Efficient Mix-Net Verification by Proofs of Random Blocks

Denise Demirel[1,2], Melanie Volkamer[1,2], Hugo Jonker[3]

[1] CASED, Darmstadt
[2] TU Darmstadt
[3] University of Luxembourg

Abstract. In order for a mix-net to be usable in practice (e.g. in electronic voting), efficient verification is a must. Despite many advances in the last decade, zero-knowledge proofs remain too computationally intense. Two alternative proof approaches have been suggested: optimistic mix-net verification and randomized partial checking. Puiggalí et al. proposed a verification method combining these two approaches. This paper investigates their mix-net and proposes a verification method which offers both improved efficiency and more privacy.

1 Introduction

In order to offer publicly verifiable vote counting in eVoting the encrypted votes have to be counted in public without revealing any information about the vote decision of the several voters. One approach is to use the homomorphic property of the underlying cryptosystem to tally the votes in encrypted form and decrypt the election results afterwards. So far this approach is rarely used in praxis because the voting systems are mostly to complex regarding dimension and legal regulation. Another approach is to decrypt and count the vote. In order to ensure secrecy of the votes first the link between voters and their vote has to be broken. In this context mix-nets are widely-used to make encrypted votes anonymous.

When Chaum introduced mix-nets [7], the focus was predominantly on anonymous communication. As such, correctness of the process needed to be proven only to the sender (achieved by signature). Later mix-nets improved upon this by providing universal verifiability. Some approaches assume that a part of the senders are honest (e.g. [10, 24]) other use zero-knowledge proofs of correctness, such as cut-and-choose proofs (e.g. [25, 22, 2]). Various efforts were made to improve the efficiency of generating a proof. This led to proofs based on pairwise permutation showing that an output is a re-encryption of one of two inputs (e.g. Millimix [18], MIP-2 [3, 1]), proofs using a matrix representation of the permutation (such as [14, 13, 17, 28, 27]), and proofs using the invariance of polynomials under permutation of of the roots (e.g. [21, 20, 16]).

For some applications, such as voting, efficiency is a prime concern. In many voting schemes, mix-nets are often used to ensure anonymity. However, to be usable in practice, a mix-net should be able to mix all votes and prove correctness within a few hours after the polling stations closed.

Two types of "proofs" of correctness emerged, which offer hight efficiency: Optimistic Mixing (e.g. [6, 15]) and Randomized Partial Checking [19]. The main idea behind Optimistic Mixing is to use batch verification: by proving that the product of the input elements is equal to the product of the output elements. This requires only a handful of efficiently computed multiplications, but does not detect all forms of cheating (intuitively, since $1 \cdot 1 = \frac{1}{2} \cdot 2$). The core concept of Randomized Partial Checking is to have every *mix* operate twice in a row, and then, for every item, reveal either the first or the second mixing. This only requires to reveal one random variable per mixed element, but each *mix* has a 50% chance per input element to cheat undetected.

In effect, both approaches trade off privacy for efficiency: there is a small possibility of the "proof" being correct while the *mix* cheated. Note that this possibility can be reduced by tweaking parameters – which carries a small increase in computational cost. This concept of trading off privacy for efficiency is interesting and deserves further exploration. Note that each approach has its own peculiar trade off between efficiency and privacy. This naturally leads to the question whether the two methods cannot be combined to increase privacy while remaining efficient. Puiggalí et al. [4] propose a mix-net consisting of precisely such a blend of these two approaches.

Contribution. The contribution of this paper is twofold. Firstly, this paper investigates the scheme by Puiggalí et al. and identifies several areas for improvement (including a privacy weakness). These improvements result in a scheme which is more efficient, more secure, and more precisely detailed. Secondly, we provide a mathematical analysis of correctness, privacy and efficiency of the new scheme, and compare these properties to properties of other mix-nets that offer a trade off between privacy and efficiency.

Structure of the paper. The rest of this paper is structured as follows: we first discuss some ElGamal mix-nets (Section 2) and related work (Section 3). As this work extends the contributions of Puiggalí et al, their research is discussed in more detail (Sections 4). Possible improvements to the verification process are discussed in Section 5, all of which are implemented by the new verification process detailed in Section 6. Correctness, privacy, and efficiency of the newly proposed verification process are determined in Section 7 and compared to other mix-nets that trade off privacy for efficiency in Section 8. This is followed by conclusions and future work in Section 9.

2 Re-encryption mix-nets with exponential elGamal

In this section we briefly describe the underlying cryptographic system (elGamal) and the mixing processes in the context of an electronic voting scheme. For legibility we explain the approach in this paper under the use of exponential elGamal. It should be noted that Pallier [23] with a the threshold version like explained in [9] and a zero knowledge proof like proposed in [5]can be used as well.

We assume that votes are encrypted using exponential elGamal and stored on a Web Bulletin Board(BB) where some connection between each encrypted vote and the corresponding voter exist. ElGamal is a randomized public key encryption scheme with homomorphic properties introduced in [12]. Consider two large primes p and q, where q divides $(p-1)$, Z_q is a q-order subgroup of $\mathbb{Z}_p^*$ and g is a generator of G_q. The secret key $x \in Z_q$ is generated and the corresponding public key is (g, y) with $y = g^x$. A plain text s (or here a vote) is encrypted in the following way: $Enc_y(s, r_1) = (g^{r_1}, g^s y^{r_1}) = (\alpha, \beta)$ with random value $r_1 \in_R Z_q$.

