Universally Verifiable Efficient Re-encryption Mixnet

Jordi Puiggalí Allepuz and Sandra Guasch Castelló

Scytl Secure Electronic Voting
Tuset 20, 1-7, 08006 Barcelona, Spain
jordi.puiggali@scytl.com, sandra.guasch@scytl.com

Abstract: Implementing a transparent audit process when an election is conducted by electronic means is of paramount importance. Universally verifiable mixnets are focused on providing such a property by means of cryptographic proofs verifiable by any auditor. While some of these systems require high amount of computing resources that make them inefficient for real elections, others proposals reduce the computation cost by sacrificing audit accuracy or reducing the voter privacy protection level. In this paper, we propose an efficient mixnet verification system that combines the advantages of the RPC and Optimistic Mixing techniques, achieving a high audit accuracy level while fully preserving voters' privacy.

1 Introduction

When developing an election by electronic means, the main problem that arises is how to implement a transparent audit process. In traditional elections, independent auditors and observers can directly oversee the election process while it is happening. An important objective of this audit process is to verify that the opening of the ballot boxes and the counting of the votes is accurately and honestly implemented. When the counting process is done by electronic means (i.e., decryption and counting of the votes), overseeing the logical process while it is executed in the machine is practically impossible: this process is a logical entity that cannot be monitored by human means as in traditional elections. Therefore, it is of paramount importance that the electronic voting system provides transparent audit means of its correct behavior.

With electronic voting, results can be verified the same way as in traditional voting: making a parallel recount of the votes. Therefore, the difficulty of the audit process relies on the proper opening of the votes: the vote decryption process.

One possible approach is to allow auditors or observers to install programs in the system to monitor the voting platform. The problem is that auditor programs should be also monitored, since the decryption process becomes also vulnerable to these programs. Therefore, the solution introduces an infinite loop that has no easy solution (who watches the watchmen?).

Alternatively, the decryption process can be audited by means of monitoring the log information generated during its execution. However, assuming that the decryption process is compromised, the log information could be also manipulated to hide any

malicious practice. Furthermore, the information provided by the decryption process should be limited, since it must preserve voter's privacy (e.g., it cannot register the relationship between a decrypted content and an encrypted vote if the later can be correlated to a voter).

In 1995, Sako and Kilian [SK95] introduced the concept of "universal verifiability" for their proposal of a vote decryption process based on a mixnet approach. This verifiability is focused on providing means for any auditor or observer to verify the correct decryption of the votes, using cryptographic proofs that are generated by the decryption process.

A mixnet or *mix network* is composed of one or various nodes that shuffle the input messages using a secret permutation. Since mix-nodes also perform a transformation process that modifies the values of the set of input encrypted votes, it is important to be able to verify the mixing and decryption procedures in such a way that privacy and integrity are preserved.

Since Chaum introduced the first mixnet in 1981 [Ch81], the search for efficient verification methods that do not break the anonymization process (i.e., revealing the secret permutation or the re-encryption factors) has been a fertile area of research. Specifically, the universal verifiability property has been the main purpose of the mixnets designed in the last fifteen years.

In this paper, we introduce a universally verifiable efficient verification method for re-encryption mixnets that achieves high correctness while preserving voters' privacy. The paper is structured as follows: in section 2 we explain our motivation to design a new mixing verification system, in section 3 the underlying cryptosystem is defined, the new verification method is presented in section 4, and the paper concludes in section 5.

2 Motivation

Providing cryptographic proofs for the universal verification of a mixing process can be complex, computationally costly, and can involve a risk of reducing the voters' privacy.

Some mixing systems ([SK95], [FS01], [Ne01]) achieve a high correctness while preserving voters' privacy at the cost of performing a great number of proofs and verifications. Since these proofs and verifications have a high computational cost, it makes them inadequate in real election environments with a large number of votes. One of the motivations for the introduction of electronic voting is to speed up the vote counting process. For this reason, there are proposals that use them to make a parallel tallying of the votes while a faster method (less accurate) is used to give faster provisional election results, as proposed in [BG02].

