a mix-net is to process a set of input messages so that the content remains unchanged while any link between a single input and its associated output is removed. Chaum pointed out that this technique is of interest for several applications where privacy plays an important role like, for instance, eVoting and electronic mail services. The original approach is the so-called decryption mix-net which makes use of public key cryptography. In 1993 Park et al. introduced the re-encryption mix-net [24] which takes advantage of the homomorphic properties of some public key cryptoschemes. The latter approach has been used in many eVoting systems and is still a component of the most current proposals.

Important for use of mix-nets in application with high requirements regarding integrity, is that the whole process can be verified by any third party. Therefore, all recent variations of mix-nets publish additional information to allow verification of its correct functioning by any observer. This information is encrypted using some public key algorithm (often ElGamal or Paillier) based on number-theoretic problems which are assumed to be computationally hard. However, attacks like brute-force that consists of simply checking all possible solution would still be successful (although a computational limited attacker needs decades).

This raises the question of how long certain information must remain secret. One good example for an application which processes sensitive data and thus needs information-theoretical security is electronic voting. To offer the voter the possibility to verify the accuracy of the election outcome, additional information of the tallying process is published. So many systems store the votes cast in encrypted form, publish all received votes, and provide a receipt to the voter which is linked to one entry. This enables the voter to check whether the entry confirms with the receipt and thus she can verify whether the vote was recorded as cast.

To protect the voter's privacy, the published ciphertexts are made anonymous before decrypting. But both the encryption, as well as the anonymisation process, provide computational privacy only. As soon as the underlying cryptosystem is broken, the encrypted published votes (associated to the receipt) can be decrypted. With the cost for storing information becoming less, the fact that the encrypted votes have been published on the internet makes it virtually impossible to remove this data later on. And processing power increases continually following Moore's law, so all an attacker needs to do is to download the published information and to wait until she can decrypt the information.

The existence of such an attack could have farreaching consequences. If mix-nets are used for publishing highly sensitive information or for elections, then the disclosure of the messages (or ballots), even decades later, can still be highly embarrassing. For instance, think of a 65-year old presidential candidate whose voting behavior when he or she was 20 becomes public. It might ruin his or her chances.

The mere possibility of this type of disclosure could have a negative effect: individuals might feel restricted about what they say, and voters about whom they vote for. Certainly there is also a legal argument here: in all democratic countries privacy of the ballot is protected by law and is eternal; no lawyer or judge should approve a system that has the property that privacy of the ballot might be violated several decades after the election was held. So mixes that only provide computational privacy should not be used for elections.

We therefore believe that developing solutions for verifiable mixing offering everlasting (or unconditional or information-theoretic) privacy is an important research question.

1.2 High-level description of our mixing process

In this paper we show that it is possible to mix Pedersen commitments[3]. We use these commitments to encode[4] a message t as follows: $u = \alpha^s \beta^t$. Here s is the (randomly chosen) decommitment value. Any mix A can recode (or rerandomise) such a commitment by multiplying it with α^{s_a}, that is, $u_A = \alpha^{s_a} u$. It can do so without knowing the value encoded. Observe that the new decommitment value of u_a is equal to $d_a = s_a + s$, where addition is modulo the order of the multiplicative group.

However, a problem with shuffling these encodings is the following: in order to open them, one needs to know what the decommitment values s are. Unlike conventional encryptions, in which the message is uniquely determined, u_A can represent any message, and its interpretation depends on the decommitment value.

The solution is to send these decommitment values as auxiliary information through a parallel mix network to which the public has no access. Any recoding $u_A = \alpha^{s_a} u$ has a matching entry $v_A = E(s_a)v$, where $v = E(s)$ and E is a suitable homomorphic encryption scheme. In this paper we will be using Paillier encryption.

So essentially we use two tightly synchronized mix networks run by the same mixes: one which is mixing Pedersen commitments and which is fully public. And a second one which uses homomorphic encryption to which the public has no access, in order to transport the decommitment values. Observe that the mixing permutations used in the private mix-net must be identical to those used in public mix-net for the process to work.

Then, after the third mix C has published its data, v_c is jointly decrypted using the Paillier private key held by the key trustee which, due to the homomorphic property, yields $d = s + s_A + s_B + s_c$, the decommitment value of $u_c = \alpha^{s+s_A+s_B+s_c} \beta^t$. Now recovering t corresponds to solving a discrete log problem. If t is small (about 50-60 bits) there exist sophisticated guessing techniques. If t is large, we require the user to send $z = E(t)$ also through the private mix net. In this case, the authorities must re-encrypt z to avoid linking, resulting in z_C. This value is jointly decrypted, yielding t.

The scheme sketched in the above paragraphs already provides everlasting privacy towards observers. But it has one drawback: the first mix, to whom the user submits her message, gets to see $E(s)$. So when Paillier gets broken, this mix, if dishonest, could reconstruct the user's message. If this is a problem, an obvious solution is to split the input in two (or more) parts, submit each part to a separate sequence of mix-nets, and have the parts recombined, decrypted, decoded and published by a special publication authority. This is the protocol we present in Section 2.

[3] Though commonly referred to as Pedersen commitments, these constructions were actually first presented in [11].

[4] Since we actually use bit commitments to encode messages, we use this term instead of using the verb *to commit*.

1.3 Efficient verification of correctness

In order to be fully transparent, it is essential that the individual mixes prove that their shuffle is correct, but often these proofs cause a performance bottleneck. Thus, many mathematical Zero-Knowledge (ZK) proofs (e.g. Neff [23], Furukawa [14], Wikström [29], and Bayer and Groth [4]), generic verification methods (e.g. Jakobsson et al.[21], Golle et al. [16], Boneh and Golle [5], and Sako and Kilian [27]) and such which combine both approaches (e.g. Allepuz and Castelló [3], Demirel et al. [13]) have been proposed.

In practice the generic verification methods are the most common ones (e.g. part of Helios [2], Civitas [12], Prêt à Voter [26], and Scantegrity [10]). A possible reason for their success might be the simple implementation and the good efficiency compared to mathematical ZK proofs, although their accuracy level is much lower. Thus, for the last election in Norway in 2011, an approach combining several generic verification methods with a mathematical ZK proof was used to provide a good trade-off between these two values [15, 3].

Nevertheless, finding an efficient mathematical ZK proof is still of interest. Good progress in this direction is made by Bayer and Groth [4]. Their approach provides both, showing the correctness of a shuffle by a mathematical Zero-Knowledge argument while at the same time providing good efficiency. A further interesting approach is generating a compact proof which can be used to show the accuracy of the whole shuffling process [6]. However, so far it has not been evaluated which information a computationally unbounded party can gain from these mathematical Zero-Knowledge based verification methods.

