SAFE RPC
Auditing mixnets safely using Randomized Partial Checking

Stefan Popoveniuc, Eugen Leontie
Computer Science Department, The George Washington University, Washington D.C., U.S.A
poste,eugenl@gwu.edu

Keywords: mix net, randomized partial checking, anonymity, privacy sets, electronic voting.

Abstract: Secure voting systems like PunchScan and Scantegrity use mixnets which are verified using Randomized Partial Checking (RPC). This simple and efficient technique can lead to privacy loss and may, in an extreme case, result in linking all the clear text ballots to the voters who cast them, thus completely destroying the secrecy of all ballots and circumventing the functionality of the mixnet. We suggest a simple technique, Secure RPC (SRPC), that uses RPC in a way that guarantees maximal privacy in all possible cases. We prove that SRPC does not asymptotically reduce the integrity offered by RPC.
Acknowledgement: This work was made possible in part by grants NSF CNS-0831149, NSF CNS-0934725l and AFOSR FA9550-09-1-0194.

1 INTRODUCTION

David Chaum introduced mix networks (mixnets) (Chaum, 1981), as one of the first constructions for protecting privacy in the digital world. Since then, mixnets have been a fertile ground for research in anonymous communications and applications. Mixnets provide a foundation for schemes in which privacy is of paramount importance, such as anonymous message delivery and electronic voting.

The role of a mix is to take a set of messages, or inputs, and (1) preserve the information in the messages while (2) shuffling their order to remove any *links* (correspondences between inputs and outputs). Given the set of inputs and a fixed output, an adversary should not identify the particular input corresponding to the given output, with a probability greater than a uniform random guess. To mitigate against the corruption of single mix, a mixnet, or mix cascade has been proposed. More mixes in a sequence improve the quality of the mixnet, but reduce its performance.

In verifiable electronic voting systems, a mixnet is used to de-correlate votes from voters; the inputs of the mixnet are encrypted ballots (possible likable to voters) and the outputs are the plaintext ballots (not linkable to voters). Recently, specialized mixnets have been proposed, such as Punchscanian mixnets (Popoveniuc and Hosp, 2006) and pointer-based mixnets (Chaum et al., 2008).

Checking the correctness of a mixnet means verifying that the mix did not modify, delete, or inject messages. Two auditing methods are common: zero knowledge proofs (ZKPs, e.g. (Neff, 2001)), and randomized partial checking (RPC (Jakobsson et al., 2002)). Both methods are based on a challenge response mechanism. ZKPs require the mix to produce new information based on challenges, whereas RPC utilizes a simple observation: for each mix, some links can be revealed, as long as there is no full path of revealed links for the entire mixnet. RPC may be more efficient than ZKPs, and many of the voting systems recently proposed use RPC.

1.1 Motivation

We explore the possibility of finding the correspondence between the input and the output for a pair of mixes that is audited using RPC. We find that the privacy may be significantly reduced, much more than was observed by the original RPC paper (Jakobsson et al., 2002). Many links may be completely revealed.

The primary motivation for this work is the ex-

istence of the specialized mixnets used by PunchScan (Popoveniuc and Hosp, 2006) and Scantegrity (Chaum et al., 2008), which have some unique characteristics: (1) the number of unique messages in the output of the last mix is very small (because the outputs represent candidates) (2) the mixnet has only two mixes and (3) the pair of mixes is audited using RPC. However, this work is general in scope and applies to any mixnet that satisfies the above properties, regardless if the mixnet is used for voting or not.

Our work does not apply to mixnets in which all the outputs of each mix are unique, or there is a very large number of such unique outputs. Also our observations do not apply if the number of mixes is larger than two. While having four mixes would solve the observed problem, this would pose a significant performance penalty, essentially doubling the amount of time it takes to obtain the final final tally. Safe RPC shows a constructive way to reduce the probability of breaking the privacy because of the RPC to zero.