To improve the efficiency of the mixing process (i.e., increase the speed of the mixing and audit process), other mixing systems focused the design of their audit mechanisms on reducing the cost of their cryptographic audit mechanisms by sacrificing to some

degree the strength of the voter's privacy or reducing the accuracy of the audit process (i.e., correctness) to an acceptable level. For instance, Random Partial Checking (RPC) [JJR02] trades-off mainly privacy, while the proposal in [Go02] preserves voters' privacy, but at the expense of sacrificing some correctness and efficiency: it performs more proofs that slow down the audit process. Another method that sacrifices some privacy and correctness on behalf of efficiency is [BG02], achieving results that can be considered good enough for an electronic process when large amounts of votes are counted.

The mixing verification system presented in this paper has a high degree of efficiency (comparable to the fastest proposals) while completely preserves voter privacy, and at the same time achieves a high level of correctness for small-medium and large elections.

3 Underlying Cryptosystem

In our scheme, voters use the ElGamal cryptosystem properly parameterized for semantic security [Pf94], [TY98] to encrypt the votes. The cryptosystem is composed by three public parameters: p, q, g, a public key h, and a private key x defined in the following way:

- The modulo p is chosen as a large safe prime, that is $p=2q+1$ and q is a prime number.
- g is a generator of Gq, the q-order subgroup of Zp^*.
- The private key x is selected from Zq, and the public key h is calculated as $h=g^x \bmod p$.

In order to make the encrypted votes indistinguishable, the voting options v are configured to be all from the quadratic residue or quadratic non-residue modulo p set. In case a voting option does not fit in the set, a padding string could be added.

The voting options are encrypted using random exponents r in Zq:

$$c = (v \cdot h^r \bmod p,\ g^r \bmod p) = (c_1, c_2)$$

Therefore, an encrypted voting option can be recovered as

$$v = c_1 \cdot c_2^{-x} \bmod p.$$

There are some interesting properties of the cryptosystem that are used in our mixing verification process, such as re-encryption and homomorphic operation of the encrypted votes.

3.1 Re-encryption of the encrypted votes

Thanks to the properties of the ElGamal cryptosystem, an encrypted message can be re-encrypted using a new randomization value without changing the decryption process.

Being the encrypted vote
$c = (v \cdot h^r \bmod p, g^r \bmod p) = (c_1, c_2)$,

The re-encryption can be performed as
$c' = (c_1 \cdot h^{r'} \bmod p, c_2 \cdot g^{r'} \bmod p) = (v \cdot h^{r+r'} \bmod p, g^{r+r'} \bmod p) = (c_1', c_2')$.

The re-encrypted vote can be decrypted as usual:
$v = c_1' \cdot c_2'^{-x} \bmod p$.

3.2 Homomorphic operation of the encrypted votes

Being two votes v_1 and v_2, an encryption operation E, and two algebraic operations Φ and Θ, the homomorphic property can be defined as
$E(v_1)\ \Phi\ E(v_2) = E(v_1\ \Theta\ v_2)$.

Since ElGamal is a cryptosystem with homomorphic properties, the product of n encrypted votes c_i generates an equivalent encrypted information Ec whose content Ev is the product of the plaintext voting options and the encryption exponent r_e is the sum of the individual encryption exponents:

$$\prod_{i=1}^{n} c_i = (\prod_{i=1}^{n} v_i \cdot h^{r_i}, \prod_{i=1}^{n} g^{r_i}) = ((\prod_{i=1}^{n} v_i) \cdot h^{\sum_{i=1}^{n} r_i}, g^{\sum_{i=1}^{n} r_i}) =$$
$$= (Ev \cdot h^{r_e}, g^{r_e}) = Ec$$

4 Mixing process and verification

4.1 Overview

The universal verification method for re-encryption mixnets presented in this paper combines the advantages of the RPC technique [JJR02] and the "Optimistic Mixing" proposal [Go02]: the partial disclosure of information is combined with proofs calculated from homomorphically aggregated groups of votes to achieve greater levels of privacy, robustness and soundness than these methods.

In the first step, each mix-node shuffles and re-encrypts the input encrypted votes, storing in a secret and secure way the permutation and re-encryption values applied for each vote. When the last node has mixed and re-encrypted its inputs the anonymized votes are ready to be decrypted, but before disclosing any significant information, the correct performance of the mixnet is universally verified.