In this paper we show how the mix-net can be verified using a straightforward cut-and-choose verification method, in order to show the concepptual simplicity of our approach. As an optimization, we then show a combination between this approach and RBV [13]. The evaluation whether [4] and [6] are applicable for our newly proposed everlasting mixing process will be part of future work.

1.4 Summary of our results

The protocol we propose in this paper has the following properties:

Correctness Even if all authorities conspire, they cannot change the contents of any message without being detected with overwhelming probability, provided that they cannot break the discrete logarithm problem before the process has ended and has been certified.

Universal verifiability Any observer can verify that the mixing process was performed honestly.

Individual verifiability Each user can convince herself that her message was included in the input batch.

Everlasting privacy All information published on the bulletin board reveals no (Shannon) information about the identity of sender of any of the messages published.

Robustness If all authorities follow the protocol correctly, then the protocol always terminates successfully. If one authority cheats, it will get caught with overwhelming probability.

Despite the vast literature on mix-nets, we are not aware of any publication that achieves everlasting privacy for mixing. This property is achieved in the Split-Ballot *election* system of Moran and Naor[22], and several ideas and techniques used here are based on their work. There is also an overlap with [18] and [25], which try to apply the same underlying idea on the Helios voting system: homomorphic Pedersen commitments are used to tally the votes, but a private channel is used to send auxiliary information to the tallying authority.

The structure of this paper is straightforward: in Section 2 we describe the mixing process, and in Section 3 the private and public verification process using RBV. Section 4 presents formal statements of the properties and proofs, followed by discussion in Section 5.

2 Description of the mixing process

2.1 Cryptographic tools

Pedersen commitments and Paillier encryption Usually Pedersen commitments are used in the multiplicative group modulo a prime number p, but it is not obvious how to find a matching homomorphic encryption scheme whose homorphic order is $p-1$. We borrow therefore the construction proposed by Moran and Noar [22, Appendix A].

They propose to use standard Paillier encryption with as public key an integer $N = p_1 p_2$ and an element $\gamma \in G_1 = \mathbb{Z}^*_{N^2}$. The primes p_1 and p_2 must be safe primes; they constitute the private key. Then the Paillier encryption of a message m using randomness $r \in G_1$ is $E(m) = \gamma^m r^N (\text{mod } N^2)$. We assume that for each encryption different value for r is used. Besides being homomorphic and (therefore) allowing re-encryption, it is well-known that Paillier offers semantical security.

The commitment scheme now takes place in the order N subgroup of $\mathbb{Z}^*_{4N+1}$, were it is required that $4N+1$ is a prime number too. Note that with overwhelming probability a random element of $\mathbb{Z}^*_{4N+1}$ has order $N, 2N$ or $4N$. So finding generators of G_1 is easy: pick α random in $\mathbb{Z}^*_{4N+1}$. Test if $\alpha^N = 1$; if not then test if $(\alpha^N)^2 = 1$; if not then test if $(\alpha^N)^4 = 1$. A second generator β is found in the same way.

The infeasibility of computing $\log_\alpha \beta$ is guaranteed if p_1 is a sufficiently large random safe prime, and α and β are random generators of G_1.

Proofs of knowledge and verification The authorities must publish additional information that will allow outsiders to audit them. This information depends on the method chosen. In this section we present a straightforward cut-and-choose method, which is easier to grasp but rather inefficient. A speed-up for this approach is described in Section 3.

We will first describe how authority A shows it acted honestly. Let $A_{in} = \{\langle u(i), v(i)\rangle\}_{i=1}^{K}$ be the input batch where $u(i) = \alpha^{s_i}\beta^{t_i}$ is a secret t_i "blinded" by a random number s_i using Pedersen commitments and $v(i) = E(s_i)$ the encryption of the commitment values s_i (using Paillier). The output batch is $A_{out} = \{\langle u_A(j), v_A(j)\rangle\}_{j=1}^{K}$ where $u_A(j)$ and $v_A(j)$ are shuffled recodings of $u(i)$ and $v(i)$. A must show that it knows a permutation π and values $s_A(i)$ such that $u_A(\pi(i)) = \alpha^{s_A(i)}u(i)$, $v_A(\pi(i)) = E(s_A(i))v(i)$ with $\pi(i) = j$. This is accomplished by breaking up the permutation π in two permutations π_1 and π_2. A creates an intermediate batch permuted under π_1, commits to π_1^{-1} and π_2, and creates yet another set of blinded versions of $u(i)$ using a different $s'_A(m)$ for each entry. Here we used a different index, m, to emphasize that $m = \pi_1(i)$.

1. Let π_1 be a random permutation of size K and define π_2 such that $\pi_2 \cdot \pi_1 = \pi$. For $m \in I$ choose $s'_A(m) \in G_1^*$ and set $u'_A(m) = \alpha^{s'_A(m)}u(\pi_1^{-1}(m))$; $v'_A(m) = E(s'_A(m))v(\pi_1^{-1}(m))$; $y_A(m) = BC(\pi_1^{-1}(m))$ and $z_A(m) = BC(\pi_2(m))$.
   For $m \in I$, A publishes the following information: $\langle y_A(m); u'_A(m); v'_A(m); z_A(m)\rangle\rangle$
2. The verifier issues a challenge, Left or Right.
3. If Left then for $m \in I$ A opens all the $y_A(m)$ and all $s'_A(m)$ and V verifies whether $u'_A(m) \overset{?}{=} \alpha^{s'_A(m)}u(\pi_1^{-1}(m))$ and $v'_A(m) \overset{?}{=} E(s'_A(m))v(\pi_1^{-1}(m))$.
   If Right then for $m \in I$ A opens all the $z_A(m)$ and all $s''_A(m) := s_A(m) - s'_A(m) (\bmod q)$ and V verifies whether $u_A(\pi_2(m)) \overset{?}{=} \alpha^{s''_A(m)}u'_A(m)$ and $v_A(\pi_2(m)) \overset{?}{=} E(s''_A(m)Ev'_A(m)$.

Note that during the public verification process, only the correct re-encryption of $u(i)$ to $u_A(j)$ is shown. This process is repeated for different permutations, decommitment values, and challenges until the probability of an undetected coercion is below a certain value, defined by the security parameter, λ.

2.2 Parties and cryptographic assumptions

For our protocol we have the following parties:

Users We have K users that will submit messages to the mix-net. We will write the indexes i, j, k, l of the users between parenthesis.

