

Random Block Verification: Improving the Norwegian Electoral Mix Net

Denise Demirel^{1,2}, Hugo Jonker³, Melanie Volkamer^{1,2}

¹ CASED, Darmstadt

² SeCuSo group, TU Darmstadt

³ SaToSS group, University of Luxembourg

Abstract. The VALG project is introducing evoting for municipal and county elections to Norway. Part of the evoting system is a mix net along the lines of Puiggalí et al.– a mix net which can be efficiently verified by combining the benefits of optimistic mixing and randomized partial checking. This paper investigates their mix net and proposes a verification method which improves both efficiency and privacy compared to Puiggalí et al..

1 Introduction

To ensure anonymity, evoting systems need to incorporate a mechanism to break the link between voter and cast vote. One popular method is the use of mix nets [2], which shuffle the list of encrypted votes while changing the appearance of the ciphertexts and keeping the used permutation secret. To reduce the trust assumption, universally verifiable mix nets have been developed [15, 4, 19, 13, 10]. In voting, efficiency is a prime concern. 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. Attempts at efficiency improvement did not raise the bar sufficiently for such a demanding task. Two separate directions in verification sought to address this: Optimistic Mixing (OM, [9]) and Randomized Partial Checking (RPC, [12]).

Intuitively, OM achieves its speed-up of verification by proving correct mixing for the whole group of inputs: the mix proves that the product of the input ciphertexts is equal to the product of the output ciphertexts (see Figure 1a). While more efficient (only one proof is needed, instead of one per input), some fraud is not detected (intuitively, $4 \cdot 6 = 3 \cdot 8$). The proposal by Golle et al. [9] uses double encryption and a cryptographic checksum to prevent this attack, however, Wikström identified [18] multiple fatal flaws in their particular design. Another optimistic approach by Boneh and Golle, Proof of Subproduct (PoS, [1]), is slightly faster as it does not use a cryptographic checksum nor double encryption. 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.

RPC lets each mix node first produce an intermediate shuffle, and then shuffle again to produce the final result. For each element of the intermediate result,

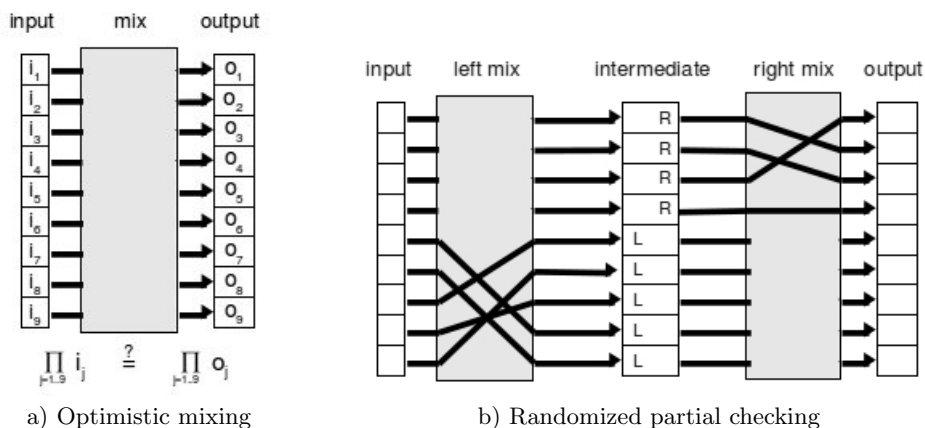


Fig. 1. Two approaches to trading verification for efficiency in mix nets

a coin is flipped to reveal the link to either its corresponding input (heads) or output (tails) element (see Figure 1b). This approach doesn't require any proof (just revealing half the rerandomization values used), but there's a 50% chance per element for the mix to cheat undetected.

Puiggalí et al. combined the advantages of OM and RPC to arrive at a mix net design that improves upon privacy and verifiability while retaining efficiency. Their work was incorporated into the Norwegian Evote Project⁴ and used for a limited number of municipality elections in Norway. In the recent past, advances have been made in efficient provably secure mixing (e.g., [19, 10, 17]). However, these approaches do not align with the current Norwegian implementation. Our goal is to propose an improved verification approach that remains close to that design, such that the current implementation may be easily updated.