In the verification process, the input encrypted votes of each node are divided into several independent groups following a random organization proposed by a verifier (i.e., an auditor). As said before, this group organization is done at the end of the mixing process (i.e., before decrypting the votes), preventing the disclosure of sensitive

information to any mixing node in order to cheat the verification process. Then each prover—the mix-node—provides information to the verifier about the global location in the mix-node's output of the votes belonging to each group in the input.

The global location of the votes of one output group does not disclose the individual position of each vote related to its original input group in the mix-node. For instance, disclosed output group positions are sorted by numerical value instead of their position in the mix-net input group.

When the verifier divides the input encrypted votes into groups, it also multiplies the votes in each group to obtain an *Input Integrity Proof* using the homomorphic properties explained in section 3.2. After the prover indicates which votes in the output of the node belong to each input group, the verifier can multiply the votes belonging to each output group to obtain an *Output Integrity Proof*. For each pair *Input-Output Integrity Proof* at each node, the prover provides a Zero-Knowledge Proof to demonstrate that the *Output Integrity Proof* is the re-encryption of the *Input Integrity Proof*.

Since the integrity proofs can be calculated and verified by any auditor, this method achieves the universal verifiability objectives. Furthermore, this proposal allows the verification of the mixing process without disclosing information about the position of individual votes in the output node after the shuffling process, preserving voters' privacy.

The next sections provide the details of vote group generation, the integrity proofs, and their related ZKPs.

4.2 Creating the groups

When the verification process starts, the verifier randomly defines how the input votes in the first mix-node are grouped by sending an array with the indexes of the position of the votes to be grouped:

For m input votes: $\{v_1, v_2, v_3, ..., v_m\}$.
An example of a grouping array is: $\{v_3, v_{m-1}, v_5, ..., v_2\}$.

Since the size of the groups is pre-defined (explained at the end of this section), the prover organizes the input votes following the grouping array order to define each vote group contents. Then, using the mixing permutation information, the prover indicates to the verifier for each mix-node output vote the group to which it belongs to. Since this information is provided following the order of the mix-node output votes, it is not possible to individually correlate input and output votes (only group affiliation).
For the next nodes of the mixnet, input vote groups are re-defined using as reference the output vote groups of the previous mix-node. We do not propose the reorganization of the groups at random, as in the first mix-node, to prevent disclosing information that could be used to correlate mixnet last output votes with first input ones: an attacker could analyze the votes belonging to each new grouping at each mix-node and identify

intersections of the groups that could facilitate the tracing of output votes with a reduced set of input votes (or in the worst case, an individual vote) of the fist mix-node. If so, the probability of an input vote being connected to a specific final output vote would be different from *1/m*, opening the door to privacy issues.

In order to prevent this attack, the new input groups are created by taking votes from different output groups of the previous mix-node. This is done in such a way that the groups in the last mix-node are composed of at least one vote from each group defined in the first mix-node. A proposal to redefine the groups consists of creating a new group by selecting votes belonging to different groups in the previous node in a consecutive way, like it is shown in figure 3. In this figure, the first group of the second node (G1,2) is formed by a vote from the first group of the first node (G1) and by one of the second (G2); the group of the second node (G3,4) is formed by a vote of the third group of the first node (G3) and one of the fourth (G4), and so on.

In order to preserve voter privacy, the size of the group also matters (e.g., if the size of the groups is too small, maybe the votes are not equally distributed at the last node of the mixnet). Furthermore, the probability of detecting manipulations of the votes during the mixing process also depends on the size of the groups (the smaller the group is, the higher the probability of detecting the manipulation of any vote is). For this reason, the groups need to be set up in a proper way to achieve the highest detection ratio without compromising voter privacy.

Being t the number of mixnet nodes (at least two) and m the total number of votes, the number of n votes inside a group should be at least:

$$n = \sqrt[t]{m} \qquad [1]$$

This formula preserves the privacy and optimizes manipulation detection rates of the votes. As shown in the formula, in our proposal the number of mixnet nodes also contributes to the correctness of the verification process. However, this optimization should be evaluated carefully, since the addition of new mixnet nodes reduces the efficiency of the proposal: increases the number of cryptographic operations required by the mixing and verification processes.

