Mirrored Commitment: Fixing "Randomized Partial Checking" and Applications

Paweł Lorek[1,5], Moti Yung[2,3], and Filip Zagórski[1,4(✉)]

[1] Wroclaw University, Wrocław, Poland
filip.zagorski@gmail.com
[2] Columbia University, New York, USA
[3] Google, New York, USA
[4] Votifica, Wrocław, Poland
[5] Tooploox, Wrocław, Poland

Abstract. Randomized Partial Checking (RPC) [16] was proposed by Jakobsson, Juels, and Rivest and attracted attention as an efficient method of verifying the correctness of the mixing process in numerous applied scenarios. In fact, RPC is a building block for many electronic voting schemes, including Prêt à Voter [6], Civitas [9], Scantegrity II [5] as well as voting-systems used in real-world elections (*e.g.,* in Australia [4]). Mixing is also used in anonymous transfers of cryptocurrencies. It turned out, however, that a series of works [17,18] showed subtle issues with analyses behind RPC. First, that the actual security level of the RPC protocol is way off the claimed [16] bounds. The probability of successful manipulation of k votes is $(\frac{3}{4})^k$ instead of the claimed $\frac{1}{2^k}$ (this difference, in turn, negatively affects actual implementations of the notion within existing election systems. This is so since concrete implemented procedures of a given length were directly based on this parameter). Further, privacy guarantees [11] that a constant number of mix-servers is enough turned out [17] to also not be correct. We can conclude from the above that these analyses of the processes of mixing are not trivial.

In this paper, we review the relevant attacks, and we present Mirrored-RPC (mRPC) – a fix to RPC based on "mirrored commitment" which makes it optimally secure; namely, having a probability of successful manipulation of k votes $\frac{1}{2^k}$.

Then, we present an analysis of the privacy level of both RPC and mRPC. We show that for n messages, the number of mix-servers (rounds) needed to be ε-close to the uniform distribution in total variation distance is lower bounded by:

$$r(n, \varepsilon) \geq \log_2 \binom{n}{2} / \varepsilon.$$

This proof of privacy, in turn, gives insights into the anonymity of various cryptocurrencies (*e.g.,* Zerocash [23]) using anonymizing pools. If a random fraction q of n existing coins is mixed (in each block), then to achieve full anonymity, the number of blocks one needs to run the protocol for, is:

$$rb(n, q, \varepsilon) \geq -\frac{\log n + \log(n-1) - \log(2\varepsilon)}{\log(1-q^2)}.$$

C. Pöpper and L. Batina (Eds.): ACNS 2024, LNCS 14585, pp. 3–27, 2024.
https://doi.org/10.1007/978-3-031-54776-8_1

1 Introduction

Mix nets, introduced by Chaum [7], constitute an important technique used in many privacy-preserving technologies. For instance, mix nets are a crucial part of many voting systems providing assurance that encrypted ballots posted by voters are correctly decrypted (and tallied). A list of schemes that use mix nets includes systems deployed in publicly binding elections: Estonia, Norway, Switzerland, Australia, USA [4,5,10,13,27]. But applications of mix nets are much wider: anonymous messaging [22], anonymous routing [8], and oblivious RAM [24]. To find a more elaborate list of applications and techniques for verifiable mix nets the reader is encouraged to read [15].

This paper focuses on a central prominent technique by Juels, Jakobsson, and Rivest called Randomized Partial Checking (RPC) [16]. The original Chaumian mix net was designed in the "honest but curious" model, to guarantee senders' privacy provided that at least one mix server is honest. But, a single malicious mix server could replace any number of ciphertexts. In order to decrease the possibility of this happening, RPC was proposed. In RPC, the more ciphertexts are replaced by a server the higher the probability of detecting malfeasance is. The main difference between RPC and other proof-of-shuffle techniques (like [12,26]) is that RPC is much more efficient than other techniques, but provides just a *strong evidence* of correct operations instead of a *proof* of correct operations (but luckily, this confidence is sufficient for many applications). Due to its efficiency, the RPC approach is used in end-to-end voter verifiable systems like Prêt à Voter [6], Scantegrity II [5], and coercion-resistant Civitas [9]. The above have been implemented and applied in real elections. Then, as interest in implementing the technique grew, a series of works [17,18] scrutinized it, and showed that the actual security level of the RPC protocol is way off the initial claim: the probability of successful manipulation of k votes is $(\frac{3}{4})^k$ instead of $\frac{1}{2^k}$ as claimed in [16]. These attacks [17] affected the implementations of Scantegrity and Civitas systems. The level of privacy was affected as well [17]. More on attacks on RPC see Sect. 2.3.