Mixes We have three mixing authorities or mixes, A1, B1, C1; and three mixes A2, B2, C2.

Publication authority We have one publication authority Z, whose task is to calculate and publish the messages submitted.

Key trustee We have one key trustee T, who keeps the private Paillier key and assists Z when decryption is stipulated.

Bulletin board There exists a write-only public bulletin board (BB) on which users and mixes can publish information. This is a fairly standard assumption. See for instance [20] for more details.

Random beacon We assume that all random challenge bits used in the verification steps come from a trusted beacon. This assumption is fairly standard. See, for instance, [22].

Auditors We have an unspecified set of auditors, who oversee the random beacon's generation of random bits, and who verify and certify the election result.

Observers Observers are interested parties, possibly voters, who verify the election result.

We make the following assumptions:

Discrete Log *The authorities cannot break the discrete log problem for the parameters chosen before the messages have been published and certified.*

Private channels between the authorities *There exist private channels between all the authorities.* Using standard cryptographic techniques this is easy to realize.

Private channels from user to A1 and A2 *There exists a private channel between the user and A1 and A2.* This assumption is somewhat harder to justify, since an unconditional channel can only be achieved using a one-time pad. See Section 5.2 for a discussion on this topic.

Destruction of all information *After the protocol has been certified all all the authorities destroy all information private to them.*

In the next sections we describe the protocol. Figure 1 shows the information flow between all parties during the mixing process. We urge the reader to consult Appendix A for a more detailed figure and a high-level description providing a good overview of the protocol.

2.3 Phase I: Message submission

This section describes what a user must do to submit her message t to the system.

1. The user chooses $s, s', s_1, s'_1, t_1 \in G_2$ and computes s_2, s'_2, and t_2 such that $s = s_1 + s_2$, $s' = s'_1 + s'_2$, and $t = t_1 + t_2$. She also generates a 128-bit random number r
2. She computes $u_l = \alpha^{s_l} \beta^{t_l}$, $x_l = \alpha^{s'_l} \beta^r$, $v_l = E(s_l)$, and $y_l = E(s'_l)$.
3. Using a private channel, she sends v_l, y_l to A_l.
4. The values u_1, u_2, x_1 and x_2 and some user identification number are sent to the bulletin board. She can and should verify that they appear, exercising her right to individual verifiability.

The above description applies only if t is small enough such that its value can be determined by using standard discrete log algorithms on β^t. For larger values the user must create an additional message $z_l = E(t_l)$ and send it through the private channel, together with v_l and y_l. It is quite easy to modify the protocol such that the z_l are processed by the mixes, then multiplied and decrypted by Z, resulting in t. To keep the presentation simple we chose to leave this out of the description.

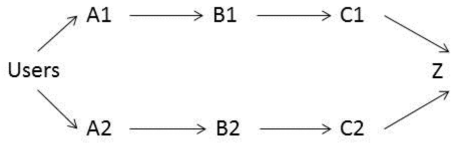
$$\begin{tikzcd}
 & A1 \arrow[r] & B1 \arrow[r] & C1 \arrow[dr] & \\
Users \arrow[ur] \arrow[dr] & & & & Z \\
 & A2 \arrow[r] & B2 \arrow[r] & C2 \arrow[ur] &
\end{tikzcd}$$

Fig. 1. Information flow between all entities during the mixing process

2.4 Phase II: The mixing process

In this section we describe the shuffling procedure for K inputs and one mix-net consisting of three mixes $A1$, $B1$, and $C1$. The shuffling procedure for the three mixes $A2$, $B2$, and $C2$ is identical.

1. (a) The input batch of the first mix $A1$ is $\{\langle u_1(i), x_1(i), v_1(i), y_1(i)\rangle\}_{i=1}^{K}$. For each $i \in [1, K]$, generate random $s_{A1}(i), s'_{A1}(i) \in G_2$. $A1$ will recode $u_1(i)$ and $x_1(i)$ as follows: $u_{A1}(i) = u_1(i)\alpha^{s_{A1}(i)}$ and $x_{A1}(i) = x_1(i)\alpha^{s'_{A1}(i)}$. $A1$ re-encodes the encrypted decommitment values accordingly: $v_{A1}(i) = v_1(i)E(s_{A1}(i))$, $y_{A1}(i) = y_1(i)E(s'_{A1}(i))$ which, due to the homomorphic property, should equal $E(s_1(i) + s_{A1}(i))$ and $E(s'_1(i) + s'_{A1}(i))$.
   (b) For the output batch, $A1$ puts the tuples $\langle u_{A1}(i), x_{A1}(i), v_{A1}(i), y_{A1}(i)\rangle$ in numerical order, thus implicitly defining a random permutation, called π_{A1}, between its input batch $\{\langle u_1(i), x_1(i), v_1(i), y_1(i)\rangle\}_{i=1}^{K}$ and output batch $\{\langle u_{A1}(j), x_{A1}(j), v_{A1}(j), y_{A1}(j)\rangle\}_{j=1}^{K}$, where $j := \pi_A(i)$. The set of public pairs $\{u_{A1}(j), x_{A1}(j)\}_{j=1}^{K}$ is sent to the BB.
   (c) $A1$ proves to the public that the public output batch is a recoding and permutation of the public input batch.
   (d) The set of private pairs $\{\langle v_{A1}(j), y_{A1}(j)\rangle\}$ and the corresponding input $\{\langle v_1(i), y_1(i)\rangle\}$ is sent to the next mix, $B1$.
   (e) $A1$ proves to $B1$ that the output batch (the public as well as the private information) is a re-encryption and permutation of the entire input batch.
2. Mix $B1$ does the same as $A1$:
   (a) Using $\{\langle u_{A1}(j), x_{A1}(j), v_{A1}(j), y_{A1}(j)\rangle\}_{j=1}^{K}$ as its input batch, and the newly generated random numbers $s_{B1}(j), s'_{B1}(j) \in G_2$, it computes $u_{B1}(j) := u_{A1}(j)\alpha^{s_{B1}(j)}$, $x_{B1}(j) := x_{A1}(j)\alpha^{s'_{B1}(j)}$, $v_{B1}(j) := v_{A1}(j)E(s_{B1}(j))$, and $y_{B1}(j) := y_{A1}(j)E(s'_{B1}(j))$.
   (b) The resulting tuples are ordered numerically resulting in the output batch $\{\langle u_{B1}(k), x_{B1}(k), v_{B1}(k), y_{B1}(k)\rangle\}_{k=1}^{K}$, implicitly defining $k := \pi_{B1}(j)$. The set of public pairs $\{u_{B1}(k), x_{B1}(k)\}_{j=1}^{K}$ is sent to the BB.
   (c) Subsequently, $B1$ proves to the public that the public output batch is a recoding and permutation of the public input batch.
   (d) The set of private pairs $\{\langle v_{B1}(k), y_{B1}(k)\rangle\}$ and the corresponding input $\{\langle v_{A1}(j), y_{A1}(j)\rangle\}$ is sent to the last mix, $C1$.
   (e) $B1$ proves to $C1$ that the output batch is a re-encryption and permutation of the input batch.