In the possible case of one or more nodes disclosing information about the individual permutations applied to the votes, they would not be taken into account in the formula 1. Therefore, privacy would still be maintained.

4.3 Generation of the ZKP of the Integrity Proofs

The integrity check of the votes grouped at each node is based in the homomorphic properties of the ElGamal encryption scheme. We call the result of multiplying a group of votes *Integrity Proof.*

The result of the multiplication of *n* votes of the same group in the input of a node, or *Input Integrity Proof* can be defined as:

$$\prod_{i=1}^{n} c_i = (\prod_{i=1}^{n} v_i \cdot h^{r_i}, \prod_{i=1}^{n} g^{r_i}) = ((\prod_{i=1}^{n} v_i) \cdot h^{\sum_{i=1}^{n} r_i}, g^{\sum_{i=1}^{n} r_i}) = (Ev \cdot h^{r_e}, g^{r_e})$$

The multiplication of the same group of votes in the output of the node (i.e., the same votes after being re-encrypted), is called *Output Integrity Proof* and it is equal to:

$$\prod_{i=1}^{n} c_i' = (\prod_{i=1}^{n} v_i \cdot h^{r_i + r_i'}, \prod_{i=1}^{n} g^{r_i + r_i'}) = ((\prod_{i=1}^{n} v_i) \cdot h^{\sum_{i=1}^{n} r_i + r_i'}, g^{\sum_{i=1}^{n} r_i + r_i'}) =$$
$$= (Ev \cdot h^{r_e + r_e'}, g^{r_e + r_e'})$$

Since the mix-node knows all the individual re-encryption factors of the votes of each group, it can calculate the accumulated re-encryption factor $r_e' = \sum_{i=1}^{n} r_i'$. Having this accumulated factor, the mix-node can make a Non-Interactive Zero Knowledge Proof of Re-encryption (NIZKP-RE), proving that the *Output Integrity Proof* is the re-encryption of the *Input Integrity Proof* using the re-encryption factor r_e' . This proof can be based on the Schnorr Identification Protocol like in [MA99] or the Chaum-Pedersen proof of equality of discrete logarithms [CP93].

Therefore, any auditor, after calculating by herself the *Input Integrity Proof* and *Output Integrity Proof* of the groups of a node, can use the NIZKP-RE to check that both proofs are based on the same contents. In other words, the global contents of the votes in a group still remain the same. Since the integrity proofs are based on the homomorphic product of the votes, there is still a possibility that a rogue mix-node could cheat the system. However, as explained in section 4.5.1, the way the groups are modeled in our proposal makes the probability of detecting such manipulation very high (e.g., it has a probability of 99.91% of detecting a manipulation of 2 votes in an election with 10,000 votes).

If the proof is successfully verified, the node is believed to behave correctly. This NIZKP-RE is done for each group of votes at each node.

4.4 Verification Protocol Summary

To summarize, the verification protocol implements the following steps after the mixing process:

1. For the first mix-node, the verifier divides at random the input votes in groups using a grouping array that is sent to the prover.
2. Then, the verifier calculates an *Input Integrity Proof* for each group.

3. The verifier asks the prover for the output destination of the votes belonging to each group and calculates an *Output Integrity Proof* for each group.
4. The prover calculates a NIZKP based on the re-encryption factor in order to demonstrate that the *Output Integrity Proof* is the re-encryption of the *Input Integrity Proof* of the same group.
5. For the next node, the groups are redefined in such a way that each new group is composed of votes from different output groups in the previous node, and the steps 2–5 are repeated until the correct behavior of the last mix-node is verified.