Related Work: Recently [14], a new RPC-type protocol was proposed, where optimal verifiability tolerance $(\frac{1}{2})^k$ is achieved. The protocol assumes that there is a special auditor that becomes the last mix server. After the auditor/mix server publishes decrypted messages it reveals its private keys. While such an approach works in theory, the new protocol role can raise trust-related issues, *e.g.,* now one needs to assume that the special auditor and the second to last mix server do not cooperate (and this configuration solves one weakness by introducing another!). Aside from the proposed attacks, the authors of [18] proposed changes to the protocol that can fix certain attacks, but then they noted that other attacks (which they, in fact, proposed) are "equally harmful." Then, given their finding, they conclude: "This seems to be an inherent problem for RPC mix nets, without an obvious fix."

Our Contributions: We present Mirrored Randomized Partial Checking (mRPC), a protocol that has exactly the same participants, roles, and trust assumptions as the original RPC. The only difference is that a (mirrored) commitment (a commitment to a different value) is published during the protocol execution and one additional value is opened and checked during the audit phase (per message, per server). These changes,

in turn, allow us to achieve optimal verifiability tolerance $(\frac{1}{2})^k$ - compared to $(\frac{3}{4})^k$ in the original RPC. The difference between $1/2^k$ and $(3/4)^k$ is highly significant when considering practical parameters (see Fig. 1). We also show how many mix servers $r(n, \varepsilon)$ are required to mix n messages so that the distribution on permutations (mapping senders to decrypted messages) is ε-close to the uniform distribution on all n-element permutations (in total variation distance). Our proof works for both versions of RPC: Scheme One (Independent Random Selections) and Scheme Two (Pairwise Dependent Selections)[1]. Analysis (Lemma 6) of Scheme One is applicable to (un)linkability in blockchains, while analysis (Lemma 7) of Scheme Two (RPC) is related to anonymity guarantees of election protocols.

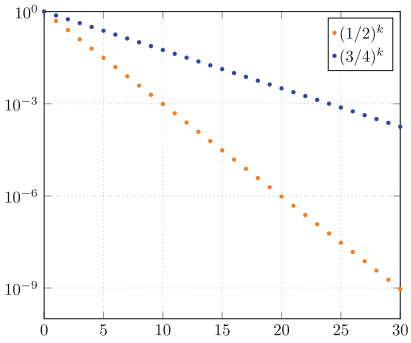


Fig. 1. mRPC guarantees better security level than original RPC. The probability of undetected manipulation of k messages is $(1/2)^k$ for mRPC and $(3/4)^k$ for RPC. x-axis corresponds to k-the number of modified entries; y-axis is the probability of undetectable manipulation.

1.1 Notation

We denote by $[n] = \{1, \ldots, n\}$. Security analysis uses standard assumptions about primitives used by Chaumian RPC mix nets: (1) public key encryption scheme ($\mathcal{E} = \langle \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec} \rangle$) used for Chaumian RPC to be IND-CCA2-secure [3], (2) commitment scheme is perfectly hiding and computationally binding (*e.g.,* Pedersen scheme [21]), (3) the encryption scheme allows for proof of correct decryption.

2 Chaumian Randomized Partial Checking (RPC) Mix Net

We try to closely follow [18] when describing the protocol. A decryption mix net [7] consists of a public, append-only *bulletin board*, mix servers $M_1, \ldots, M_r$, message senders $S_1, \ldots, S_n$ (sometimes we will call them voters) and *auditors*.

[1] As most authors we refer to Pairwise Dependent Selection scheme as to original RPC.

2.1 Protocol Description

The goal of the protocol: mix servers jointly decrypt messages sent by senders (voters), while auditors verify if the decryption process was performed correctly. The following steps are performed.