Contribution. The contribution of this paper is twofold. Firstly, this paper identifies several areas for improvement (including a privacy weakness) in the scheme proposed by Puiggalí et al.. These improvements are incorporated into Random Block Verification (RBV), a scheme which is more efficient, more secure, and more precisely detailed. The architecture of RBV remains sufficiently close to the scheme by Puiggalí et al. to allow for easy adoption into the Norwegian system. Secondly, we analyze verifiability, privacy and efficiency of RBV, and compare these properties to properties of other mix nets that offer a trade off between verifiability and efficiency.

Structure of the paper. The rest of this paper is structured as follows: we first discuss some ElGamal mix nets (Section 2). As this work improves the contributions of Puiggalí et al, their research is discussed in more detail (Sections 3).

⁴ <http://www.regjeringen.no/en/dep/krd/prosjekter/e-vote-2011-project/about-the-e-vote-project.html>

Possible improvements to the verification process are discussed in Section 3.1, all of which are implemented by the new verification process detailed in Section 4. Correctness, privacy, and efficiency of the newly proposed verification process are determined in Section 5 and compared to other mix nets that trade off privacy for efficiency. This is followed by conclusions and future work in Section 6.

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.

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 [5]. Consider two large primes p and q , where $q \mid p - 1$. 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 $r_1 \in Z_q$.

To ensure anonymity, the votes are processed by 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. A re-encryption mix net with m mix nodes works as follows: The first mix node 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 $r_2 \in Z_q$: $ReEnc_y = ((\alpha, \beta), r_2) = (\alpha g^{r_2}, \beta y^{r_2}) = (g^{r_1} g^{r_2}, g^s y^{r_1} y^{r_2}) = (g^{r_1+r_2}, g^s y^{r_1+r_2}) = (\alpha', \beta')$. (Note that while the plaintext remains unchanged, the ciphertext is completely altered.) Next, 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 node 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 node is honest and keeps the permutation secret. In order to also ensure that mix nodes cannot cheat by replacing encrypted votes with new ones, verifiability needs to be implemented, ideally without decreasing the level of privacy.

3 Norwegian mix net by Puiggalí et al.

In [14], Puiggalí et al. describe an approach to verify a re-encryption mix net (with exponential elGamal) which combines the idea of optimistic mixing and RPC. This verification is executed after the last mix node has published its output on the Bulletin Board. The analysis of the Norwegian Election system [8] treated this mix net as a solid building block. Nevertheless, there is room for improvement – in particular, verification efficiency of the mix net can be improved.

Below their verification process is described and several points for improvement are highlighted.

The Puiggalí et al. verification process operates as follows:

1. An independent verifier provides a random permutation (the challenge) of all input votes of the first mix node.
2. To verify, the list of votes is divided into $l = \sqrt[m]{n}$ equally-sized blocks, for m mix nodes and n input ciphertexts (i.e., votes). Since l is well defined, this can be executed by either the independent verifier, the BB, or the mix node.
3. For every input block, the first mix node identifies the corresponding output block. Moreover, for every block, the mix node publishes the product of the ciphertexts in that block. Finally, the mix node publishes a zero knowledge proof (e.g. using the Chaum-Pedersen protocol [3] or Schorr’s signature scheme [16]) to prove that the ciphertext product of the input block is equal to that of the corresponding output block.
4. The verifier checks the proofs of the first mix node.
5. This process continues for each mix node, where the assignment of nodes to blocks depends on the previous node’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 node is composed of at least one ciphertext of every input block of the first mix node. 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 . Note that such a manipulation will remain undetected if a malicious mix node 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.

3.1 Remarks

There are some remarks to this approach, discussed below. Corresponding improvements are sketched in this section and worked out in Section 4.

Inefficient zero-knowledge proofs. In [14], the correct processing of each block is proven with computationally costly zero knowledge proofs. A more efficient solution is to publish the sum of the random values used for the re-encryption per block. As this does not reveal anything but random noise, this value can serve as a zero knowledge proof. This is very efficient (as it does not require any zero knowledge proof). However, proving that this does indeed not reveal any usable information whatsoever in a mathematically rigid fashion is an open question. Therefore, as an alternative while work on this proof continues, is to use efficient zero knowledge proofs as those from [11]. With this improvement, proof generation and verification requires either 2 exponentiations per block (re-encrypting the ciphertext of the block’s “sum” with the claimed randomness), or 3 exponentiations (1 for proof generation, 2 to verify the zero knowledge proof).