An example of the procedure is shown in Fig. 1. The figure also shows the group configuration at each mix-node:

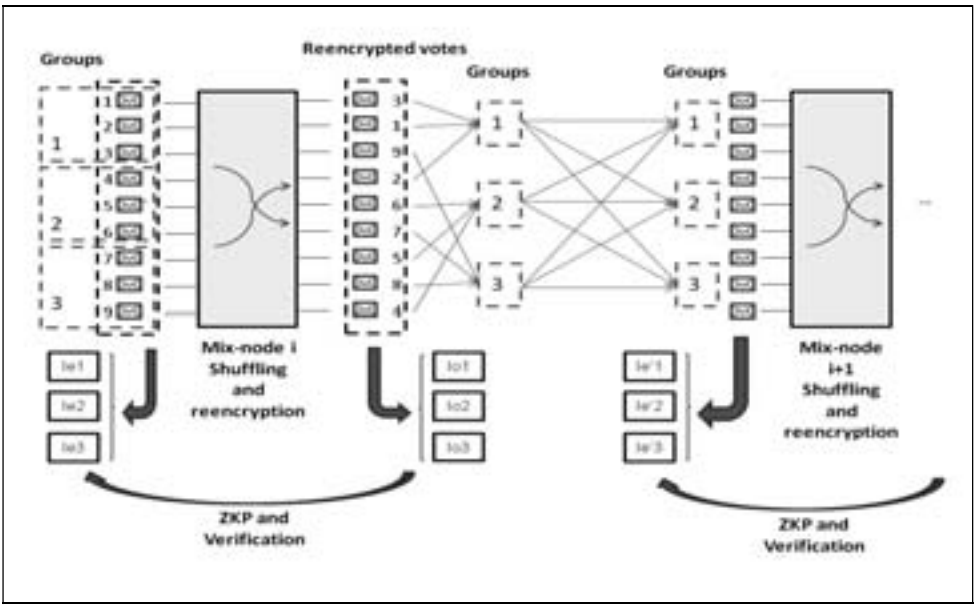

Fig. 1: Mixing verification process

4.5 Properties of the new system

We analyze the new verification method proposed from four points of view: soundness, efficiency, privacy, and universal verifiability.

4.5.1 Soundness

Since the verification process is based on the *Integrity Proofs* that are calculated by multiplying groups of votes, an attacker could take advantage of the cryptosystem's homomorphic properties in order to modify the votes in the mixnet without being detected. In fact, if several votes in the same group are modified in such a way that the

modifications are cancelled when the *Integrity Proof* is calculated, these changes are not detected in the verification process. However, since the group configuration is unknown until the mixing process finishes, the probability of an attacker changing a significant amount of votes without being detected is negligible.

The chance of an attacker not being detected depends on the amount of votes in the mixnet, the number of groups in which the votes are divided, and the number of manipulated votes. Since the probability of being undetected decreases with the number of modified votes, we can define the most successful scenario for the attacker as the one where only two votes are manipulated, they are in the same group, and the modifications cancel out when the *Integrity Proof* is calculated.

The probability of detecting a pair of manipulated votes is:

$$P_{success} = 1 - \frac{n-1}{m-1}, \qquad [2]$$

where m is the total number of votes and n is the number of votes in each group.

It is important to maintain a convenient relationship between the total number of votes processed by the mixnet and the size of the groups: the smaller the groups are, the higher the probability of detecting an attacker is. Otherwise, the larger the groups are, the faster the verification process becomes.