Setup Phase. Every mix server M_j generates two public/private key pairs $(pk_{2j-1}, sk_{2j-1}), (pk_{2j}, sk_{2j})$ and publishes its public keys pk_{2j-1}, pk_{2j} on the bulletin board.

Submit Phase. Every sender S_i chooses her input plaintext m_i (sometimes we refer to m_i as to a ballot/vote) and submits to the bulletin board B a ciphertext generated in the following process. She first encrypts m_i using pk_{2r} obtaining $c^i_{2r} = \mathsf{Enc}(pk_{2r}, m_i)$. Then, she repeats the following process for $j = 2r-1, 2r-2, \ldots, 0$:

$$c^i_j = \mathsf{Enc}(pk_j, c^i_{j+1}),$$

and submits c^i_0 to the bulletin board B.

Mixing Phase. The sequence $C_0 = \left\langle c^1_0, \ldots, c^n_0 \right\rangle$ of ciphertexts submitted by senders to B is the input to the mixing phase. We denote by $C_0[i] = c^i_0$ and similarly for other sequences.

C_0 is fetched by the first server M_1 which outputs C_2 (each M_j performs two mixing steps $C_{2j-2} \rightsquigarrow C_{2j-1}$ and then $C_{2j-1} \rightsquigarrow C_{2j}$) that is an input to M_2, and so on.

The output produced by M_r (the last mix server): C_{2r} should contain a permuted list of unencrypted input messages $m_1, \ldots, m_n$.

The steps performed by each M_j, $j < r$ are following:

1. **Duplicate Elimination.** M_j removes duplicate entries from its input C_{2j-2}, leaving only a single copy of each entry. Moreover, all messages that correspond to decryption failures $\perp$ are removed. Denote by C'_{2j-2} the resulting sequence, and by $l \leq n$ the number of messages of C'_{2j-2}.
2. **First Mixing.** M_j chooses uniformly at random a permutation π_{2j-1} of $[l]$ and posts on B the sequence C_{2j-1}, where $C_{2j-1}[i] = \mathsf{Dec}(sk_{2j-1}, C'_{2j-2}[\pi_{2j-1}])$.
3. **Second Mixing.** M_j performs the same steps as during the first mixing: selects uniformly at random a permutation π_{2j} of $[l]$. Then it posts on B the sequence C_{2j} where $C_{2j}[i] = \mathsf{Dec}(sk_{2j}, C'_{2j-1}[\pi_{2j}])$.
4. **Posting Commitments.** M_j posts two sequences of commitments on B:
   (a) commitments to the values $\pi^{-1}_{2j-1}(1), \ldots, \pi^{-1}_{2j-1}(l)$,
   (b) commitments to the values $\pi_{2j}(1), \ldots, \pi_{2j}(l)$.

For the clarity of presentation we assume no duplicate elimination took place, *i.e.*, $l = n$.

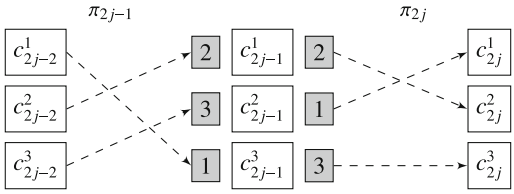


Fig. 2. Original RPC: commitments to $\pi^{-1}_{2j-1}(i)$ and to $\pi_{2j}(i)$ are in shaded squares. Dashed edges/arrows remain secret.

2.2 RPC Audit

During the audit phase, each mix server M_j opens half of the commitments. A set $I_j \subset \{1, \ldots, n\}$ is computed by *e.g.,* xor-ing random bit strings provided by the auditors.

AL If $i \in I_j$ then the mix server M_j is supposed to:
1. open the left link for i, *i.e.,* M_j is supposed to open its i-th commitment from its first sequence of commitments, which should be a commitment on the value $\pi^{-1}_{2j-1}(i)$.
2. post a (non-interavtive zero-knowledge) proof demonstrating that indeed $C_{2j-1}[i]$ is obtained from decrypting $C'_{2j-2}[\pi^{-1}_{2j-1}(i)]$ using sk_{2j-1}.