3. Authority $C1$ acts analogically to $B1$. Its output batch $\{\langle u_{C1}(l), x_{C1}(l), v_{C1}(l), y_{C1}(l)\rangle\}_{l=1}^{K}$ is considered the final result of the shuffling operation. The u_{C1}, x_{C1} are published, the v_{C1}, y_{C1} are sent privately to the Publication Authority Z, together with the proofs for the correct re-encryption and permutation of mix C1.

2.5 Phase III: Decoding and publication of the messages

1. The output batches $\{\langle u_{C1}(l), x_{C1}(l), v_{C1}(l), y_{C1}(l)\rangle\}_{l=1}^{K}$ and $\{\langle u_{C2}(l), x_{C2}(l), v_{C2}(l), y_{C2}(l)\rangle\}_{l=1}^{K}$ from $C1$ and $C2$ respectively constitute Z's input batch.
   With the help of the key trustee, Z decrypts the $y_{C1}(l)$ and $y_{C2}(l)$, reveals $d'_1(l) = s'_1(l) + s'_{A1}(l) + s'_{B1}(l) + s'_{C1}(l)$ and $d'_2(l) = s'_2(l) + s'_{A2}(l) + s'_{B2}(l) + s'_{C2}(l)$ and decodes $x_{C1}(l)$ and $x_{C2}(l)$. These should decrypt to two equal sets of random numbers $\beta^{r(l)}$, showing the correspondence between tuples coming from the same user. Z publishes this correspondence in the form of a permutation σ from $\{\langle u_{C1}(l), x_{C1}(l)\rangle\}_{l=1}^{K}$ to $\{\langle u_{C2}(\sigma(l)), x_{C2}(\sigma(l))\rangle\}_{l=1}^{K}$ on the BB. It also proves that the $\beta^{r(l)}$ used in $x_{C1}(l)$ and $x_{C2}(\sigma(l))$ is the same.
2. Z publishes a proof that $Dec(x_{C1}(l)) = Dec(x_{C2}(\sigma(l)))$ by showing that it has knowledge of the "recoding value" $d'_2(l) - d'_1(l)$ such that $x_{C1}(l) * \alpha^{d'_2(l)-d'_1(l)} = x_{C2}(\sigma(l))$. This can be checked by applying the standard cut-and-choose verification method.
3. Z privately computes $v^*(l) = v_{C1}(l)v_{C2}(\sigma(l))$. With the help of the key trustee, Z privately decrypts: $d(l) = s_1(l) + s_{A1}(l) + s_{B1}(l) + s_{C1}(l) + s_2(l) + s_{A2}(l) + s_{B2}(l) + s_{C2}(l) = D(v^*(l))$ and decodes $t^*(l) = D(u^*(l))$. It publishes $\{d(l), t^*(l)\}_{l=1}^{K}$ on the BB.
4. Z and the public can verify whether $u^*(l) = u_{C1}(l)u_{C2}(\sigma(l)) = \alpha^{d(l)}\beta^{t^*(l)}$ for $l \in [1, K]$.

3 Verification speed-up for large input sets

3.1 Preliminaries

Introduction to Randomized Block Verification Universally verifiable mix-nets introduce new advantages in eVoting schemes like end-to-end verifiability while still providing voter anonymity. However, to motivate their use for an election on state level, early verification methods were not fast enough. The current systems (e.g. manual tallying, DREs, scanning solutions) provide the first results already some hours after the poll station closed while, depending on the size of the input set, the verification methods may take days before the election outcome can be announced. Thus in 2002 Jakobsson et al. proposed Randomized Partial Checking (RPC) [21] which provides sufficient efficiency to be used in practise. The basic idea is that each mix shuffles its input set two times and shows for each element in the intermediate batch either its correct association to one input element or to one output element (e.g. decided by random coin).

However, this approach has two drawbacks. First, the probability of doing an undetected coercion is significantly higher compared to mathematical ZK proofs (the probability of modifying n votes is at most $\frac{1}{2^n}$ because just the half of the associations is checked). Second, the correspondence between input and output elements got restricted. More precisely, each output element, which correct association to the intermediate batch was checked, is originated from an input ciphertext which association to the intermediate batch has not been revealed. This privacy leakage was identified by Chaum and addressed in [7]. Further, from the legal point of view, the probability of detecting a modified vote might not be sufficient for a legally binding election.

In the same year Golle et al. introduced Optimistic Mixing [16]. The basic idea is to use the homomorphic properties of the shuffled elements to prove the correct re-encryption for the whole set of input and output elements. First all input ciphertexts and output ciphertexts are multiplied. Due to the homomorphic property the value encrypted in the results equates the sum of all secrets. The fact that the sum of all encrypted secrets should remain unchanged during a correct shuffling process is used to prove the accurate re-encryption. However, this approach has been shown unsecure regarding anonymity of the voter by Wikström [28] and Abe and Imai [1]. A weakness is that a mix can change ciphertexts and adapt a further one so that the sum of the secret remains unchanged while this modification is not detected before decoding. The proposed roll-back process then reveals sensitive information, breaking voter privacy.

In the same year Boneh and Golle proposed another verification method [5] in which not the product of all elements but of some randomly generated blocks is checked. Though this modification provides a high chance of detecting such an attack, the blocks can overlap and subsets of input elements can be mapped to subsets of output elements.

Allepuz and Castelló further developed this idea. In their proposal, all input elements are distributed to blocks in a way that each element is contained in exactly one block and all blocks have the same size. In addition, the block division for each mix depends on its predecessor to provide better privacy, similar to [7]. However, to provide that each output of the mix-net can be descended from each input, all mixes have to be honest and keep their used permutation secret. This drawback is addressed by Random Block Verification (RBV) [13]. Similar to RPC each mix shuffles its set of input elements two times. This allows generating the block fragmentation in a way that the statistical correspondence between each input element and output element of each mix is almost equal.

High-level description of RBV We use the RBV approach to speed up the slow verification offered by cut-and-choose. The cut-and-choose approach described in Section 2.1 is very inefficient. To keep the probability of doing an undetected coercion low, the described process has to be repeated several times.