1.2 Contributions

The contributions of this work are: (1) to raise the issue that RPC may reduce the ideal privacy sets, and may expose the associations between the outputs of a mixnet and its inputs, thus defeating the very purpose of the mix, and (2) to proposes a simple and provable solution that prevents this situation from appearing, while not diluting the integrity assurance.

Our result primarily impacts designers of electronic voting systems, a number of which are based on mixnet. To exemplify, assume that the results of an election say that Alice got 52% of the votes and Bob got 48%. In the case of classical RPC, the set of votes that produced the final tally gets split into two subsets, resulting in two partial tallies. While unlikely, it may happen that one of the partial tallies contains only votes for Alice. In this extreme case, the privacy of half of the votes becomes totally compromised.

Our approach is simple: if two consecutive mixes need to be audited for correctness, instead of doing the random choices on the output of the first mix (and thus the input to the second mix), we do the random choices on the output of the second mix, such that the resulting subsets maintain the characteristics of the entire output. We audit the second mix using one of these sets and the first mix using the complement of the pre-images of this set.

In the example above, the final tally is divided into two sub-tallies, each having 52% of the votes for Alice and 48% of the votes for Bob. This is not a completely random partitioning, but rather an educated split of the tally that maintains maximal privacy.

2 RELATED WORK

Much of the previous work that addresses privacy leakage caused by mixnet auditing focuses on the relationship among multiple consecutive mixes. In the original RPC, two mixes can be paired so that the revealed outputs of the first mix are not the revealed inputs of the second mix.

Chaum (Chaum, 2004) uses RPC across four consecutive mixes, with the first two used as before, but with the third mix revealing inputs corresponding to half the outputs of the second mix, and with the fourth mix revealing only the unrevealed outputs of the third. This addresses the problem of RPC cutting the privacy set in half.

Gomulkiewicz et al. (Gomulkiewicz et al., 2003) provide formal analysis of the information loss induced by Chaum's scheme with respect to the probability distribution of an input being linked to an output. They find that the connection between the inputs and outputs of a mixnet is sufficiently random (assuming a good shuffle at each mix) to have assurance that the mixnet meets the privacy demands of voting systems.

While there are other ways to audit mixnets besides RPC, this work is solely focused on this checking technique.

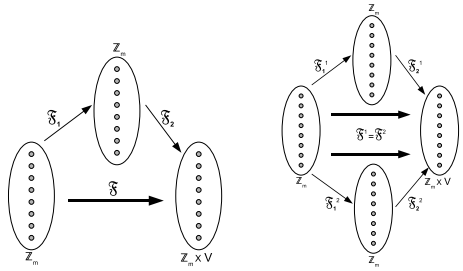
3 USEFUL DEFINITIONS

We model two sequenced mixes by a function $\mathfrak{F} : \mathbb{Z}_m \to \mathbb{Z}_m \times \mathcal{V}$ where m is the the number of inputs to the mixnet, $\mathbb{Z}_m$ is the set of numbers from zero to $m-1$, and $\mathcal{V}$ is the set of clear text messages produced by the mixnet (e.g. votes). Let n be the cardinal of $\mathcal{V}$. In a typical voting system, n represents the number of candidates and is between 2 and 10, often times much closer to 2 than to 10.

The mixnet $\mathfrak{F}$ consists of two mixes $\mathfrak{F}_1$ and $\mathfrak{F}_2$, $\mathfrak{F} = \mathfrak{F}_2 \circ \mathfrak{F}_1$, where $\mathfrak{F}_1 : \mathbb{Z}_m \to \mathbb{Z}_m$ and $\mathfrak{F}_2 : \mathbb{Z}_m \to \mathbb{Z}_m \times \mathcal{V}$. Figure 1(a) portraits the setting.