Obviously, the encrypted votes cannot just be decrypted because this would violate the secrecy of the voter. One possibility to avoid this is to first anonymize the encrypted votes using a re-encryption mix-net. The output of this mix-net is a set of anonymized re-encrypted votes that can then be decrypted and counted. The re-encryption mix-net with i *mixes* works as follows: The first *mix* loads all encrypted votes (while removing any possible link to the voter - like signatures) published on the BB as input. Every input ciphertext is re-encrypted by exponentiating (α, β) with a generated random value $r_2 \in_R Z_q$: $ReEnc_y = ((\alpha, \beta), r_2) = (\alpha g^{r_2}, \beta y^{r_2}) = (g^{r_1} g^{r_2}, g^s y^{r_1} y^{r_2}) = (g^{r_1+r_2}, g^s y^{r_1+r_2}) = (\alpha', \beta')$. Afterwards, the re-encrypted ciphertexts are shuffled with a random permutation π and the resulting output ciphertexts are published on the BB. Afterwards, the second *mix* loads the output ciphertexts from the first one published on the BB and re-encrypts and shuffles them, as well. This process is repeated until the last one publishes its output ciphertexts on the BB. These are the ciphertexts which are decrypted and counted. Privacy is ensured if at least one *mix* is honest and keeps the permutation secret. In order to also ensure that *mixes* cannot cheat by replacing encrypted votes with new ones, verifiability needs to be implemented, ideally without decreasing the level of privacy.

3 Related Work

3.1 Randomized Partial Checking

A mix-net can be verified by Randomized Partial Checking (RPC, introduced by [19]). Using this method each *mix* in the network perform two consecutive shuffles. During the verification, for each element in the intermediate set (the output of the first *mix*), a coin is flipped to choose whether to reveal either its origin or its destination. For an ElGamal re-encryption mix-net this can be done by publishing the randomness used. Since for the *mixes* the randomness is available already, constructing this proof can be done in constant time. Verification of the proof in this case requires the verifier to perform a re-encryption for every element. The coinflips must occur after mixing to prevent that the *mix* knows in advance which links will be revealed and modify elements accordingly. Using RPC every *mix* has a 50% chance per element of doing and undetected fraud and correspondingly a 2^{-n} chance of changing n items.

3.2 Optimistic Mixing

The optimistic mixing (OM) proposed in [15] by Golle et al. is the first approach which uses an "optimistic" or "fast-track" approach for mix-nets. The input of the mix-net is a double encrypted vote and a hash value over the single encrypted vote: $(Enc_{pk}(\alpha, r_1), Enc_{pk}(\beta, r_2), Enc_{pk}(h(\alpha, \beta), r_3))$, with random numbers r_1, r_2, r_3 and hash value $h\ \{0,1\}^* \rightarrow Z_q$. To verify that all *mixes* of the mix-net generated their output properly, they are asked to prove that the product of the plaintexts of their input ciphertexts equals the product of the plaintexts of their output ciphertexts, meaning $\prod_{i=1}^{n} \alpha_i = \prod_{i=1}^{n} \alpha'_i$ and $\prod_{i=1}^{n} \beta_i = \prod_{i=1}^{n} \beta'_i$ and $\prod_{i=1}^{n} h_i(\alpha, \beta) = \prod_{i=1}^{n} h'_i(\alpha', \beta)$.
The hash value is used to prevent that pairs of ciphertexts are modified in a way which remains the overall product of the plaintexts unchanged. Further the double encryption allows "back-up" mixing, meaning that if cheating is detected the outer-layer encryption can be revealed and the resulting inner-layer ciphertexts can be input of a slower and more fine-grained mixing like e.g. the one proposed by Groth [16].

The proposed optimistic mixing approach by Boneh et al.(PoS),[6]) is slightly faster than the approach proposed by Golle et al., because no cryptographic checksum and no double encryption is used. A drawback of this approach is that the verification only guarantees almost entirely correct mixing. Boneh et al. recommend the use of a slower verification protocol in parallel to guarantee correctness. For the verification process, a security parameter $\gamma \leq 5$ is defined which declares the number of blocks to be checked. A higher value for γ results in a stronger guarantee of correct mixing but also offers less privacy. Every block S_i should have a size of around $\frac{n}{2}$ for a total number of n ciphertexts and is created in a way that every index $1 \leq k \leq n$ is included in S_i independently at random with probability $\frac{1}{2}$. For this a hash function and a non-malleable commitment [11], made by all *mixes* in advance, is used to decide whether a specific input ciphertext c_k is part of a subset S_i. For every block the *mix* is asked to prove, by using the Chaum-Pedersen protocol [8], that the product of the plaintexts for this subset of input ciphertexts equals to the product of the plaintexts of the corresponding subset of output ciphertexts.

4 Mix-net Verification by Puiggalí et al.

In [4], Puiggalí et al. propose an approach to verify a re-encryption mix-net (which uses elGamal as underlaying cryptosystem) which combines the idea of optimistic mixing 3.2 and RPC 3.1. It is executed after the last *mix* has published its output on the Bulletin Board.

This approach encompasses the following steps:

1. An independent verifier provides a random permutation (the challenge).
2. All input votes of the first *mix* are permuted accordingly to the challenge.
3. The list of input votes (of the first *mix*) is divided into $l = \sqrt[i]{n}$ equally-sized blocks, where i is the number of *mixes* and n the number of input

ciphertexts (i.e., votes). Since l is well defined, this can be executed by either the independent verifier, the BB, or the *mix*.