Therefore, to verify all blocks of one mix node, this would require either $\frac{n}{\sqrt[m]{n}}$ exponentiations or $3 \cdot \frac{n}{\sqrt[m]{n}}$ exponentiations for all blocks of a mix node (where m is the total number of mix nodes). Both improve upon the $6 \cdot \frac{n}{\sqrt[m]{n}}$ exponentiations needed by Puiggalí et al. to generate the proofs (2 exponentiations) and verify (4 exponentiations) each of these for n ciphertexts and m mix nodes.

Introducing parallelisation. During the mixing process every mix node of the mix net re-encrypts and shuffles the input ciphertexts. The origin idea of Puiggalí et al. was to process the encrypted votes by one mix node after the other. It is possible to speed up this process by parallelizing in the following way: the set of input ciphertexts is divided into m subsets (where m is the number of mix nodes). Then all mix nodes start with one of the subsets and forward that to their neighbor after shuffling. This improvement⁵ increases the efficiency by factor m .

Reducing trust assumptions. Optimal privacy in [14] is only ensured if all mix nodes are honest. However, this is not the idea of a mix net where privacy should be ensured if one single mix node is honest. Therefore, we propose to build single mix nodes similar to RPC where each mix node shuffles twice.

Furthermore, correctness in [14] depends on the assumption that the verifier and the first mix node do not maliciously collaborate. (Otherwise, the first mix knows what the block selection will be and therefore knows how to cheat undetectably). As such, it is essential for correctness that the challenge is unpredictable and generated after the mixing process. We sketch a method for ensuring this.

Clarifying block sizes. The approach by Puiggalí et al. assumes that the total number of ciphertexts can be grouped in equally sized blocks with block size $l = \sqrt[m]{n}$, for m mix nodes and n votes. In general, there will be a remainder when computing l . We make this explicit⁶ and incorporate its handling into our design.

4 RBV: Verifying Integrity of Random Blocks

In this section we describe Random Block Verification, a mix net with a detailed verification process, based on the proposal of Puiggalí et al., which includes all improvements proposed above.

Notation. In the remainder of this section, we consider n ciphertexts posted on the bulletin board (BB) and a mix net consisting of m mix nodes. We use the following notation: the set of input ciphertexts of mix node j is C_j , the set of output ciphertexts after the first re-encryption/shuffling step is C'_j , and the

⁵ This improvement was implemented for the Norwegian voting trials.

⁶ The Norwegian implementation of [14] addresses this as well.

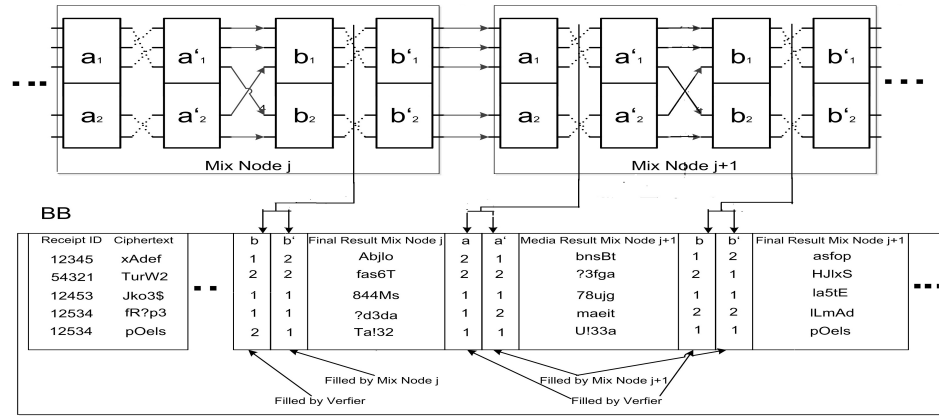


Fig. 2. Verification of one Mixnode for 5 ciphertexts, 2 blocks

set of ciphertexts after the second re-encryption/shuffling step is C_j'' . During verification, C_j will be divided into l blocks $a_1^j, a_2^j, \dots, a_l^j$. The corresponding output blocks (containing the same plaintexts) in C_j' are $a_1^j, a_2^j, \dots, a_l^j$, the input blocks for the second verification step are $b_1^j, b_2^j, \dots, b_l^j$, and the corresponding output blocks in C_j'' are $b_1^j, b_2^j, \dots, b_l^j$.