The current RPC method to audit the mixnet is: an independent auditor flips an unbiased coin for each output o_1 of $\mathfrak{F}_1$. If the coin is heads, the mixnet reveals the pre-image i of o_1 through $\mathfrak{F}_1$ and all the data that allows the public to check that $\mathfrak{F}_1(i) = o_1$. If the coin is tails, the mixnet reveals the post image o of o_1 through $\mathfrak{F}_2$ along with all the data needed to check that $\mathfrak{F}_2(o_1) = o$.

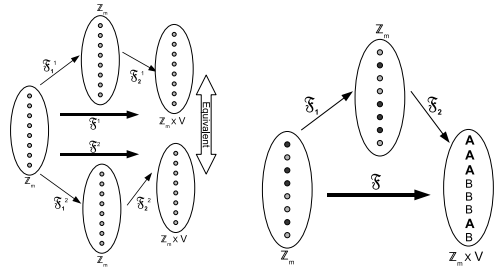
We assume that all the mappings done by $\mathfrak{F}_1$ and $\mathfrak{F}_2$ are equally likely. Because no transformation is revealed from the input of $\mathfrak{F}$ to its output, no outside observer knows the correlation between the input and



(a) Two mixes forming a mixnet

(b) Linked $\mathfrak{F}$s.

Figure 1



(a) Unlinked $\mathfrak{F}$s.

(b) A situation that can occur under RPC

Figure 2

the permuted output. The probability that the mixnet cheats on k ballots and is not detected is $\frac{1}{2^k}$.

3.1 Privacy leakage

We now define privacy leakage. We start by giving a small example. Lets assume we have an election with 2 candidates, and in the final tally each candidate got exactly 50% of the votes. Looking at the final tally, any voter is equally likely to have voted for any of the two candidates. RPC divides the final tally into two partial tallies. We assume one of the partial tallies has 60% of the votes for one candidate and 40% for the other candidate. A voter belonging to the first partial tally is now more likely to have voted for the candidate that got 60%, whereas in the initial case the voter was equally likely to have voted for either of the candidates. We say that privacy has been breached.

Lets assume we have a winner-takes-all election and the final tally is $\{p_1\%, p_2\%, ..., p_n\%\}$. We say that privacy has been breached if there exists a partial tally $\{p_1^{'}\%, p_2^{'}\%, ..., p_n^{'}\%\}$, such that $\exists x \in \{1, ..., n\}$ and $\exists \varepsilon > 0$, such that $|\frac{p_x - p_x^{'}}{p_x}| \geq \varepsilon$. In other words, the percentages from the final tally are different from the percentages from the partial tallies. The larger ε is, the larger the privacy leakage.

In the example above, $|\frac{50\% - 40\%}{50\%}| = 20\%$, thus choosing $\varepsilon = 0.20$ suffices to prove that there is privacy leakage.

An interesting case is when one can prove how a voter did not vote. For example, in a contest with three candidates, it may be possible to prove that a voter did not vote for any of the three candidates ($\exists x \in \{1, ..., n\}$ such that $p_x^{'} = 0$, or, equivalent $\varepsilon = 1$).

In an extreme case, if $\exists x \in \{1, ..., n\}$ such that $p_x^{'} = 1$, it can be proven how one or more voters voted. This implies that all other $p_y^{'} = 0$, $\forall x \neq y$.

3.2 Linked $\mathfrak{F}$s

It may be that there is more than one function $\mathfrak{F}$ that links the same inputs to the same outputs. We call this technique linked $\mathfrak{F}$s.

Let $\mathfrak{F}^i : \mathbb{Z}_m \rightarrow \mathbb{Z}_m \times \mathcal{V}$ be a family of functions (with f members), such that $\mathfrak{F}^i(x) = \mathfrak{F}^j(x)$, $\forall i, j, x \in \mathbb{Z}_f$. Each function $\mathfrak{F}^i$ is a composition of two functions $\mathfrak{F}^i = \mathfrak{F}^i_2 \circ \mathfrak{F}^i_1$ (see figure 1(b)). It is not necessary that $\mathfrak{F}^i_1 = \mathfrak{F}^j_1$ or $\mathfrak{F}^i_2 = \mathfrak{F}^j_2$, for $\forall i \neq j$; the two can be different or the same, as long as $\mathfrak{F}^i_2 \circ \mathfrak{F}^i_1 = \mathfrak{F}^j_2 \circ \mathfrak{F}^j_1$.