4. For every input block, the first *mix* identifies the corresponding output block. Moreover, for every block, the *mix* publishes the product of the ciphertexts in that block and the Chaum-Pedersen protocol [8] or Schorr's signature scheme [26] can be used to show that the ciphertext product of the input block is equal to that of the corresponding output block.
5. The verifier checks the proofs of the first *mix*.
6. This process continues for each *mix*, where the assignment of *mixes* to blocks depends on the previous *mix*'s assignment – thus ensuring an equal distribution of input ciphertexts over all blocks.

Regarding privacy, Puiggalí et al. state that every output block of the last *mix* is composed of at least one ciphertext of every input block of the first *mix*. Regarding correctness, the authors determine that the probability of detecting two modified votes is $p = 1 - \frac{l-1}{n-1}$ for block size l and a total number of ciphertexts n. This relies on th fact that a manipulation will remain undetected if a malicious *mix* changes two votes without changing the product of the two $(1 \cdot 1 = \frac{1}{2} \cdot 2)$, and these two votes are assigned to the same block.

5 Discussion

The general idea of Puiggalí et al. is rather interesting and promising. However, there are some pitfalls of this approach which are discussed in this section. Corresponding improvements are mentioned in this section while the details are provided in Section 6.

Drawbacks and corresponding improvements. In [4], the estimated cost of performance for every *mix* is (besides the re-encryption) $2b$ for generating the proof and $4b$ for verification, where b is the number of used blocks. A more efficient zero knowledge proof is the one proposed by Jakobsson and Juels in [18]. With this improvement, the proof generation and verification needs just half the time compared to the proposal in [4].

During the mixing process every *mix* of the mix-net re-encrypts and shuffles the input ciphertexts. The origin idea of Puiggalí et al. was to proceed the encrypted votes by one *mix* after the other. It is possible to speed up this process by parallelizing in the following way: the set of input ciphertexts is separated into i subsets (while i is the number of *mixes*). Then all *mixes* get one of the subset to start with and afterwards forward it to the next one (while the next one is different for all of them). This improvement increases the efficiency by factor i[4].

The optimal privacy as it is called in [4] is only ensured if all *mixes* are honest. However, this is not the idea of a mix-net where privacy should be ensured if

[4] This improvement was implemented for the Norwegian Internet voting trials.

one single *mix* is honest. Therefore, we propose to build single *mixes* similar to RPC where each *mix* shuffles twice.

The approach by Puiggalí et al. assumes that the total number of ciphertexts can be grouped in equally sized blocks with block size $l = \sqrt[i]{n}$, for i *mixes* and a n votes. However, in general there will be a remainder when calculating the root of the number of cast votes. In our approach, we address this remainder as well[5].

In [4], the correctness depends on the assumption that the verifier and the first *mix* do not maliciously collaborate. Therefore, it is essential for the correctness that the challenge is really random and generated after the mixing process. The situation can either be improved by a proper random number generator or several independent verifiers.

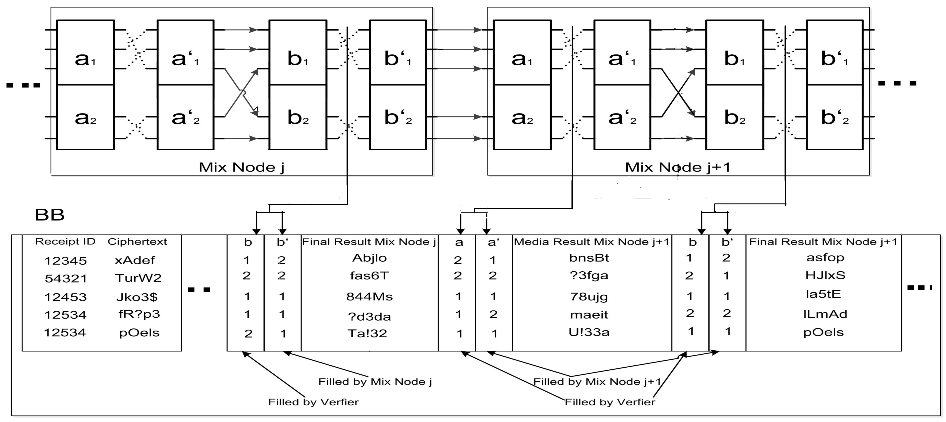
More grounded. [4] gives a formula for calculating the probability of detecting two modified votes in one block. But there is neither a derivation nor a proof for that formula. Furthermore, the paper gives a formula for calculating the minimal block size as a function of the number of used *mixes*. However it is not explained how the ciphertexts should be distributed on the blocks nor proven why the calculated size ensures privacy. We will give an algorithm for our improved version, determine formulas for correctness, privacy and efficiency, and also prove them.

6 Verification Using Integrity Proofs of Random Blocks

In this section we describe a detailed verification process, based on the proposal of Puiggalí et al., which includes all improvements proposed in Section 5. We introduce some notation and the overall mixing process first. Next, we describe verification of proper re-encryption and shuffling for each *mix*. To this end, we first describe which steps have to be done before the verification starts (the setup phase), and then explain how each *mix's* ciphertexts are grouped. Finally the verification of every *mix* is described, first describing verification of the first re-encryption and shuffling step, and then verification of the second re-encryption and shuffling step.