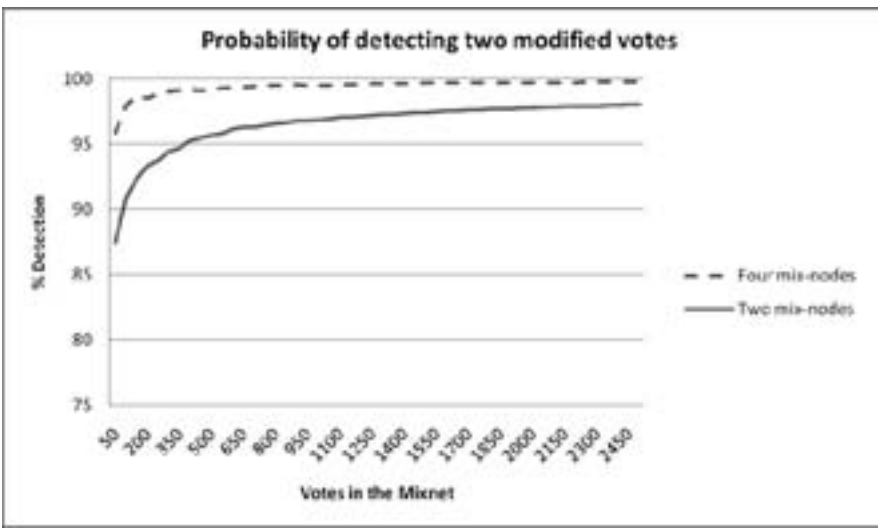

Fig. 2: Graphic showing the probability of detecting two modified votes when two mix-nodes and four mix-nodes are used.

Formula 1 gives an optimized relationship between the size of a group of votes and the total number of votes that are processed by the mixnet to meet the efficiency, soundness, and privacy requirements.

For example, in an election with 10,000 votes and a mixing of two nodes, the minimum size of the groups in order to preserve the voter privacy is 100 votes. With this configuration the probability of detection of two modified votes is 99%. If the mixing is performed with four nodes, the minimum size for each group is ten votes, which gives a probability of detection of 99.91%.

The probability of detecting two modified votes in a mixnet composed of two mix-nodes (bigger groups) or of four mix-nodes (smaller groups) is shown in Fig. 2. In both cases the probability of detection tends toward 100%, but when more mix-nodes are used and smaller groups are configured, the probability of detection increases faster.

4.5.2 Privacy

Following the procedure described in section 4.2., groups at the input of each node contain votes from different groups of the previous node's output, in such a way that it is impossible for an attacker to track back the output votes to the groups defined in the first mix-node. Therefore, the privacy level of the verification method does not compromise the original privacy provided by the re-encryption mixnet.

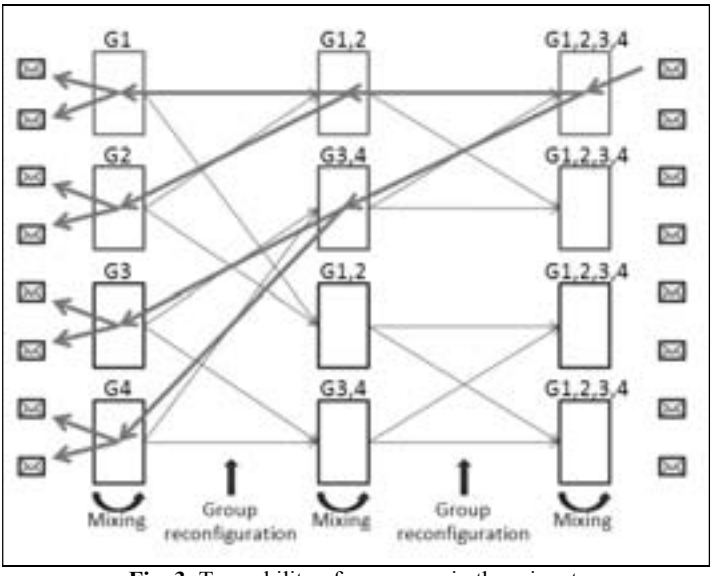

Fig. 3: Traceability of a message in the mixnet.

Fig. 3 shows how privacy is maintained due to the group reconfiguration at each node. An attacker choosing any encrypted vote of the mixnet output cannot successfully track it back to an individual encrypted vote in the input or any subset of input votes.

Therefore, all the votes in the input have the same probability of being in a specific output.

Formula 1 defines the group size depending on the number of mix-nodes for a fixed amount of votes in the mixnet. In the case that it is desirable to use small groups to increase the probability of detecting manipulated votes (the soundness of the proofs), more nodes in the mixnet are needed to preserve voter privacy.

4.5.3 Efficiency

Preserving voters' privacy and audit soundness by dividing the votes into small non-overlapping groups has an odd behavior: it reduces the efficiency of the mixnet. The computation costs of the verification method depend on the number of votes in the system and the amount of groups created for the verification process, since the proofs of correct behavior are done over them. Therefore, for a fixed number of votes in the mixnet, the more groups there are, the more the computation costs are consumed. On the other hand, the probability of detecting manipulated votes increases since there are less votes in each group.