This construction can be used to increase the integrity assurance during the audit phase, or to recover from a failed audit. The probability of cheating on k ballots without being detected is $\frac{1}{2^{k \times f}}$.

3.3 Unlinked $\mathfrak{F}$s

If we relax the previous requirement $\mathfrak{F}^i(x) = \mathfrak{F}^j(x)$, $\forall i, j, \forall x \in \mathbb{Z}_m$ to a requirement that, when fixing any element in $\mathcal{V}$, the number of such elements in $\mathfrak{F}^i(I)$ is equal to the one in $\mathfrak{F}^j(I)$, $\forall i, j$ we obtain a different flavor of mixnet, that we call unlinked $\mathfrak{F}$s. Intuitively, this transforms the given input into a set of output messages that are all equivalent (when considering the messages in $\mathcal{V}$) but the output messages are not necessarily associated with the same output indexes.

The tally is the same, but each $\mathfrak{F}$ provides a different order of the votes. Figure 2(a) has an example. As in the previous case, the probability of cheating on k ballots and not being detected is one in $2^{k \times f}$

4 IDENTIFYING THE PROBLEM

The problem we have identified is caused by the small number of unique messages that are transmitted via the mixnet. In the case of a voting system, the number

of messages is usually under 10, each one corresponding to a candidate running in a given race. But the number of total non-unique messages is very large, equal to the number of cast ballots.

It may happen that the RPC splits the output set into two subsets, in a way such that all the messages that are equal to one another (say to A) end up in the same subset. To better portrait the problem we give some example bellow.

Assume $m = 8$ (eight votes) and $n = 2$ (two candidates, *A* and *B*). Assume the output of $\mathfrak{F}$ is the set $\mathbb{O}$ ={(0,A), (1,A), (2,A), (3,B), (4,B), (5,B), (6,A), (7,B)}. Figure 2(b) describes the setup.

For our small example, the chance of breaching privacy is reasonably high if RPC is used. In practice, the output of a mixnet will have a large number of ballots and the chance that an unfortunate partitioning is performed by the auditors drops exponentially. However, this probability never reaches zero.

Let's consider a single function $\mathfrak{F}$. Let $\mathfrak{F}_1 : \mathbb{Z}_8 \rightarrow \mathbb{P}$ where $\mathbb{P} = \mathbb{Z}_8$ (*P* stands for partially decrypted), and assume the coin flips divided the set $\mathbb{P}$ into $\mathbb{P}_{left} = \{0,2,3,7\}$ and $\mathbb{P}_{right} = \{1,4,5,6\}$. While RPC reveals the actual one to one mappings, for this exemplification we are only interested in the overall sets. Assume that the pre-image of the set $\mathbb{P}_{left}$ is $I_{left} = \mathfrak{F}_1^{-1}(\mathbb{P}_{left}) = \{1,4,5,7\}$ and the post-image of the set $\mathbb{P}_{right}$ is $\mathbb{O}_{right} = \mathfrak{F}_2(\mathbb{P}_{right})$ ={(0,A), (1,A), (2,A), (6,A)} (see Figure 2(b)).

Because $\mathfrak{F}_1$ is a bijection and $\mathfrak{F}_2$ is one to one, it can be easily inferred that $I_{right} = \{0,2,3,6\}$ and $\mathfrak{F}(I_{right}) = \mathbb{O}_{right}$={$(0,A),(1,A),(2,A),(6,A)$} and thus that all the inputs $\{0,2,3,6\}$ correspond to votes for the same candidate, A. While no one knows to which particular element from $\mathbb{O}_{right}$ any element from I_{right} goes to, this is irrelevant, since all of them represent the same message (a vote for candidate *A*). All the other inputs, thus, correspond to candidate *B*.