Notation. Consider we have n ciphertexts posted on the BB and a mix-net consisting of i *mixes*. We use the following notation: the set of input ciphertexts of *mix* j is C_j, the set of output ciphertexts after the first re-encryption/shuffling step is C'_j, and the set of ciphertexts after the second re-encryption/shuffling step is C''_j. During verification, C_j will be divided into l blocks $a^j_1, a^j_2, \cdots a^j_l$. The corresponding output blocks (containing the same plaintexts) in C'_j are $a'^j_1, a'^j_2, \cdots a'^j_l$, the input blocks for the second verification step are $b^j_1, b^j_2, \cdots b^j_l$, and the corresponding output blocks in C''_j are $b'^j_1, b'^j_2, \cdots b'^j_l$.

[5] As [4] is intended for use in the Norwegian Internet election, its implementation must also address this case.



Receipt ID	Ciphertext
12345	xAdef
54321	TurW2
12453	Jko3$
12534	fR?p3
12534	pOels

b	b'	Final Result Mix Node j	a	a'	Media Result Mix Node j+1	b	b'	Final Result Mix Node j+1
1	2	Abjlo	2	1	bnsBt	1	2	asfop
2	2	fas6T	2	2	?3fga	2	1	HJlxS
1	1	844Ms	1	1	78ujg	1	1	la5tE
1	1	?d3da	1	2	maeit	2	2	ILmAd
2	1	Ta!32	1	1	U!33a	1	1	pOels

Fig. 1. Verification of one *mix* for 5 ciphertexts, 2 blocks

Mixing. For i *mixes* the set of input ciphertexts is divided into i subsets. To ensure the privacy of the ciphertexts, even though they are grouped, the subsets should be selected for example by district or municipality. In doing so it is prevented to divide the input ciphertexts into blocks but using the fragmentation given by the election. The j^{th} subset becomes the input of the j^{th} *mix*, which re-encrypts and shuffles the ciphertexts two times and publishes intermediate result C'_j and final result C''_j on the BB. After *mix* $j-1$ publishes its results, this becomes the input of *mix* j and the final result of the last *mix* i becomes the input of *mix* one. This is repeated until every subset has been mixed by all i *mixes.*

Verification setup. The verification parameters are set as follows: the number of blocks m is determined by $m = \lfloor\sqrt{n}\rfloor$; the size l of blocks is calculated by $l = \lfloor\sqrt{n}\rceil$); there are $r = n - m \cdot l$ blocks with *l+1* elements, and *m-r* blocks with l elements. Verification begins by generating a random permutation. For security reasons, this permutation should not be under full control of one party, and only be determined after the mixing process has finished. One possible approach is that all *mixes* and the verifier contribute a random permutation which are concatenated by the verifier at random. Another, probably more practical approach, is to generate the random permutation along with the audience of the verification process. After the random permutation has been generated, the input ciphertexts of the mix-net on the BB are permuted. This is done to ensure that the block generation for the first *mix* is not predictable.

Block fragmentation. In the next step, the input ciphertexts processed by the *mix* are grouped into m blocks. For the first *mix*, the first *l+1* ciphertexts are assigned to the first block, the second *l+1* ciphertexts to the second block, etc., until r blocks are filled. The remaining *m-r* blocks are similarly filled with l consecutive ciphertexts (instead of *l+1*). For all other *mixes* j, the input blocks are determined by the output blocks of the previous *mix j-1*, meaning $a_1^j = b_1'^{j-1}, a_2^j = b_2'^{j-1}, \ldots$. After the input ciphertexts are grouped, the verifier asks the *mix j* to show the correspondence between output blocks $a_1'^j, a_2'^j, \cdots a_l'^j$, in C_j' and input blocks $a_1^j, a_2^j, \cdots a_l^j$ in C_j. In the next step, the verifier distributes the ciphertexts of all output blocks $a_1'^j, a_2'^j, \cdots a_l'^j$ over input blocks $b_1^j, b_2^j, \cdots b_l^j$ in such a way that all ciphertexts which were in the same input block are now part of different blocks to preserve privacy. The first r input blocks contain *l+1* ciphertexts, one ciphertext of every block and one additional ciphertext of block r (input block one contains two votes of output block one, input block two contains two votes of output block two, ...). All other *m-r* blocks contain l ciphertexts, one of each block. Next, the verifier asks *mix j* to show the correspondence between output blocks $b_1'^j, b_2'^j, \cdots b_l'^j$ in C_j'', and input blocks $b_1^j, b_2^j, \cdots b_l^j$ in C_j'. This whole process is done sequentially to raise the efficiency. First the ciphertexts of *mix* one are grouped. As soon as the corresponding output blocks are known, the input blocks of *mix* two are grouped and so forth.

Mix verification. In the verification step, *mix j* proves correctness of the re-encryption of every block s during the first and second iteration of re-encryption/shuffling by using the plaintext equivalence proof proposed in [18]. As a result, any observer including the verifier can verify the correct re-encryption of the blocks and correspondingly verify whether the sum of the plaintexts remains unchanged (Figure 1).

7 Analysis

In this section we analyse our approach described in Section 6 regarding correctness, privacy, and efficiency while providing the definitions in the corresponding subsections.

7.1 Correctness

In this section we analyse the correctness of our approach by calculating the probability that ciphertexts which have been modified by a *mix* are grouped in the same block and therefore are undetected during the verification process. We first explain how ciphertexts may be altered without affecting the product of the ciphertexts. Next, we provide a formula to determine the probability of having k modified ciphertexts in one block during the verification process. This formula holds for our approach and the one proposed by Puiggalí et al.

If a *mix* wants to modify ciphertexts without being detected it needs to modify at least two ciphertexts. One is for example an additional vote for a candidate and the second one is adapted to remain the product of the plaintexts unchanged. This modification is not detected if the two adapted ciphertext are in the same block during the verification process[6].