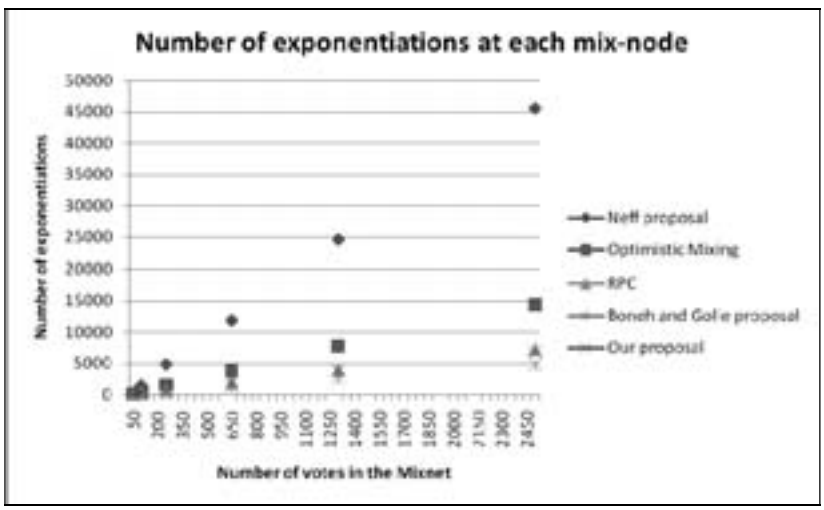

Fig. 4: Comparison of the number of exponentiations required at each mix-node in some mixing verification systems

We have estimated the cost of performance of our method based on the number of exponentiations done at each phase:

- Mixing: the re-encryption of the votes at each mix-node requires $2m$ exponentiations, where m is the total number of votes in the mixing.
- Proof of correct mixing: calculating the zero-knowledge proofs of correct performance at each node requires $2(m/n)$ exponentiations, where (m/n) is the

number of groups in which the total number of votes is divided at each node, and *n* is the number of votes per group.
- Verifying correct mixing: the verification of correct mixing at each node requires 4*(m/n)* exponentiations.

In the Fig. 4, a comparison of our method with other mixing verification systems in terms of the number of exponentiations is provided, showing that our system is one of the fastest for large amounts of votes.

4.5.4 Universal verifiability

A universally verifiable mixnet provides a proof of correct mixing that any observer can verify. For this purpose, some information is stored to let any auditor check the verification process after the mixing. Since the verification is made in zero knowledge, there is no need for the auditor to have any special or private data (i.e., private key of the election) to perform this check. The information collected during the mixing process for further verification consists of the set of encrypted votes in the mixing input and the re-encrypted votes at the output of each mix-node. During the verification process, the configuration of the votes in (input/output) groups and the zero-knowledge proofs performed by each node are also stored. Therefore, any auditor can check the verification process later using this information: the *Input* and *Output Integrity Proofs* can be calculated from the input/output sets at each node and the zero-knowledge proofs between them can be verified.

5 Conclusions

In this paper we described a new proposal of a universally verifiable and efficient method for re-encryption mixnets that achieves high correctness while preserving voters' privacy. Specifically, our proposal achieves an efficiency level comparable to the current faster existing systems, while our capacity of detecting manipulated votes is closer to the most accurate methods without compromising the voters' privacy.

Assuming an implementation of four nodes and setting the vote group size of the verification process to optimize the relationship between full voter privacy, efficiency, and fraud detection (using the formula 1 described in section 4.2), we can achieve the following conclusions.

From the point of view of efficiency, the computation cost of our proposal is close to the Boneh and Golle method [BG02]: the fastest one as shown in the figure 4. Regarding RPC method [JJR02], this is more efficient only for small batches of votes (less than 1500), but when the amount of votes increases, our system becomes faster. Considering the other methods [Go02][Ne01], the efficiency improvements are clear.