4.1 Problems with linked $\mathfrak{F}$s

Assume we have the same output as in the previous case $\mathbb{O}$ ={(0,B,), (1,A), (2,B), (3,A), (4,B), (5,B), (6,A), (7,A)}. In the case of linked $\mathfrak{F}$s the output is the same for any $\mathfrak{F}^i$. Assume we have two linked $\mathfrak{F}^i$s. Following the same audit procedure for each of the $\mathfrak{F}^i$s, assume we obtain I^1_{left} = {0, 1, 2, 3}, $\mathbb{O}^1_{left}$ = {(0,B), (1,A), (2,B), (5,B),}; I^1_{right} = {4, 5, 6, 7}, $\mathbb{O}^1_{right}$ = {(3,A), (4,B), (6,A), (7,A)}; I^2_{left} = {0, 1, 4, 5}, $\mathbb{O}^2_{left}$ = {(2,B), (3,A), (4,B), (5,B)}; I^2_{right} = {2, 3, 6, 7}, $\mathbb{O}^2_{right}$ = {(0,B), (1,A), (6,A), (7,A)}.

When analyzing only $\mathfrak{F}^1$, we can see that the inputs in I^1_{left} = {0, 1, 2, 3} are more likely to correspond to *B*s, because $\mathbb{O}^1_{left}$ = {(0,B), (1,A), (2,B), (5,B),} (a 75% chance as opposed to a 50% chance).

If we intersect I^1_{left} with I^2_{left} and $\mathbb{O}^1_{left}$ with $\mathbb{O}^2_{left}$, we can extract further information. $I^{12}_{left} = I^1_{left} \bigcap I^2_{left}$ = {0,1}; $\mathbb{O}^1_{left} \bigcap \mathbb{O}^2_{left}$ = {(2,B), (5,B)} and therefore we know that the inputs $\{0,1\}$ correspond to the same message, *B*. Applying the same logic $I^{12}_{right} = I^1_{right} \bigcap I^2_{right}$ = {6,7}; $\mathbb{O}^1_{right} \bigcap \mathbb{O}^2_{right}$ = {(6,A), (7,A)}, and thus we've found another two inputs that correspond to the same message, *A*.

4.2 Problems with unlinked $\mathfrak{F}$s

Assume we have the same output as in the previous case $\mathbb{O}$ = {(0,B,), (1,A), (2,B), (3,A), (4,B), (5,B), (6,A), (7,A)}. In the case of unlinked $\mathfrak{F}$s the messages carried by the output are the same (and in the same proportion), but the order of the messages is different for each $\mathfrak{F}^i$s. The main difference from the previous example is that we cannot intersect the outputs of $\mathfrak{F}^i$s, as the first element in the output pair may not represent the same output (each $\mathfrak{F}^i$ performs a different shuffle, but produces the same unordered set of messages). For simplification, we drop the first element of the output from our analysis and prove that there may be situations in which privacy is still lost.

Like in the previous case, assume we only have two $\mathfrak{F}^i$s. Following the same audit procedure for each of the $\mathfrak{F}^i$s, assume we obtain I^1_{left} = {0, 1, 2, 3}, $\mathbb{O}^1_{left}$ = {B, A, B, B,}; I^1_{right} = {4, 5, 6, 7}, $\mathbb{O}^1_{right}$ = {A, B, A, A}; I^2_{left} = {0, 1, 4, 5}, $\mathbb{O}^2_{left}$ = {B, A, B, B}; I^2_{right} = {2, 3, 6, 7}, $\mathbb{O}^2_{right}$ = {B, A, A, A}.