[6] Such an attack is detected during counting if votes have no linear dependency. In this case, the decrypted vote makes no sense. However, it is not possible to determine

A *mix* receives n different ciphertexts encrypted with exponential elGamal. To add a vote for candidate c the *mix* first encrypts a vote for candidate c and add the ciphertext to the set of output ciphertexts. Following it multiplies two random ciphertexts a and b to reduce the number of overall ciphertexts. The new value is re-encrypted and add to the set of output ciphertexts. The overall product stays unchanged if all ciphertexts are assigned to the same block.

Theorem 1. *Consider we have n ciphertexts, a block size of l, in total m blocks with $m \cdot l + r = n$, r blocks with l+1 elements, and (m-r) blocks with l elements. The number of possible fragmentation is:*

$$\frac{\binom{n}{l+1}\binom{n-(l+1)}{l+1}\cdots\binom{n-(r-1)(l+1)}{l+1}}{r!} \cdot \frac{\binom{n-r(l+1)}{l}\binom{n-r(l+1)-l}{l}\cdots\binom{l}{l}}{(m-r)!}.$$

Proof. There are $\binom{n}{l+1}$ ways of choosing *l+1* ciphertexts (disregarding order) for the first block out of n ciphertexts. After the first block was chosen there are *(n-(l+1))* ciphertexts left which have to be spread over *(m-1)* blocks. The same is done for all blocks with l elements. Finally the total among of fragmentation is divided by the number of block-permutations *r!* (and correspondingly *(m-r)!*) because the order of the blocks is unregarded.

Theorem 2. *Consider we have n ciphertexts, r blocks with a blocksize of l+1, m-r blocks with a blocksize of l, and k modified ciphertexts.*

1. *If $k > l+1$ the probability of having k particular ciphertexts in one block is zero.*
2. *If k = l+1 the probability of having k particular ciphertexts in one block with l+1 elements is $\frac{r}{\binom{n}{l+1}}$.*
3. *If k = l the probability of having k particular ciphertexts in one block with l elements is $\frac{m-r}{\binom{n}{l}}$.*
4. *if $k \leq l$ the probability of having k particular ciphertexts in one block is $\binom{n-k}{l+1-k} \cdot \frac{r}{\binom{n}{l+1}}$ for blocks with l+1 elements and $\binom{n-k}{l-k} \cdot \frac{m-r}{\binom{n}{l}}$ for blocks with l elements.*
5. *The average probability of having k modified elements in one block is* $\binom{n-k}{l+1-k} \cdot \frac{r^2}{m\binom{n}{l+1}} + \binom{n-k}{l-k} \cdot \frac{(m-r)^2}{m\binom{n}{l}}$.

Proof. The probabilities can be calculated in the following way

1. Not all modified ciphertexts can be in one block if the number k of modified ciphertexts is larger than the blocksize l and *l+1*.
2. Consider all k modified ciphertexts are in one block with *l+1* elements. There are *n-(l+1)* ciphertexts left which have to be segmented in *(m-1)*

which *mix* tried to manipulate the result. In the case that valid votes are linearly dependent (e.g. by encrypting the position of the chosen candidate in the candidate list) this modification cannot be detected either by decrypting and checking the format nor by the verifier of the mix-net.

several blocks: $\frac{\binom{n-(l+1)}{l+1}\cdots\binom{n-(r-1)(l+1)}{l+1}}{(r-1)!} \cdot \frac{\binom{n-r(l+1)}{l}\binom{n-r(l+1)-l}{l}\cdots\binom{l}{l}}{(m-r)!}$. Dividing by the total among of possible segmentation the resulting probability is $\frac{r}{\binom{n}{l+1}}$.

3. The probability for blocks with l elements can be shown in the same way as for blocks with *l+1* elements.
4. Consider $k = l$. A block with l elements, containing the k ciphertexts can be generated in $\frac{m-r}{\binom{n}{l}}$ ways and a blocks with *l+1* elements in $\frac{r}{\binom{n}{l+1}}$ ways. In addition to the number calculated for blocks with *l+1* elements, there are several possibilities of generating the block containing the modified ciphertexts because the remaining *(l+1-k)* blockelements can be chosen from the remaining *(n-k)* ciphertexts in $\binom{n-k}{l+1-k}$ different ways. This leads to a probability of $\binom{n-k}{l+1-k} \cdot \frac{r}{\binom{n}{l+1}}$. The probability for blocks with l elements and $k < l$ can be shown in the same way.
5. The probability that k ciphertexts are in one block with *l+1* elements during the verification is $\frac{r}{m} \cdot \binom{n-k}{l+1-k} \cdot \frac{r}{\binom{n}{l+1}}$ and in one block with l elements is $\frac{m-r}{m} \cdot \binom{n-k}{l-k} \cdot \frac{m-r}{\binom{n}{l}}$. This results in an overall probability of $\binom{n-k}{l+1-k} \cdot \frac{r^2}{m\binom{n}{l+1}} + \binom{n-k}{l-k} \cdot \frac{(m-r)^2}{m\binom{n}{l}}$ that modified k ciphertexts are in the same block during the verification and remains undetected.

In our approach the values for m and l are fix and can be calculated by $m = \lfloor\sqrt{n}\rfloor$ and $l = \lfloor\sqrt{n}\rceil$). As a result the correctness is independent of the number of *mixes i*. In contrast the values for the approach proposed by Puiggalí et al. depends on the number of *mixes* and can be calculated by $l = \sqrt[i]{n}$ and $m = \frac{n}{l}$.