AR If $i \notin I_j$ then the mix server M_j is supposed to:
1. open the right link: the commitment to the value $\pi_{2j}(i)$.
2. post a (non-interactive zero-knowledge) proof that $C_{2j}[\pi_{2j}(i)]$ is obtained from decrypting $C_{2j-1}[i]$ using sk_{2j}.

Set I_j defines a corresponding challenge string (also called audit string) $B_j = b_{j,1}b_{j,2}\ldots b_{j,n}$ for $b_{j,i} \in \{0, 1\}$, where $b_{j,i} = 0$ if and only if $i \in I_j$.

Example 1 (RPC audit). Let us assume that the jth server committed to the values presented in the Fig. 2 and during the audit, an audit string $B_j = 010$ was selected ($I_j = \{1, 3\}$). Commitments to $\pi^{-1}_{2j-1}(1)$, $\pi^{-1}_{2j-1}(3)$ and to $\pi_{2j}(2)$ are opened. Corresponding proofs of correct decryptions are shown (along solid arrows) *e.g.,* that c^2_{2j-1} correctly decrypts to c^1_{2j} under the public key pk_{2j}. It is visualized on Fig. 3.

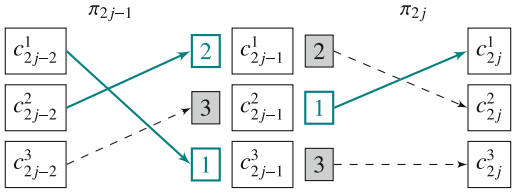


Fig. 3. RPC audit example for server M_j and audit string $B_j = 010$. Dashed edges and corresponding commitments remain hidden.

2.3 Attacks on RPC

In this section we describe and analyse attacks on RPC. The first attack was presented in [18] and later described in [17].

Attacks by the Last Mix Sever. To ilustrate the attack by the last mix-server, let us consider the following example with $n = l = 3$ votes. Let $m_1 = m_2 = A$ (2 votes for candidate A) while $m_3 = B$ (1 vote for B). Say, the honest permutation is $\pi_{2r} = (2, 1, 3)$ (Fig. 2), however, M_r is cheating and it publishes commitments to $\pi'_{2r} = (1, 1, 3)$ (which is not a permutation) (Fig. 4).

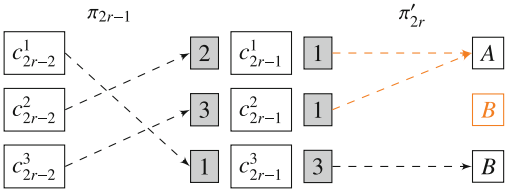
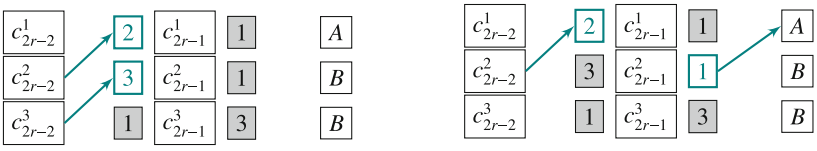


Fig. 4. Example: attack by the last mix server. A vote for B is copied while a vote for A is removed.

The audit string is of the form $B_r = b_{r,1}b_{r,2}b_{r,3}$. The value of $b_{r,3}$ is irrelevant for this attack, we are thus left with four choices for $b_{r,1}b_{r,2}$. All four situations are depicted in Fig. 5. If $b_{r,1} = b_{r,2} = 1$ then M_r is asked to open $\pi'_{2r}(1)$ and $\pi'_{2r}(2)$ and the cheating is detected. In all other cases, the cheating is not detected (since there are two entries pointing to the same element). In other words, one can detect a single message manipulation with probability $1/4$.



(a) Challenge: $00*$ ($I_r = \{1, 2\}$). Audit passed.

(b) Challenge: $01*$ ($I_r = \{1\}$). Audit passed.

(c) Challenge: $10*$ ($I_r = \{2\}$). Audit passed.

(d) Challenge $11*$ ($I_r = \emptyset$). Audit fails.

Fig. 5. RPC detects a single message manipulation just with probability $\frac{1}{4}$.

Attacks by Any Mix Server. Here, we present an attack that was proposed in [17]. This attack can be performed by any server. Let $l = n = 3$ and consider the server M_j with honest inverse permutation $\pi^{-1}_{2j-1} = (2, 3, 1)$ (Fig. 2), however the server publishes commitments to $\pi'^{-1}_{2j-1} = (1, 3, 1)$ (Fig. 6).