We compute $I^{12}_{left_right} = I^1_{left} \bigcap I^2_{right}$={2, 3} and run through the possible messages of these two inputs $\{2,3\}$. It cannot be that both have *A*s corresponding to them, since $\mathbb{O}^1_{left}$ does not contain two *A*s; similarly, it cannot be that both are *B*s, since $\mathbb{O}^2_{right}$ does not have two *B*s. So it must be that one is *A* and one is *B*. But if we remove one *A* and one *B* from $\mathbb{O}^1_{left}$ we get two *B*s, thus it must be that inputs $\{0,1\}$=$I^1_{left}/\{2,3\}$ both correspond to *B*s. Following the same logic, inputs $\{6,7\}$=$I^2_{right}/\ \{2,3\}$ both correspond to *A*s. Thus we have completely broken the privacy of four messages.

5 DESCRIBING SRPC

We present a technique, Safe RPC, that ensures there is no privacy leakage (as per definition from section 3) whenever possible. SRPC ensures that $\forall x \in \{1,2,...,n\}$ $p_x = p'_x$, and therefore $p_x - p'_x = 0$ re-

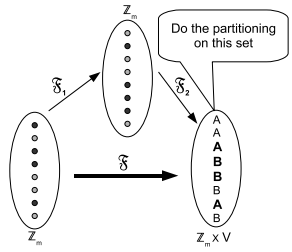


Figure 3: Instead of doing the random choices on the middle, the output set is partitioned such that the distribution of messages in the resulting sets is the same as in the entire output set

sulting in no ε strictly greater than zero such that $|\frac{p_x - p_x'}{p_x}| = 0 \geq \varepsilon$.

Our technique is based on the observation that the random choices can be made on the output of the mixnet, as opposed to the output of the first mix. We suggest to divide the output of the mixnet into two sets, where each set has the same distribution of messages (votes) as the distribution of all the messages (final tally).

Instead of making the random choices on the $\mathbb{P}$ set (the output of $\mathfrak{F}_1$), our technique makes the choices on the $\mathbb{O} = \mathbb{Z}_m \times \mathcal{V}$ set, such that the resulting two partitions ($\mathbb{O}_{left}$ and $\mathbb{O}_{right}$) each have the same distribution of elements from $\mathcal{V}$ as their union $\mathbb{O} = \mathbb{O}_{left} \bigcup \mathbb{O}_{right}$. This way it is guaranteed that any of the inputs from I can end-up in any of the messages of the outputs, with the same probability. Figure 3 has an example.

Formally, let $\mathbb{O} = \mathfrak{F}(I)$. $\mathbb{O}$ is a set of pairs $(number, message)$, where the numbers are all the numbers from zero to $m-1$, but the messages are limited to a small set of values $\mathcal{V}$. Let the distribution of messages in $\mathbb{O}$ be $\{p_1\%, p_2\%, ..., p_n\%\}$, where $0 \leq p_i \leq 1$ represents the number of a certain unique message i divided by m. Our technique divides $\mathbb{O}$ into $\mathbb{O}_{left}$ and $\mathbb{O}_{right}$ such that the distribution of messages in both $\mathbb{O}_{left}$ and $\mathbb{O}_{right}$ is also $\{p_1\%, p_2\%, ..., p_n\%\}$.

We group all the elements in the output of the mixnet, $\mathbb{O}$, such that each group contains only one unique message (e.g. group one has {A,A,A,A} and group two has {B,B,B,B}). We then break each group in half (e.g. {A,A},{A,A},{B,B},{B,B}), and combine the halves from the multiple groups (e.g. {A,A,B,B} and {A,A,B,B}).

While this is not the most general way of breaking $\mathbb{O}$ into two sets such that the distribution of messages remains the same in the resulting two sets, it is easy to see that it guarantees our distribution requirement. The next section proves that, even in this particular case, the integrity assurance of our technique is at the same level as the original RPC method.