7.2 Privacy

In this section we analyse the privacy of our approach. First, we explain how privacy could be violated or decreased during the verification process. Afterwards, we analyse our approach under the assumption that just one *mix* is honest.

The whole verification process, inclusive the block fragmentation of the ciphertexts, is public and published on the BB. Consider we have just one *mix*, group the input of the *mix* in two blocks, and ask the *mix* to show the corresponding two blocks of the output ciphertexts than after the decryption we can see which block contained which cast vote. Correspondingly, if we know that the vote by Alice was contained in block one and the encrypted ciphertexts of block one do not include a vote for Bob, than we know that Alice did not cast a vote for Bob.

While this problem exists with the proposal in [4], we show that this information is not leaked in our approach. For the following theorem we assume that no additional information which increases the probability of linking the output cipher to a given input cipher (for example no ballot papers with different or slightly different candidate lists are used).

Theorem 3. *For a block size $l = \sqrt{n}$ and a total number n of ciphertexts, the probability that a certain input ciphertext of the mix-net corresponds to a certain output ciphertext of the mix-net varies up to $\frac{1}{(l+1)^2}$.*

Proof. Consider the block size l is the root of the total number n of ciphertexts and we want to trace a certain input ciphertext which is after the first verification step of *mix* one in block $a^1_j = a'^1_j$. During the second verification step of *mix* one the ciphertexts of this block are equally distributed. All input blocks b^1 contain one vote of all output blocks a'^1. Therefore, also the probability of one block $b^1_j = b'^1_j$ to contain a certain input ciphertext is equally distributed and correspondingly the probability for all output ciphertexts of this *mix* is equal.

If the block size l is not a root of the total number of ciphertexts n then some blocks contain one ciphertext more than the others. As a result some blocks b^1 of the second verification step contain l ciphertexts and some blocks $l+1$ ciphertexts where two ciphertexts come from the same block a'^1, generated during the first verification step of one *mix*. Consider we want to trace a certain input ciphertext of the *mix* which goes during the first verification step to a block with $l+1$ elements $a^1_j = a'^1_j$. Note, we do not consider the case that a ciphertext is traced which was part of a block with l elements because these items are equally distributed as shown before. All output ciphertexts of block a'^1_j have a probability of $\frac{1}{l+1}$ of being this particular input ciphertext. After the second verification step of one *mix*, one of the blocks with $l+1$ elements $b^1_{j'} = b'^1_{j'}$ contains two ciphertexts of this block a'^1_j, all other blocks just one. The output ciphertexts of a block with l elements have a probability of $\frac{1}{l+1} \cdot \frac{1}{l} = \frac{1}{(l+1)l}$. The output ciphertexts of a block with $l+1$ elements, which contains just one ciphertext of the block a'^1_j, have a probability of $\frac{1}{l+1} \cdot \frac{1}{l+1} = \frac{1}{(l+1)^2}$. Finally the output ciphertexts of the one block $b^1_{j'}$ with $l+1$ elements, which contains two ciphertexts of the specific block a'^1_j, have a probability of $\frac{2}{l+1} \cdot \frac{1}{l+1} = \frac{2}{(l+1)^2}$. Because of $\frac{2}{(l+1)^2} > \frac{1}{(l+1)l} > \frac{1}{(l+1)^2}$, the probability varies up to $\frac{1}{(l+1)^2}$.

During the verification of *mix i* we have one block with $l+1$ elements a^i_k containing two ciphertexts of block $b'^{i-1}_{j''}$, generated during the verification of *mix i-1*. The output ciphertexts of block a'^i_k have the highest probability to be the traced ciphertext, in fact $\frac{1}{(l+1)^i}$ higher than for all other ciphertexts. During the second verification step the ciphertexts are equally distributed while one block $b^i_{k'}$ receives two ciphertexts of block a'^i_k. The probability of all output ciphertexts of this block $b^{i'}_{k'}$ will be about $\frac{1}{(l+1)^i} \cdot \frac{1}{(l+1)} = \frac{1}{(l+1)^{i+1}}$ higher than for all other output ciphertexts of *mix i*. This shows that the difference in the probability is reduced if the number of honest *mixes* increases and the maximum that can be achieved for one *mix* is $\frac{1}{(l+1)^2}$.

Further note that the difference between the probabilities decreases with the number of honest *mixes*.

The fact that the whole set of ciphertexts is divided into subsets, to enable a parallel processing of the *mixes* during the mixing process, does not affect the privacy. This holds because we assume that the set of ciphertexts contained in different subsets are from different districts or municipalities and differ anyway in their parties, candidate lists, or number of votes.

7.3 Efficiency

In this section the efficiency of our approach is determined. Note, we only consider the number of needed exponentiations because performing any other arithmetic operation requires less computational effort. The total number of needed exponentiations is determined by three components: the mixing phase, proof generation by the mix-net and verification by the verifier. We compute the computational costs only for one *mix*. It should be noted that due to the parallelization, all steps can be executed in parallel.

The computational cost for one *mix* during the mixing depends on the number of ciphertexts n. Two exponentiations are needed to re-encrypt one ciphertext (α, β) and correspondingly $4n$ for all n ciphertexts for both steps. The computational cost to verify the plaintext equivalence depends on the number of blocks. For n ciphertexts $m = \lfloor \sqrt{n} \rfloor$ blocks are used. During proof generation the *mix* needs one exponentiation per block to calculate the witness. From this follows that for m blocks $2m$ exponentiations are needed (m for each mixing step). Afterwards the verifier needs two exponentiations per block to check the integrity, what gives a total of $4m$ exponentiations to verify all blocks in both verification steps.