In addition to the verification of the proof, for each iteration a new shuffled and recoded output for all inputs has to be computed. This is very inefficient, especially when the mix processed a large set of input values.

Using RBV, we divide the input elements of the mix into different, equally sized, subsets (which in the following we will refer to as blocks) and ask the mix to show the corresponding output subsets (each containing all elements of the associated input block in recoded form). The mix needs knowledge of the used permutation to answers this challenge. In the next step all elements of the input block and all elements of the corresponding output block are multiplied. Due to the additive homomorphic properties of Paillier encryption and Pedersen commitments, the secret of every element of the input block is added and thus the product of the input block encodes their sum. If the mix recoded and shuffled its input correctly, the product of the corresponding output block encodes the same secret in recoded form. Thus, the mix is asked to prove, by cut-and-choose, that it has knowledge of the recoding value. If the mix processed its input correctly, this value can easily be computed by adding all random numbers, used to recode the elements of the input block.

In order to provide user privacy, it should be avoided that the block fragmentation can be used to associate one output element of the mix-net to a subset of mix-net inputs. To prevent overlapping of subsets, each ciphertext is contained in exactly one block. Further, we require each mix to shuffle and recode its input set two times to provide that each input ciphertext can be contained in each output block. More precisely, first it is checked whether the intermediate batch is a correct recoding of the input batch. The block distribution for this verification step is generated by using the random beacon. Following, the input blocks to verify the correct recoding of the intermediate batch are generated such that each input block contains at least one ciphertext of each output block (generated in the last verification step). This is possible, if the size L of the used blocks is chosen such that for K inputs $L^2 \geq K$.

Note that the higher the block size, the lower the probability of doing an undetected coercion. More precisely, to modify the secret of a ciphertext, the mix has to adapt the secret of a further ciphertext, so that the sum remains unchanged, and hope that both ciphertexts end up in the same block. The lowest probability for such an undetected coercion is reached for block size $L = \sqrt{K}$ and is $\frac{\sqrt{K}-1}{K-1}$ (for more information see [13]). This probability decreases exponential as a function of the number of modified outputs.

Notation Using the mix-net verification method RBV, each mix has to shuffle its set of input values $\{\langle u_1(i), x_1(i), v_1(i), y_1(i)\rangle\}_{i=1}^{K}$ two times. The output of the first mix $A1$ is $\{u_{A1}(j), x_{A1}(j), v_{A1}(j), y_{A1}(j)\}_{j=1}^{K}$ after the first and $\{u_{A1'}(j'), x_{A1'}(j'), v_{A1'}(j'), y_{A1'}(j')\}_{j'=1}^{K}$ after the second shuffling step. The output of $A1$ is the input of mix $B1$ which correspondingly outputs $\{u_{B1}(k), x_{B1}(k), v_{B1}(k), y_{B1}(k)\}_{k=1}^{K}$ after the first and $\{u_{B1'}(k'), x_{B1'}(k'), v_{B1'}(k'), y_{B1'}(k')\}_{k'=1}^{K}$ after the second shuffling step. The same notation is used for mix $C1$.

3.2 Verification by RBV

In this section we show the public verification process for K inputs and one mix-net consisting of three mixes $A1$, $B1$, and $C1$ using a different approach. Note that this verification process (in contrast to the cut-and-choose approach described in Section 2.1) takes place after each mix finished its recoding and shuffling process (in Phase II) and before Z matches the output and decodes the secret (in Phase III). Note that the verifier of each mix is still its successor, since it knows the private input and output values v and y, needed for verification.

Public Verification Process

1. After each mix completed its computation, a permutation on the set of input ciphertexts $\{u_1(i), x_1(i)\}_{i=1}^{K}$ is done. This permutation has to be unknown during the shuffling process to prevent that the block fragmentation is predictable and will be created using the random beacon.
2. The set of permuted input ciphertexts $\{u_1(\pi(i)), x_1(\pi(i))\}_{i=1}^{K}$ is divided into $L := \lfloor\sqrt{K}\rfloor$ blocks. Thus we have $R := K - L^2$ blocks of $L + 1$ elements and $L - R$ blocks of L elements.
3. For each $l \in [1, L]$ the first mix (A1) has to show a block $n_{A1}(l)$ composed of output ciphertexts which contains the ciphertexts of the input block $m_{A1}(l)$ in recoded form. As a consequence, for each $l \in [1, L]$ the block product u^l_{A1} is a recoding of the block product u^l_1 with recoding value s^l_{A1} . The same holds true for every block x^l_{A1} and recoding value s'^l_{A1}. The association between input and output blocks is published to provide universal verifiability.
4. Now mix A1 has to prove for each $l \in [1, L]$ that
$u^l_{A1} = \alpha^{s^l_{A1}} * u^l_1$. and that it has knowledge of s^l_{A1}. This is done by using the "Zero-Knowledge Proof That Two Commitments Are Equivalent" introduced in [22, Appendix B].
5. After each block has been verified, the output of the second shuffling step $\{u_{A1'}(j'), x_{A1'}(j')\}_{j'=1}^{K}$ needs to be checked. In the verification process of the second shuffling step the division of the input ciphertexts always depends on the blocks build in the previous verification process. More precisely, the set of input values $\{u_{A1'}(j'), x_{A1'}(j')\}_{j'=1}^{K}$ is divided into L blocks in a way that each block $m_{A1'}(l)$ of $\{u_{A1'}(j'), x_{A1'}(j')\}_{j'=1}^{K}$ contains at least one ciphertext of each block $n_{A1}(l)$ (output block determined in the verification of the previous shuffling step).
6. Step 1 to 5 are repeated until the shuffling of each authority has been checked.

Private Verification by RBV

1. The verifier (in this case $B1$) divides the received encrypted decommitment values $\{v_1(i), y_1(i)\}_{i=1}^{K}$ and $\{v_{A1}(j), y_{A1}(j)\}_{j=1}^{K}$ correspondingly to the block division published by $A1$. Thus, for each $l \in [1, L]$ the block v^l_{A1} is a recoding of block v^l_1 with recoding value s^l_{A1}. The same holds true for every block y^l_{A1} y^l_1 and recoding value s'^l_{A1}.

2. Thus, mix $A1$ has to show that for each $l \in [1, L]$ it used the same decommitment value s^l_{A1} to recode u^l_1 and v^l_1. More precisely, $A1$ shows $u^l_{A1} = \alpha^{s^l_{A1}} * u^l_1$, $v^l_{A1} = E(s^l_{A1}) * v^l_1$ and that it has knowledge of s^l_{A1}. This can be done with the "Zero-Knowledge Proof That Two Commitments Are Equivalent" described in [22, Appendix B]. The same is shown for the decommitment values s'^l_{A1} and blocks x^l_1 and y^l_1.
3. Step 1 and 2 are repeated until the shuffling of each mix has been checked.