Note that, in some cases, our technique may be unable to keep the exact same distribution of messages. If the number of outputs carrying the same message is odd, one cannot divide it exactly in half. From this point of view, our technique is best effort: whenever possible, it provides the same distribution, but when not possible, it provides the closest possible distribution. In particular, if there is a single output carrying a unique message (a single vote for one of the candidates), our technique will result in revealing that half of the inputs do not correspond to that message. The original RPC technique suffers from the same problem in these extreme cases.

6 PROVING THAT THE INTEGRITY ASSURANCE IS MAINTAINED

By design, SRPC guarantees maximal privacy offered by a mixnet audited with RPC. We now prove that SRPC offers essentially the same level of integrity.

Using Stirling's approximation, it can be easily derived that $(x \; choose \; \frac{x}{2}) = \binom{x}{\frac{x}{2}} \approx \frac{2^{x-1}}{\sqrt{x}}$.

Assume we have n candidates and each candidate received m_i votes, $\sum_{i=1}^{n} m_i = m$. The number of ways to divide m_i identical votes into two equal sets, is approximately $\frac{2^{m_i-1}}{\sqrt{m_i}}$. If we aggregate this result for all candidates, we get $\prod_{i=1}^{n} \frac{2^{m_i-1}}{\sqrt{m_i}} = \frac{2^{\sum_{i=1}^{n} m_i - 1}}{\sqrt{\prod_{i=1}^{n} m_i}} = \frac{2^{m-n}}{\sqrt{\prod_{i=1}^{n} m_i}}$. To correct for all the possible ways these half parts can be associated, we have to multiple by 2^n. The final number of possible combinations is $\frac{2^m}{\sqrt{\prod_{i=1}^{n} m_i}}$.

For all practical cases, $\frac{2^m}{\sqrt{\prod_{i=1}^{n} m_i}}$ is as good as 2^m, the number of possibilities without the technique presented in this paper. If RPC divides the set $\mathbb{P}$ into two sets with the same cardinality, the number of possibilities is $\binom{m}{\frac{m}{2}} \approx \frac{2^{m-1}}{\sqrt{m}}$, a number even closer to $\frac{2^m}{\sqrt{\prod_{i=1}^{n} m_i}}$.

6.1 Cheating on k ballots

We now analyze a rational case, when the mixnet does not cheat on all ballots, but only on k of them.

Assume there are n candidates and the voters gave m_i votes to candidate i, $\forall i \in \mathbb{Z}_n$. Without loss of generality we assume that $m_{i-1} > m_i, \forall i \neq 0 \in \mathbb{Z}_n$, such

that candidate 0 gets the most votes. We also assume that the mixnet favors the runner-up, candidate 1, and wants to modify the transformations such that the published ballots at the output of the mixnet indicate that candidate 1 won.

One possibility is that the mixnet switches votes only from candidates $2, 3, ..., n-1$ in favor of candidate 1. In this case, the margin plus one, $m_0 - m_1 + 1$, votes need to be switched. Another possibility is that the mixnet switches votes only from candidate m_0 (the true winner). In this case, half the margin plus one, $\frac{m_0 - m_1}{2} + 1$, votes need to be switched. Without demonstration, we claim that the second possibility is less risky for the mixnet. An intuitive explanation is that in the second case only about half the number of ballots need to be cheated on.

We need to compute the probability that the mixnet cheats on $k = \frac{m_0 - m_1}{2} + 1$ ballots and is not detected. With regular RPC this probability is $\frac{1}{2^k}$. We now calculate the probability for Safe RPC.

Let $m_i^{`}$ be the number of votes reported by the mixnet at its output and $m_i$ the number of voters that voted for candidate $i$. Then, for candidate 0 to win, $m_0^{`}$ should be at least equal to $m_0 - k$ and thus $m_1^{`} = m_1 + k$. There are two possible cases. Either $k < \frac{m_1^{`}}{2}$ or $k \geq \frac{m_1^{`}}{2}$.