Mixing. For m mix nodes the set of input ciphertexts is divided into m subsets. To ensure the privacy of the ciphertexts, even though they are grouped, the subsets should be selected for example by district or municipality. The j^{th} subset becomes the input of the j^{th} mix node, 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 node $j - 1$ publishes its results, this becomes the input of mix node j and the final result of the last mix node m becomes the input of mix node one. This is repeated until every subset has been mixed by all m mix nodes.

Verification setup. The verification parameters are set as follows: the number of blocks l is determined by $l = \lfloor \sqrt{n} \rfloor$; there are $r = n - l \cdot l$ blocks with $l + 1$ elements, and $l - r$ blocks with l elements. Verification begins by generating a random distribution of ciphertexts over verification blocks.

Distributing ciphertexts over blocks. Each mix node is verified in an optimistic fashion: both input and output ciphertexts are grouped into blocks, and equivalence of the blocks is proven. As remarked above, if the assignment of ciphertexts to blocks is known to the mix node prior to mixing, the mix knows how to cheat without being detected. Hence, this initial distribution must be generated randomly. Puiggalí et al. rely on an independent party to provide an initial random distribution. In contrast, we leverage the Fiat-Shamir technique [6] to determine

how ciphertexts are grouped into blocks. Simply put, the first verifier computes the hash of its own output, and uses that as the seed for a publicly known random number generator. The resulting random stream is then used to assign ciphertexts randomly to blocks for the first mix (see Appendix A for details). As Fiat and Shamir point out [7], there is no way to tweak the input to the hash function to get a predictable output. Therefore, the resulting output is sufficiently unpredictable for the first mix and may be used as described. For all other mix nodes j , the input blocks are determined by the output blocks of the previous mix node $j - 1$, meaning $a_1^j = b_1^{j-1}, a_2^j = b_2^{j-1}, \dots$.

After dividing the input ciphertexts into blocks, the mix node proves the correspondence between input block a_1^j and output block a_1^j , between input block a_2^j and output block a_2^j , etc. In the next step, the verifier distributes the ciphertexts of the output blocks $a_1^j, a_2^j, \dots, a_l^j$ over input blocks $b_1^j, b_2^j, \dots, b_l^j$. As each block contains roughly as many ciphertexts as there are blocks, this is done to maximize privacy: the blocks of the input are chosen such that each input block b_y^j contains one ciphertext from every output block a_x^j .

Of course, there are two block sizes: l and $l + 1$. So, to be specific: 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, \dots). All other $l - r$ blocks contain l ciphertexts, one from each block. Then mix node j proves the correspondence between output blocks $b_1^j, b_2^j, \dots, b_l^j$ and input blocks $b_1^j, b_2^j, \dots, b_l^j$.

Verifying blocks. To verify that a block of input ciphertexts was correctly processed by a mix node, there are two options. Either the node reveals the sum of the used re-encryption random numbers (believed to be secure, but not proven so), or the node uses the zero knowledge proofs of [11]. In either case, the node proves that the sum of the plaintexts of the block was not changed by the mixing step (Figure 2).

5 Analysis

In this section we analyse Random Block Verification regarding fraud detection, privacy, and efficiency. In addition, the results are compared with those of of Randomized Partial Checking, the Proof of Subproduct mix by Golle et al., and the “Norwegian mix” by Puiggalí et al.

5.1 Detecting malicious mixes

The optimistic verification approach is not perfect – an error (e.g., changing a 1 to a 3) can be counterbalanced (e.g., $1 + 4 = 3 + 2$) and pass undetected. To achieve undetected corruption of the mix result, a malicious mix has to change (drops, alters, inserts) at least two ciphertexts to ensure balancing the introduced error. This will remain undetected *if and only if* the introduced errors are

properly balanced within the same block. Since the division of ciphertexts into blocks is not known to the mix during mixing, the malicious mix cannot ensure this. Below, we investigate the probability of this happening by chance. As an aside, note that in any optimistic approach, a change must be counterbalanced. Therefore, to affect a change of k votes, at least one ciphertext extra has to be tweaked, leading to at least $k + 1$ changed ciphertexts. This is in contrast to RPC, where changes to ciphertexts cannot be balanced by other changes. Hence, below we compare the chance of changing k ciphertexts in RPC to $k + 1$ ciphertexts in optimistic approaches.