4 Properties claimed and proofs

4.1 Correctness, Individual and Universal Verifiability

With respect to the correctness of the protocol, we claim the following property.

Claim. Let $T_{in} = \{t(1) \ldots t(K)\}$ be the set of messages as submitted by the users, and let $T_{out} = \{t^*(1) \ldots t^*(K)\}$ be the set of message published by Z. Then, with overwhelming probability, $T_{in} = T_{out}$ even if all authorities conspire to act maliciously.

Proof sketch: First, observe that for correctness, all auxiliary information sent over the private channels is irrelevant. It only serves to help Z to open the u's. Second, even a dishonest mix cannot change the challenge bits used in verification steps of the protocol since they come from a trusted beacon. Third, in step III.1, Z must prove that the $r(l)$ used in $x_{C1}(l)$ and $x_{C2}(\sigma(l))$ is the same, making cheating during the matching procedure on behalf of Z impossible, unless breaking the discrete log is possible.

Now we can follow the familiar argument of re-encryption mixes together with the binding property of the commitment scheme. Suppose that all authorities conspire. The information that is published as a response to the challenges proves (with overwhelming probability) that, during the various mixing phases, each input has been multiplied by some factor $\alpha^{s.}$ and has been permuted. This, together with the matching procedure, implies that each value $u^*(l)$ published by Z must be of the form $\alpha^{d(l)}\beta^{t(l)}$, where $d(l) = s_1(l) + s_{A1}(l) + s_{B1}(l) + s_{C1}(l) + s_2(l) + s_{A2}(l) + s_{B2}(l) + s_{C2}(l)$. If the authorities open $u^*(l)$ for a different value $t' \neq t(l)$, this would imply that $\alpha^{d'}\beta^{t'} = \alpha^{d(l)}\beta^{t(l)}$, meaning that they could compute the discrete log of α with respect to base β, contradicting our first assumption. Therefore $T_{in} = T_{out}$, completing the proof.

Note that, because of the audit information published during all the mixing steps, any observer can perform the checks. This, together with the guaranteed randomness of the challenge bit, means that the protocol is universally verifiable. And individual verifiability follows, simply because of the fact that a user can check that u_1 and u_2 appear on the BB.

Observe that the proof of the user in Phase I is not public. This means that if a user and $A1$ or $A2$ conspire, then wrong values of d may be sent to Z and she will not be able to decrypt/open the u belonging to that user. This does not affect messages submitted by other users.

4.2 Privacy

Claim. For each message published, the public view, consisting of all the input and output batches of the respective mixes together with the proofs and the information published by Z, reveals no (Shannon) information about the identity of its originator. This remains true even if one authority is dishonest.

Proof sketch: If all mixes were honest, then secret-splitting would not be necessary, and privacy follows from the fact that the output batch T_{out} is an unknown permutation of the input batch, T_{in}. No information is revealed, since the commitments used to encode the messages are unconditional.

Note however the user submits to $A1$ and that $B1$ acts as a verifier for $A1$, so both to see v and y. So without splitting s, a dishonest $A1$ or $B1$ can find out t when the homomorphic encryption algorithm E has been broken. To eliminate this possibility, the user has to split s and t and submit the shares to two different authorities, $A1$ and $A2$. Obviously, the output batch of $B1$ and $B2$ cannot be the mix-net's final output batch. Therefore two more mixes, $C1$ and $C2$, are used to shuffle the messages once more, thus guaranteeing privacy even if one of the mixing authorities is dishonest.

Note that the speed-up described in Section 3 should only be used for a large set of input ciphertexts. An attacker can determine the sum of the commitment values, used to decode each block. Thus, the minimum size of one block (the square root of K) should equal the size of the smallest set of inputs which can be (securely) processed by a mix-net.

4.3 Robustness

Claim. The output of the mix-net can be decoded as long as the output set of each mix has been verified.

It is clear from the construction that if everybody is honest, then the process must succeed, always. Now let us first assume that all the user's submissions are constructed correctly. Each mix processes its input set and proves the correctness of the output to the successor. More precisely, each mix proves that it has knowledge of the used permutation, the values used for recoding and re-encrypting the input set, and that the same recoding values has been used to recode the Pedersen Commitments on the public and the Paillier encryptions of the private channel. If cheating is detected, the mix can simply be replaced. However, in case the first mix is malicious, the users need to re-send their data. For applications where this is undesirable, re-sending of data can be prevented by redundancy.

Further, to avoid that a mix can lie about its input towards the verifier, we ask the user and each mix to sign its output. Thus, a verifier can check the authenticity of the input batch. Note that the digital signature scheme used does not need to be information-theoretically secure.

A problem is that a user can submit inconsistent inputs. More specifically, it seems impossible to verify that the public shares t_1, t_2 encoded, and the private shares t_1, t_2 encrypted, are consistent while maintaining unconditional privacy towards the first two mixes, A1 and A2. If a user errs, it just means her message cannot be decoded by Z; it does not affect any other message. When it comes to a dispute, the first two mixes can challenge the user to supply a proof almost identical to the proofs used by the mixes that her public and private inputs are consistent. Such a proof would reduce the user's privacy to be only computational, but the user will not be able to supply such a proof anyway.

5 Discussion

5.1 The extra work of the adversary

Note that if an adversary wants to attack the new protocol she needs to do a lot more work than before. In current protocols for mixing an adversary can sit and wait for years or decades until the homomorphic scheme is broken. She copies the information from the bulletin board locally, decrypts, and find out who sent which message.

In the new scheme, if all authorities are honest, an adversary's only course of action is either to eavesdrop on the communication between each individual and the first authority, or intercept the communication between the authorities, and then break the underlying cryptographic assumption.

The secret splitting is only justified when users distrust authorities. Observe that in many instances, including elections, most users are not so worried about the mix authorities being dishonest. But they could, rightfully, be worried about their authorship of their (supposedly anonymous) message being exposed twenty years later. Before this paper, even willing mixing authorities had no way of offering a better service in this respect.

5.2 On the private channel assumption of the user

In order to guarantee its own privacy, a user should, ideally, use a private channel to each mix authority to which she submits input shares. Implementing a private channel using a one-time pad implies that the user must exchange a private key through a reliable channel. This can be arduous or not, depending on the situation. If the user does not want to go through all that trouble, it could give up on unconditional privacy and use some public key cryptography scheme to send her private key. The difference is that now the user can choose the scheme (it could be post-quantum, for instance), as opposed to the old situation in which everybody´s privacy depended on one scheme resisting attacks.