In the first case, when less than half of the reported votes for the winner are fraudulent, the mixnet has to correctly guess in which of the two partitions all of the k ballots go to, thus requiring k correct guesses. The guesses are independent, because $k < \frac{m_1^{`}}{2}$. Thus the probability of cheating and not getting caught is $\frac{1}{2^k}$ in the case of Safe RPC, exactly the same as RPC.

In the second case, $k \geq \frac{m_1^{`}}{2}$, the mixnet has to correctly guess in which of the two partitions the k votes are going to be in, but the guesses are not independent anymore. Lets assume now that for each of the first $\frac{m_1^{`}}{2}$ votes, the mixnet correctly guesses that they are going to be in the same partition. This is the worst case scenario, since, once the first half of the votes are correctly guessed to be in the first partition, the other $\frac{m_1^{`}}{2}$ ballots are in the opposite partition (no guessing is needed for the second half of the $m_i^{`}$ ballots). The probability of making this correct guess is lower than one in $2^{\frac{m_1^{`}}{2}}$. Thus, in this case, when $k \geq \frac{m_1^{`}}{2}$, the probability of cheating is lower than $\frac{1}{2^{\frac{m_1^{`}}{2}}}$, where $m_1^{`}$ is the number of votes received by the winner declared by the mixnet. This probability is higher than what RPC offers ($\frac{1}{2^k}$), but still a number very close to zero, if the declared winner got a decent number of votes (e.g $m_1^{'} = 40$ implies a probability lower than 0.00000095).

Also, we can prove that if $k \geq \frac{m_1^{`}}{2}$, then the voters prefer candidate 0 almost three times more than candidate 1 in a fair election. Having candidate 1 win may trigger other alarms. Proof: $k \geq \frac{m_1^{`}}{2} \Rightarrow k \geq \frac{m_1 + k}{2} \Rightarrow k \geq m_1 \Rightarrow \frac{m_0 - m_1}{2} + 1 \geq m_1 \Rightarrow m_0 - m_1 + 2 \geq 2 \times m_1 \Rightarrow m_0 \geq 3 \times m_1 - 2$.

7 CONCLUSIONS

In the traditional way Randomized Partial Checking is used for paired mixes, when the number of messages at the output of the second mix is small, situations may arise in which the privacy offered by the two mixes is partially or completely lost.

We described Safe RPC, a technique that does a better audit partitioning, using the output of the second mix. We suggest dividing the output messages into two sets, such that the distribution of each unique message in each of the two sub-sets is the same as the distribution of the entire set. This way, maximal privacy is guaranteed.

We further prove that our technique does not degrade the integrity assurances that the traditional RPC brings, the order of magnitude of the integrity assurance remaining essentially unchanged.

REFERENCES

Chaum, D. (2004). Secret-ballot receipts: True voter-verifiable elections. *IEEE Security and Privacy*, pages 38–47.

Chaum, D., Essex, A., Carback, R., Clark, J., Popoveniuc, S., Sherman, A. T., and Vora, P. (2008). Scantegrity: End-to-end voter verifiable optical-scan voting. *IEEE Security and Privacy*.

Chaum, D. L. (1981). Untraceable electronic mail, return address, and digital pseudonyms. *Communication of ACM*, pages 84–90.

Gomulkiewicz, M., Klonowski, M., and Kutylowski, M. (2003). Rapid mixing and security of Chaums visual electronic voting. In *In Proceedings of ESORICS 2003*, pages 132–145. Springer-Verlag.

Jakobsson, M., Juels, A., and Rivest, R. L. (2002). Making mix nets robust for electronic voting by randomized partial checking. In *Proceedings of the 11th USENIX Security Symposium*, pages 339–353, Berkeley, CA, USA. USENIX Association.

Neff, C. A. (2001). A verifiable secret shuffle and its application to e-voting. In *8th ACM Conference on Computer and Communications Security*, pages 116–125.

Popoveniuc, S. and Hosp, B. (2006). An introduction to PunchScan. In *IAVoSS Workshop On Trustworthy Elections (WOTE 2006)*, Robinson College, Cambridge UK.