In terms of privacy, compared with our proposal, the original RPC proposal offers a weaker privacy level, since the input votes could be connected with some specific output

votes with a probability higher than *1/m*. An improvement proposed by Chaum [Ch02] solves this privacy issue by grouping pairs of mix-nodes in a special way during the verification and requiring at least four nodes. However, the problem still remains if the information from intermediate nodes is disclosed. On the other hand, in the method explained in [BG02], full voter privacy is difficult to achieve: each verification round done to increase the accuracy of the verification process discloses sensitive information that could be used to increase the probability of correlating input and output votes. In our proposal, we keep full voter privacy.

In terms of accuracy, our proposal achieves a high level of cheating detection for a small number of manipulated votes (i.e., 2 votes). This probability is closer to 100% when the number of votes is near 300 votes (99%). The other methods, except [Ne01], have similar or lower accuracy levels.

In summary, compared with the current verification methods, our solution is the most well-balanced in terms of efficiency, privacy, and accuracy, while providing universal verification properties.

Bibliography

[BG02] Boneh, D., and P. Golle. 2002. Almost entirely correct mixing with applications to voting. In *Proceedings of the 9th ACM conference on computer and communications security (Washington, DC, USA, November 18–22, 2002) CCS '02*, ed. V. Atluri, 68-77. New York NY: ACM.

[Ch02] Chaum, D. Secret ballot receipts and transparent integrity. Better and less-costly electronic voting at polling places.White Paper. http://www.vreceipt.com/article.pdf/.

[Ch81] Chaum, D. L. 1981. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM 24*, 2 (February): 84-90.

[CP93] Chaum, D., and T. P. Pedersen. 1993. Wallet databases with observers. In *Proceedings of the 12th annual international cryptology conference on advances in cryptology (August 16–20, 1992). Lecture notes in computer science, vol. 740*,ed. E. F. Brickell, 89–105. London: Springer-Verlag.

[FS01] Furukawa, J., and K. Sako. 2001. An efficient scheme for proving a shuffle. In *Proceedings of the 21st annual international cryptology conference on advances in cryptology (August 19 - 23, 2001). Lecture notes in computer science, vol. 2139*. ed. J. Kilian, 368–387. London: Springer-Verlag.

[Go02] Golle, P., S. Zhong, D. Boneh, M. Jakobsson, and A. Juels. 2002. Optimistic mixing for exit-polls. In *Proceedings of the 8th international conference on the theory and application of cryptology and information Security. Advances in cryptology (December 01 - 05, 2002). Lecture notes in computer science, vol. 2501,* ed. Y. Zheng, 451–465. London: Springer-Verlag.

[JJR02] Jakobsson, M., A. Juels, and R. L. Rivest. 2002. Making mix nets robust for electronic voting by randomized partial checking. In *Proceedings of the 11th USENIX security symposium (August 05–09, 2002). USENIX Security Symposium,* ed. D. Boneh, 339–353. Berkeley CA, USA: USENIX Association.

[MA99] Markus, J., and J. Ari. 1999 *Millimix. Mixing in small batches*. Technical Report 99-33. Center for Discrete Mathematics & Theoretical Computer Science.

[Ne01] Neff, C. A. 2001. A verifiable secret shuffle and its application to e-voting. In *Proceedings of the 8th ACM conference on computer and communications security (Philadelphia, PA, USA, November 05 - 08, 2001). CCS '01*, ed. P. Samarati, 116–125. New York, NY: ACM.

[Pf94] Pfitzmann, B. 1994. Breaking efficient anonymous channel. In *Advances in cryptology (Eurocrypt '94), volume 950 of LNCS*, ed. A. D. Santis, 332–340. Berlin: Springer-Verlag.

[SK95] Sako, K., and J. Kilian. 1995. Receipt-free mix-type voting scheme. A practical solution to the implementation of a voting booth. In *Advances in cryptology - EUROCRYPT '95. Lecture notes in computer science*, ed. C. Guillou and J. Quisquater, 393-403. Berlin: Springer-Verlag.

[TY98] Tsiounis, Y. and M. Yung. 1998. On the security of ElGamal based encryption. In *Proceedings of the first international workshop on practice and theory in public key cryptography. Public key cryptography (February 05 - 06, 1998). Lecture notes In computer science, vol. 1431.* ed. H. Imai and Y. Zheng, 117–134. London: Springer-Verlag.