8 Comparison

In this section we compare our approach with all so far known verification processes which on one hand ensure correctness with just a hight probability but on the other hand have less computational costs. Therefore we are looking at the approach of Puiggalí et al.[4](Section 4), RPC [19] (Section 3.1) and PoS (Section 3.2).

8.1 Correctness

The correctness is measured by the probability of modifying ciphertexts by a *mix* without detection. The probability of every *mix* to change k ciphertexts when RPC is used is $\frac{1}{2^k}$[19]. The probability of doing an undetected modification is about $(\frac{5}{8})^\gamma$[6] (for $\gamma \leq 5$) and therefore at minimum $(\frac{5}{8})^5$ if the mix-net is verified by PoS. The approach by Puiggalí et al. can just be used if the ciphertexts can be distributed equally. In this case the probability of having k modified ciphertexts in the same block is the same than for our verification process. The comparison shows that the approach by Puiggalí et al. and our improved version has the lowest probability of doing an undetected modification of ciphertexts.

8.2 Privacy

The privacy is determined by the size of the subset of output ciphertexts a certain input ciphertext is hidden in. We consider the case that only one mix-net is honest and keeps the input-output ciphertext relation secret. After using RPC for verification all ciphertext of one *mix* are divided into two blocks. One which includes the ciphertexts with a revealed relation between the input ciphertexts of the *mix* and the media result and one with a revealed associations between the media result and the output ciphertexts of the *mix*. Therefore it is possible to say which ciphertext is contained in which block what results in a subset of $\frac{n}{2}$ output ciphertexts where a certain input ciphertext is hidden in. Using PoS the ciphertexts are grouped in up to 5 random blocks and the paper proves that the average size of the intersections in this case is $\frac{n}{2^5}$. Therefore after the verification of one *mix* it can be specified in which $\frac{n}{10}$ ciphertexts a specific input ciphertext is contained. The approach proposed by Puiggalí et al. reduce the blocksize dependent on the number of used *mixes*. For i *mixes* a blocksize of $\sqrt[i]{n}$ is used and following a certain input ciphertext is hidden among $\sqrt[i]{n}$ ciphertexts. The comparison shows that our approach has the highest privacy.

8.3 Efficiency

Efficiency is compared by the number of used exponentiations. For re-encryption our approach, like RPC, needs twice as many exponentiation per *mix* as the approach by Puiggalí et al. and PoS, as re-encryption and shuffling are performed twice. But the impact of this is reduced as the *mixes* all process a subset of ciphertexts in parallel.

During the verification of RPC two times the association between $\frac{n}{2}$ ciphertexts is shown. This can be done by revealing the random value and be verified by recalculating the re-encryption. Therefore two times $\frac{n}{2}$ exponentiations for α and two times $\frac{n}{2}$ for β are needed. In total the computational costs per *mix* are $2 \cdot 2 \cdot \frac{n}{2} = 2n$ exponentiations. The number of exponentiations during the PoS verification is $2\gamma(2i - 1)$[6] per *mix* for a total number of i *mixes* and depends on the security parameter γ (for $\gamma \leq 5$). Therefore the maximum number of exponentiations per *mix* is $10(2i-1)$ which is achieved if $\gamma = 5$. In the verification process by Puiggalí et al. a zero knowledge proof is used to show the correctness of every block. In [4] they estimate a total number of $6\frac{n}{\sqrt[i]{n}}$ exponentiations done by the mix-net and the verifier . We see in Figure 2 that only PoS is faster than our approach (CoRB) because the number of exponentiations does not depend on the number of input ciphertexts of the mix-net.

9 Conclusion and Future Work

We discussed the mix-net verification scheme by Puiggalí et al., a mix of Randomized Partial Checking (RPC) and Optimistic Mixing (OM). We highlighted several possibilities to improve efficiency, identified a privacy risk in case just

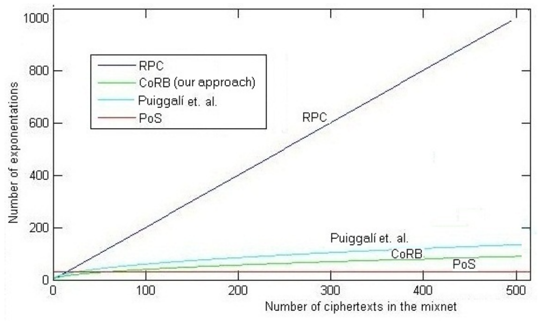


Fig. 2. Comparison of efficiency for one *mix*

one mix-net is honest (keeping the re-encryption and shuffling secret), and noted several unclarities concerning verification block size and allocation of elements to verification blocks. We proposed an improved verification scheme, based on randomized partial checking of blocks, to address these issues. We provided a detailed analysis of the effectiveness (in terms of privacy, efficiency and correctness) of our scheme and compared this with other schemes that enable a trade off between privacy, correctness and efficiency. We showed that the privacy and correctness of our scheme improve upon that offered by RPC and OM, as well as other approaches that offer a trade off between efficiency, privacy and correctness. In addition, our scheme is less computationally expensive than RPC. Specifically, our scheme provides a high probability of correctness for all elements for low computational cost. This contrasts starkly with RPC, which validates some elements at an elevated computational cost. Moreover, our scheme breaks the link between input and output. Finally, the probability that a specific input element ends up in a specific output block is almost equal for all output blocks.