Randomized Partial Checking. To cheat, a mix would drop/alter a ciphertext either in the first or in the second mixing stage. Since the mix has to reveal either the first or the second mixing stage, the chance of getting away with this is $\frac{1}{2}$. Since this is independent, the chance of remaining undetected for k changes is

$$P_{rpc}(k \text{ undetected changes}) = 2^{-k}.$$

Proof of Subproduct. During the verification α random blocks (for $\alpha \leq 5$) are generated with an average size of $\frac{n}{2}$ and compared with the corresponding output blocks. In case a malicious mix node adapted k ciphertexts, the prover has to find another set of output ciphertexts which has the desired properties. The chance of doing this in polynomial time is at most $(\frac{5}{8})^\alpha [1]$. Thus a high number of used random blocks increases the probability that the modified ciphertext is checked. For $\alpha = 5$, for instance, the chance of getting away is $(\frac{5}{8})^5$. The maximum probability of changing k ciphertexts without detection is reached at $\alpha = 1$. Thus, in general,

$$P_{PoS}(k + 1 \text{ undetected changes}) = \left(\frac{5}{8}\right)^\alpha.$$

Norwegian mix. Puiggalí et al. claim in [14] that the chance of not detecting that two ciphertexts have been altered by one mix is $P(\text{undetected}) = \frac{l-1}{n-1}$ – the first ciphertext can be in any block, as long as the second is in the same. Given that in their proposal, $l = \sqrt[m]{n}$ (with m being the number of mixes), this gives the following chance of changing $k + 1$ ciphertexts without being detected:

$$P_{Norway}(k + 1 \text{ undetected changes}) = \left(\frac{\sqrt[m]{n} - 1}{n - 1}\right)^k.$$

Random Block Verification. The chance of affecting a change of size k requires changing $k + 1$ ciphertexts. In the case of two changed ciphertexts, the RVB mix net performs as good as Puiggalí et al.. In case of more than two, the Norwegian mix net performs slightly better, as their block size is inversely proportional to the number of mix nodes, whereas ours is constant in this regard. Intuitively, our approach has \sqrt{n} blocks of (almost) equal size, and therefore the chance of a ciphertext occurring in one block is roughly $(\sqrt{n})^{-1}$. The chance of $k + 1$

ciphertexts occurring in the same block is therefore roughly $(\sqrt{n})^{-k}$. In reality, it is slightly better as some blocks are smaller than others. To be precise,

$$P_{rbv}(k + 1 \text{ undetected changes}) = \left(\frac{\sqrt{n} - 1}{n - 1} \right)^k.$$

In RBV, the values for m and l are fixed at $m = l = \lfloor \sqrt{n} \rfloor$. As a result the correctness is independent of the number of mix nodes m . In contrast the values for the approach proposed by Puiggalí et al. depends on the number of mix nodes and are given by $l = \sqrt[m]{n}$ and $m = \frac{n}{l}$.

5.2 Privacy

In mix nets, privacy is the question of how traceable a given ciphertext is through the mix net. In general, there remains some imprecision – some output ciphertexts can be ruled out, but others may or may not be a re-encryption of the sought ciphertext. The size of the group that cannot be ruled out (which we will call “Anonymity group” or AG) gives a measure for how much privacy is achieved by the mix net. In the following we consider the case that only one mix net is honest and keeps the input-output ciphertext relation secret.

Randomized Partial Checking. Depending on a coin flip, the verification procedure reveals either the link between an intermediate ciphertext and the input, or its link with an output ciphertext. In the worst case, the coin is completely fair, and hence 50% of the links with input ciphertexts are linked, and similarly 50% of the links with output ciphertexts. Hence, $\frac{n}{2}$ output ciphertexts are not linked yet – and must belong to the input ciphertexts whose link was revealed. Thus, for each ciphertext whose input link is revealed, the anonymity group has a size of $\frac{n}{2}$. A similar reasoning holds for ciphertexts whose output link is revealed. Thus, the anonymity group of an RPC mix net with one honest mix is

$$|AG_{rpc}| = \frac{n}{2}.$$

Proof of Subproduct. Using PoS the ciphertexts are grouped in up to α random blocks (with α being the security parameter, $0 < \alpha \leq 5$). The authors show that the average anonymity group size is

$$|AG_{PoS}| = \frac{n}{2^\alpha}.$$

Thus, increasing the security (i.e., the assuredness afforded by the verifiability) has a negative effect on privacy: the larger α , the smaller the anonymity group. Consequently, PoS achieves the best privacy result for $\alpha = 1$, and the smallest amount of privacy is achieved for $\alpha = 5$ – in this case, $|AG_{PoS}| = \frac{n}{32}$.