5.3 Alternative implementations and generalizations

We chose Pedersen commitments because of its homomorphic properties in both the decommitment value and the commitment value. This propery is shared by few commitment schemes. We believe that an alternative implementation with unconditional Jacobi symbols commitments and Rabin encryption might work, but the scheme would be very inefficient since each Jacobi symbol would implement only one bit.

More interesting is the recent suggestion of Olivier Pereira (private communication) for an efficient unconditional bit commitment scheme with matching homomorphic encryption scheme based on elliptic curves. If this could be used in combination with the non-interactive proof (or rather, argument) techniques proposed by Groth[19], this might lead to a very efficient verification procedure.

An obvious generalization of the scheme is to use more generators to encode various ts: $u = \alpha^s \beta_L^{t_1} \ldots \beta_L^{t_L}$.

Conclusion

We presented a novel protocol for a publicly-verifiable mix-net which offers everlasting privacy towards observers, meaning that the information made public on the bulletin board does not help an adversary with unlimited computational resources. We believe this is an important step forward since, as Chaum himself already argued in 1984 [9], individuals cannot expect to understand the difference between computational and unconditional security, and they should not have to worry about it.

In particular, computational security is simply not enough in elections were there is a lot at stake, invalidating many proposed schemes based on homomorphisms and/or mix-nets. Several voting scheme with everlasting privacy have been proposed [22, 17] but they only work with a very specific ballot layout. The advantage of using a mix-net in an election is that it does not impose any restrictions on the ballot layout, since the message to be mixed can be any sequence of bits.

The main purpose of this paper is to show that mixing with unconditional privacy could be done in principle, maybe relaxing somewhat on the matter of formal proofs on the one hand, and questions of optimizations and performance on the other. We plan to work on these matters in the future, and encourage other researchers to do the same.

Acknowledgement

This work is a direct result of the second author's visit to Dagstuhl and Darmstadt in July 2011, and he would like to express his gratitude to the Dagstuhl Institute, to the Interdisciplinary Centre for Security, Reliability and Trust at the University of Luxemburg, to Centre for Advanced Security Research at the University of Darmstadt, and to his hosts, Peter Ryan and Johannes Buchmann, for making this visit possible. He also would like to thank Tal Moran and Olivier Pereira for interesting discussions. His research was partially supported by a FAPEMIG grant, number APQ-02719-10. Further this paper has been developed within the project 'VerKonWa' — Verfassungskonforme Umsetzung von elektronischen Wahlen — which is funded by the Deutsche Forschungsgemeinschaft (DFG, German Science Foundation).

References

1. Abe, M., Imai, H.: Flaws in some robust optimistic mix-nets. In: ACISP. pp. 39–50 (2003)
2. Adida, B.: Helios: Web-based open-audit voting. In: USENIX Security Symposium. pp. 335–348 (2008)
3. Allepuz, J.P., Castelló, S.G.: Universally verifiable efficient re-encryption mixnet. In: Electronic Voting. pp. 241–254 (2010)
4. Bayer, S., Groth, J.: Efficient zero-knowledge argument for correctness of a shuffle. In: To appear at Advances in Cryptology - EUROCRYPT 2012 (2012)
5. Boneh, D., Golle, P.: Almost entirely correct mixing with applications to voting. In: ACM Conference on Computer and Communications Security. pp. 68–77 (2002)
6. Chase, M., Kohlweiss, M., Lysyanskaya, A., Meiklejohn, S.: Malleable proof systems and applications. In: To appear at EUROCRYPT 2012 (2012)
7. Chaum, D.: Secret-ballot receipts and transparent integrity. better and less-costly electronic voting and polling places (May 2002)
8. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM 24(2), 84–88 (1981)
9. Chaum, D.: A new paradigm for individuals in the information age. In: IEEE Symposium on Security and Privacy. pp. 99–106 (1984)
10. Chaum, D., Carback, R., Clark, J., Essex, A., Popoveniuc, S., Rivest, R.L., Ryan, P.Y.A., Shen, E., Sherman, A.T., Vora, P.L.: Scantegrity ii: end-to-end verifiability by voters of optical scan elections through confirmation codes. IEEE Transactions on Information Forensics and Security 4(4), 611–627 (2009)
11. Chaum, D., Damgård, I., van de Graaf, J.: Multiparty computations ensuring privacy of each party's input and correctness of the result. In: CRYPTO. pp. 87–119 (1987)
12. Clarkson, M.R., Chong, S., Myers, A.C.: Civitas: Toward a secure voting system. In: IEEE Symposium on Security and Privacy. pp. 354–368 (2008)
13. Demirel, D., Jonker, H., Volkamer, M.: Random block verification: Improving the norwegian electoral mix net. In: To appear at EVote 2012 (2012)
14. Furukawa, J.: Efficient and verifiable shuffling and shuffle-decryption. IEICE Transactions 88-A(1), 172–188 (2005)
15. Gjøsteen, K.: Analysis of an internet voting protocol. Cryptology ePrint Archive, Report 2010/380 (2010), `http://eprint.iacr.org/`
16. Golle, P., Zhong, S., Boneh, D., Jakobsson, M., Juels, A.: Optimistic mixing for exit-polls. In: ASIACRYPT. pp. 451–465 (2002)
17. van de Graaf, J.: Voting with unconditional privacy by merging prêt à voter and punchscan. IEEE Transactions on Information Forensics and Security 4(4), 674–684 (2009)
18. van de Graaf, J., Samarone, R., Demirel, D.: Improving helios with unconditional privacy. in preparation (2012)
19. Groth, J.: Short pairing-based non-interactive zero-knowledge arguments. In: Abe, M. (ed.) ASIACRYPT. Lecture Notes in Computer Science, vol. 6477, pp. 321–340. Springer (2010)
20. Heather, J., Lundin, D.: The append-only web bulletin board. In: Formal Aspects in Security and Trust. pp. 242–256 (2008)
21. Jakobsson, M., Juels, A., Rivest, R.L.: Making mix nets robust for electronic voting by randomized partial checking. In: USENIX Security Symposium. pp. 339–353 (2002)