There are several directions in which this work can be extended further. In this paper we did not address malicious inputs. These could occur e.g. in the case of a coerced voter. Finally, we're interested in applying this verification approach to improve the efficiency of an actual mix-net, such as Verificatum[7]. We also plan to discuss which probabilities satisfy legal requirements with legal scientists.

References

1. Abe, H.: Remarks on mix-network based on permutation networks. In: In proceedings of PKC 01, LNCS series. LNCS, vol. 1992, pp. 317–324. Springer-Verlag (2001)
2. Abe, M.: Universally verifiable mix-net with verification work independent of the number of mix servers. In: Proceedings of EUROCRYPT 1998. pp. 437–446. Springer-Verlag, LNCS 1403 (1998), `http://www.springerlink.com/content/hl8838u4l9354544/`

[7] `http://www.verificatum.com/`

3. Abe, M.: Mix-networks on permutation networks. In: Proceedings ASIACRYPT'99. LNCS, vol. 1716, pp. 258–273. Springer-Verlag (1999)
4. Allepuz Puiggalí, J., Castelló Guasch, S.: Universally verifiable efficient re-encryption mixnet. In: Proceedings EVOTE 2010. LNI, vol. 167, pp. 241–254. GI (2010)
5. Baudron, O., Fouque, P.A., Pointcheval, D., Stern, J., Poupard, G.: Practical multi-candidate election system. In: In PODC. pp. 274–283. ACM Press (2001)
6. Boneh, D., Golle, P.: Almost entirely correct mixing with applications to voting. In: Proceedings of the 9'th ACM conference on Computer and Communications Security (CCS) (2002)
7. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM 24(2), 84–88 (1981)
8. Chaum, D., Pedersen, T.: Wallet databases with observers. In: Brickell, E. (ed.) CRYPTO'92, LNCS, vol. 740, pp. 89–105. Springer Verlag (1993)
9. Damgard, I., Koprowski, M.: Practical threshold rsa signatures without a trusted dealer. pp. 152–165. Springer Verlag (2000)
10. Desmedt, Kurosawa: How to break a practical mix and design a new one. In: Proceedings of the 19th international conference on Theory and application of cryptographic techniques. LNCS, vol. 1807, pp. 557–572. Springer-Verlag, Berlin, Heidelberg (2000), `http://dl.acm.org/citation.cfm?id=1756169.1756223`
11. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography. In: SIAM Journal on Computing. pp. 542–552 (2000)
12. El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Proceedings CRYPTO 84. pp. 10–18. Springer-Verlag New York, Inc., New York, NY, USA (1985)
13. Furukawa, J.: Efficient and verifiable shuffling and shuffle-decryption. vol. 88-A, pp. 172–188 (2005)
14. Furukawa, J., Sako, K.: An efficient scheme for proving a shuffle. In: Proceedings of CRYPTO 01. pp. 368–387. Springer-Verlag (2001)
15. Golle, P., Zhong, S., Boneh, D., Jakobsson, M., Juels, A.: Optimistic mixing for exit-polls. In: Asiacrypt 2002, LNCS 2501. pp. 451–465. Springer-Verlag (2002)
16. Groth, J.: A verifiable secret shuffle of homomorphic encryptions. vol. 23, pp. 546–579 (2010)
17. Groth, J., Lu, S.: Verifiable shuffle of large size ciphertexts. In: In proceedings of Practice and Theory in Public Key Cryptography - PKC 07. LNCS, vol. 4450, pp. 377–392 (2007)
18. Jakobsson, M., Juels, A.: Millimix: Mixing in small batches. Tech. rep., Center for Discrete Mathematics; Theoretical Computer Science (1999)
19. Jakobsson, M., Juels, A., Rivest, R.L.: Making mix nets robust for electronic voting by randomized partial checking. In: Proceedings of USENIX'02 (2002)
20. Neff, A.C.: Verifiable mixing (shuffling) of elgamal pairs. Tech. rep., In proceedings of PET '03, LNCS series (2003)
21. Neff, C.A.: A verifiable secret shuffle and its application to e-voting. In: CCS'01. pp. 116–125. ACM, New York, NY, USA (2001)
22. Ogata, W., Kurosawa, K., Sako, K., Takatani, K.: Fault tolerant anonymous channel. In: Proceedings of ICICS'97. pp. 440 – 444. LNCS 1334 (1997)
23. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Advances in Cryptology - EUROCRYPT 99. Lecture Notes in Computer Science, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
24. Peng, K., Boyd, C., Dawson, E., Viswanathan, K.:

25. Sako, K., Kilian, J.: Receipt-free mix-type voting scheme. In: Guillou, L., Quisquater, J.J. (eds.) Proc. EUROCRYPT'95. LNCS, vol. 921, pp. 393–403 (1995)
26. Schnorr, C.P.: Efficient signature generation by smart cards. Journal of Cryptology 4, 161–174 (1991), http://dx.doi.org/10.1007/BF00196725
27. Terelius, B., Wikström, D.: Proofs of restricted shuffles. In: AFRICACRYPT. LNCS, vol. 6055, pp. 100–113 (2010)
28. Wikström, D.: A commitment-consistent proof of a shuffle. In: Proceedings of the 14th Australasian Conference on Information Security and Privacy, LNCS, vol. 5594, pp. 407–421. Springer-Verlag, Berlin, Heidelberg (2009)