Norwegian mix. The approach proposed by Puiggali et al. reduce the blocksize dependent on the number of used mix nodes. For m mix nodes a blocksize of $\sqrt[m]{n}$ is used. Thus, assuming that just one mixnode is honest the “anonymity group” has a size of

$$|AG_{Norway}| = \frac{n}{\sqrt[m]{n}}.$$

Random Block Verification. In RBV, each mix node shuffles twice. For verification, the ciphertexts are grouped into blocks of size \sqrt{n} . So, after the first shuffle, the size of the anonymity group is \sqrt{n} . However, for the second process, the blocks for the second shuffle are chosen such, that they include at least⁷ one ciphertext of each of the output blocks of the first shuffle. Therefore, to trace the ciphertext through the second shuffle, *all* input blocks need to be considered, which means in turn that all output blocks need to be considered. Hence, for one mix,

$$|AG_{rbv}| = n.$$

5.3 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 two components: proof generation by the mix net and verification by the verifier. We compute the computational costs only for one mix node. For re-encryption our approach, like RPC, needs twice as many exponentiation per mix node as the approach by Puiggali et al. and PoS. That is because re-encryption and shuffling are performed twice. But the impact of this is reduced as the mix nodes all process a subset of ciphertexts in parallel.

Randomized Partial Checking. 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 of the ciphertext and two times $\frac{n}{2}$ for β of the ciphertext are needed. In total the computational costs per mix node are

$$E_{rpc} = 2 \cdot 2 \cdot \frac{n}{2} = 2n.$$

Proof of Subproduct. The number of exponentiations during the PoS verification is $2\alpha(2m-1)[1]$ per mix node (for a total number of m mix nodes) and depends on the security parameter α (for $\alpha \leq 5$). Therefore the maximum number of exponentiations per mix node is $10(2m-1)$ which is achieved for $\alpha = 5$. Thus, efficiency also depends on the security parameter, and is given by

$$E_{PoS} = 2\alpha \cdot (2m - 1).$$

⁷ Since, in general, $\sqrt{n} \notin \mathbb{N}$, exactly one per block is not possible. However, our approach remains as close to that ideal as possible.

Norwegian mix. The verification process by Puiggalí et al. uses a zero knowledge proof to show the correctness of every block. The computational cost to verify the plaintext equivalence depends on the number of blocks. For n ciphertexts $\frac{n}{\sqrt[m]{n}}$ blocks are used. The calculation of the proof for each blocks requires 2 exponentiations and the verification of the correct mixing takes 4. Therefore the total number of exponentiations done by the mix net and the verifier are

$$E_{Norway} = 6 \frac{n}{\sqrt[m]{n}}.$$

Random Block Verification. Also the efficiency of our approach depends on the number of blocks. For n ciphertexts $m = \lfloor \sqrt{n} \rfloor$ blocks are used. During proof generation it takes 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 it takes the verifier two exponentiations per block to check the integrity of all blocks and thus $4m$ exponentiations for both verification steps. This leads to a total number of

$$E_{rbc} = 6 \frac{n}{\lfloor \sqrt{n} \rfloor}.$$

5.4 Conclusion

In Table 1, we summarise our findings. The ‘‘Fraud’’ row gives the chance of getting away with affecting the result with k votes (i.e., k changes for RPC, $k+1$ changes for the others). Privacy is expressed in terms of the anonymity group of one mix, and efficiency is expressed in terms of the number of exponentiations. The bold numbers are the top scores in each row.

	<i>RPC</i>	<i>PoS</i>	<i>Puiggalí et al.</i>	<i>RBV</i>
Fraud ($P(\text{undetected})$)	2^{-k}	$(\frac{5}{8})^\alpha$	$(\frac{\sqrt[n]{n}-1}{n-1})^k$	$(\frac{\sqrt{n}-1}{n-1})^k$
Privacy ($ AG $)	$\frac{n}{2}$	$\frac{n}{2^\alpha}$	$\frac{n}{\sqrt[m]{n}}$	n
Efficiency (# exp.)	$2n$	$2\alpha \cdot (2m - 1)$	$6 \frac{n}{\sqrt[m]{n}}$	$6 \frac{n}{\lfloor \sqrt{n} \rfloor}$

Table 1. Comparison (for n ciphertexts and m mix-nodes) of fraud detection (for one modified ciphertext), privacy and efficiency (for verification of one mix-node).