22. Moran, T., Naor, M.: Split-ballot voting: Everlasting privacy with distributed trust. ACM Trans. Inf. Syst. Secur. 13(2) (2010)
23. Neff, C.A.: A verifiable secret shuffle and its application to e-voting. In: ACM Conference on Computer and Communications Security. pp. 116–125 (2001)
24. Park, C., Itoh, K., Kurosawa, K.: Efficient anonymous channel and all/nothing election scheme. In: EUROCRYPT. pp. 248–259 (1993)
25. Pereira, O.: Election verifiability or vote privacy: Do we need to choose? in preparation (2012)
26. Ryan, P.Y.A., Bismark, D., Heather, J., Schneider, S., Xia, Z.: Prêt à voter: a voter-verifiable voting system. IEEE Transactions on Information Forensics and Security 4(4), 662–673 (2009)
27. Sako, K., Kilian, J.: Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth. In: EUROCRYPT. pp. 393–403 (1995)
28. Wikström, D.: Five practical attacks for "optimistic mixing for exit-polls". In: Selected Areas in Cryptography. pp. 160–175 (2003)
29. Wikström, D.: A commitment-consistent proof of a shuffle. In: ACISP. pp. 407–421 (2009)

A Mixing process overview

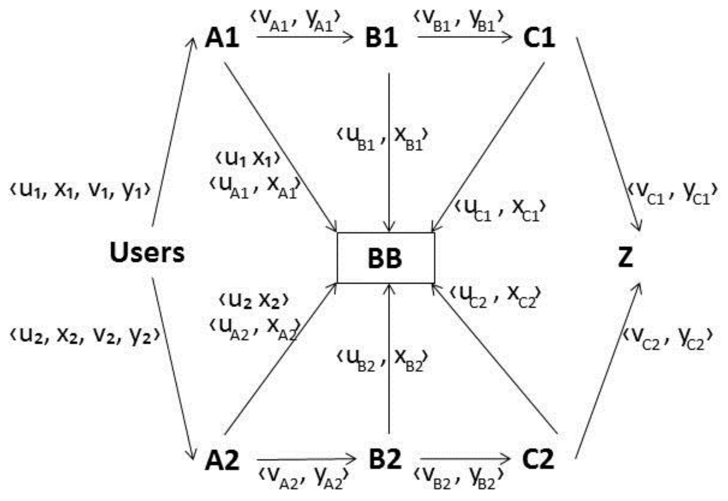

Fig. 2. Detailed information flow between all entities during the mixing process

private channel	public channel	public channel	private channel
User privately splits $s = s_1 + s_2$, $s' = s'_1 + s'_2$ and $t = t_1 + t_2$			
$v_1 = E(s_1)$ $y_1 = E(s'_1)$	$u_1 = \alpha^{s_1}\beta^{t_1}$ $x_1 = \alpha^{s'_1}\beta^r$	$u_2 = \alpha^{s_2}\beta^{t_2}$ $x_2 = \alpha^{s'_2}\beta^r$	$v_2 = E(s_2)$ $v_2 = E(s'_2)$
send to A1	send to BB	send to BB	send to A2
A1's input batch		A2's input batch	
v_1, y_1	u_1, x_1	u_2, x_2	v_2, y_2
$v_{A1} = v_1 E(s_{A1})$ $y_{A1} = y_1 E(s'_{A1})$	$u_{A1} = u_1 \alpha^{s_{A1}}$ $x_{A1} = x_1 \alpha^{s'_{A1}}$	$u_{A2} = u_2 \alpha^{s_{A2}}$ $x_{A2} = x_2 \alpha^{s'_{A2}}$	$v_{A2} = v_2 E(s_{A2})$ $y_{A2} = y_2 E(s'_{A2})$
send to B1	send to BB	send to BB	send to B2
A1's output batch		A2's output batch	
v_{A1}, y_{A1}	u_{A1}, x_{A1}	u_{A2}, x_{A2}	v_{A2}, y_{A2}
B1's input batch		B2's input batch	
$v_{B1} = v_{A1} E(s_{B1})$ $y_{B1} = y_{A1} E(s'_{B1})$	$u_{B1} = u_{A1} \alpha^{s_{B1}}$ $x_{B1} = x_{A1} \alpha^{s'_{B1}}$	$u_{B2} = u_{A2} \alpha^{s_{B2}}$ $x_{B2} = x_{A2} \alpha^{s'_{B2}}$	$v_{B2} = v_{A2} E(s_{B2})$ $y_{B2} = y_{A2} E(s'_{B2})$
B1's output batch		B2's output batch	
send to C1	send to BB	send to BB	send to C2
v_{B1}, y_{B1}	u_{B1}, x_{B1}	u_{B2}, x_{B2}	v_{B2}, y_{B2}
C1's input batch		C2's input batch	
$v_{C1} = v_{B1} E(s_{C1})$ $y_{C1} = y_{B1} E(s'_{C1})$	$u_{C1} = u_{B1} \alpha^{s_{C1}}$ $x_{C1} = x_{B1} \alpha^{s'_{C1}}$	$u_{C2} = u_{B2} \alpha^{s_{C2}}$ $x_{C2} = x_{B2} \alpha^{s'_{C2}}$	$v_{C2} = v_{B2} E(s_{C2})$ $y_{C2} = y_{B2} E(s'_{C2})$
C1's output batch		C2's output batch	
send to Z	send to BB	send to BB	send to Z
v_{C1}, y_{C1}	u_{C1}, x_{C1}	$u_{C2}, x_{C2},$	v_{C2}, y_{C2}
Z's input batch			
Z privately decrypts $y_{C1} \to d'_1 = s'_1 + s'_{A1} + s'_{B1} + s'_{C1}$		Z privately decrypts $y_{C2} \to d'_2 = s'_2 + s'_{A2} + s'_{B2} + s'_{C2}$	
Z privately decodes $x_{C1} \to \beta^r$		Z privately decodes $x_{C2} \to \beta^r$	
Z privately matches the random identities β^r			
Z publishes the correspondence as permutation σ			
Z proves correct matching and knowledge of $d'_2 - d'_1$			
Z (privately) computes $u^* = u_{C1}\sigma(u_{C2})$			
Z privately computes $v^* = v_{C1}\sigma(v_{C2})$			
Z privately decrypts $v^* \to d = s_1 + s_{A1} + s_{B1} + s_{C1} + s_2 + s_{A2} + s_{B2} + s_{C2}$			
Z privately determines t^* from $u^*\alpha^{-d} = \beta^t$			
Z publishes $u_{C1}, \sigma(u_{C2}), d, t^*$			
Z and the public verify that $u^* = u_{C1}\sigma(u_{C2}) = \alpha^d \beta^{t^*}$			

Table 1. Summary of the protocol, with the information flow top-down (instead of left-right, as in the figures). For clearness, the parentheses and indexes of the users have been suppressed, leading to a slght abuse of notation with respect to the matching permutation σ.