The table illustrates that RBV significantly improves privacy and efficiency over Puiggalí et al., at the cost of a slightly reduced ability to detect fraud. To get a feeling for how serious this reduction in fraud detection is, consider the following example. Suppose 3 ciphertexts are changed in a set of 1000 votes. The chance of not detecting this is less than $(\sqrt{1000})^{-2} \approx 0.1\%$.

6 Conclusion and Future Work

We discussed the mix net verification scheme by Puiggali 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 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.

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 Verificatum⁸. We also plan to discuss which probabilities satisfy legal requirements with legal scientists.

Acknowledgements. 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) and conducted in cooperation of provet (Project Group Constitutionally Compatible Technology Design at the University of Kassel) and CASED (Center for Advanced Security Research Darmstadt).

References

1. Boneh, D., Golle, P.: Almost entirely correct mixing with applications to voting. In: Proc. CCS'02. pp. 68–77. ACM (2002)
2. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM 24(2), 84–88 (1981)
3. Chaum, D., Pedersen, T.: Wallet databases with observers. In: Brickell, E. (ed.) CRYPTO'92, LNCS, vol. 740, pp. 89–105. Springer Verlag (1993)
4. 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>

⁸ <http://www.verificatum.com/>

5. 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)
6. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Advances in Cryptology – CRYPTO’86. LNCS, vol. 263, pp. 186–194. Springer (1986)
7. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Proc. CRYPTO ’86. pp. 186–194. Springer-Verlag, London, UK (1987), <http://portal.acm.org/citation.cfm?id=36664.36676>
8. Gjøsteen, K.: Analysis of an internet voting protocol. Cryptology ePrint Archive, Report 2010/380 (2010), <http://eprint.iacr.org/>
9. 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)
10. Groth, J.: A verifiable secret shuffle of homomorphic encryptions. vol. 23, pp. 546–579 (2010)
11. Jakobsson, M., Juels, A.: Millimix: Mixing in small batches. Tech. rep., Center for Discrete Mathematics & Theoretical Computer Science (1999)
12. Jakobsson, M., Juels, A., Rivest, R.L.: Making mix nets robust for electronic voting by randomized partial checking. In: Proceedings of USENIX’02 (2002)
13. 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)
14. Puiggali Allepuz, J., Guasch Castelló, S.: Universally verifiable efficient re-encryption mixnet. In: Proc. EVOTE 2010. LNI, vol. P-167, pp. 241–254. GI (2010)
15. 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)
16. Schnorr, C.P.: Efficient signature generation by smart cards. Journal of Cryptology 4, 161–174 (1991), <http://dx.doi.org/10.1007/BF00196725>
17. Terelius, B., Wikström, D.: Proofs of restricted shuffles. In: AFRICACRYPT. LNCS, vol. 6055, pp. 100–113 (2010)
18. Wikström, D.: Five practical attacks for ”optimistic mixing for exit-polls”. In: Selected Areas in Cryptography. pp. 160–175 (2003)
19. 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)

A Random distribution of ciphertexts over blocks using Fiat-Shamir

This section details how to arrive at a random distribution of ciphertexts over blocks. Consider a setting with m mixes and n input ciphertexts, and thus with $l = \sqrt{n}$ blocks, identified as $i \in \{0, \dots, l - 1\}$. Of these, $r = n - l \cdot l$ should have $l + 1$ elements, and the others are to end up with l elements. To ensure the initial assignment of ciphertexts to blocks is random, the first mix takes a hash of its input (by concatenating all ciphertexts), and uses the resulting number as seed of a random number generator. The stream of random bits from the generator is chopped into parts of size $s = \lceil \log_2 l \rceil$. Then, the first ciphertext is assigned to the block with the number given by the first part. Should this be a number

$> l$, this part is dropped. The second ciphertext is assigned the block identified by the second part, and so on.

In case a part identifies a number for which there is no corresponding block, the part is dropped. When a block is full, its index number is dropped. Initially, blocks are considered full when they have $l + 1$ elements. As soon as r blocks have been filled, blocks are considered full (and their indexes dropped) when they have l elements.

To speed up the assignment, the available blocks can be reindexed and s can be updated to limit the number of parts for which there is no corresponding block.