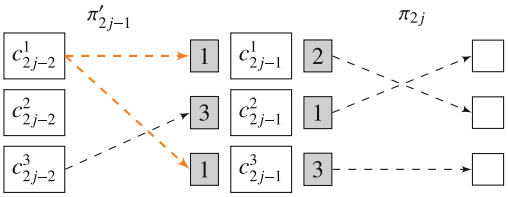
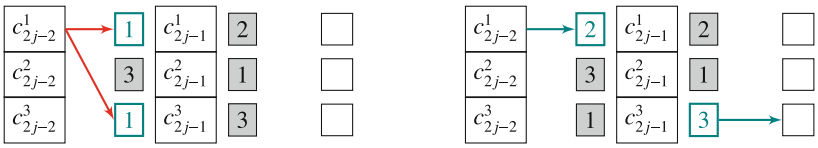


Fig. 6. Example: attack on any layer by server M_j. The result of the attack can be: a vote for B is copied while a vote for A is removed.

If the audit string $b_{j,1}b_{j,2}b_{j,3}$ is such that $b_{j,1} = b_{j,3} = 0$, then the server is asked to open $\pi'^{-1}_{2j-1}(1)$ and $\pi'^{-1}_{2j-1}(3)$ and the manipulation is detected. Note that in any other case for values of $b_{j,1}$, $b_{j,3}$, it is not detected. All four situations for values of $b_{j,1}$, $b_{j,3}$ are depicted in Fig. 7.

Summarizing, such a manipulation is detected with probability $1/4$ (and thus it is undetected with probability $3/4$) in case of one manipulation. In general, when k messages are manipulated, the probability of not detecting it is $(3/4)^k$ (for details see Theorem 1 in [18]).



(a) Challenge: $0 * 0$ ($I_r = \{1, 3\}$). Audit fails.

(b) Challenge: $0 * 1$. Audit passed.

(c) Challenge: $1 * 0$. Audit passed.

(d) Challenge $1 * 1$. Audit passed.

Fig. 7. RPC detects a single message manipulation just with probability $\frac{1}{4}$.

3 Mirrored Randomized Partial Checking (mRPC)

In this section, we present a fix to RPC which we call Mirrored-RPC (mRPC) protocol and prove that it guarantees optimal level of manipulation detection *i.e.,* manipulation of k messages is detected with probability $1 - (1/2)^k$.

In RPC protocol, each mix server publishes two lists of commitments to the "middle column" (ciphertexts that are the result of the first mixing phase, see Fig. 2), more precisely server M_j for each entry i publishes:

- commitments to $\pi_{2j-1}^{-1}(i)$ (where data comes from), and
- commitments to $\pi_{2j}(i)$ (where data goes to).

In mRPC commitments are published on the "outer columns" – see Fig. 8. This change allows for detecting manipulations with higher probability than the original RPC.

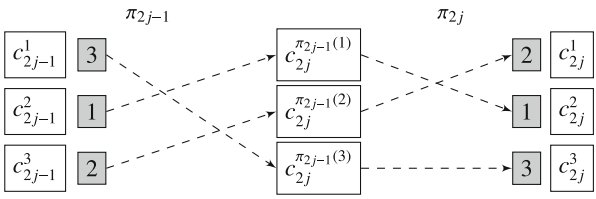


Fig. 8. mRPC: For each entry of a left column, a commitment to "where to" is published and a commitment to "where from" for entries of a right column is published (In the original RPC these commitments are only published for the entries that are the result of **first mixing**.).

3.1 Protocol Description

Setup phase The setup is exactly the same as in the original RPC (Sect. 2.1).

Submit phase This phase is exactly the same as in the original RPC (Sect. 2.1).

Mixing phase The mixing phase stays almost the same as the *Mixing phase* of the original RPC (see Sect. 2.1) the only difference is in the part: *Posting commitments*.

1. **Duplicate elimination** the same as in original RPC.
2. **First mixing** – the same as in original RPC.
3. **Second mixing** – the same as in original RPC.
4. **Posting commitments** M_j posts **two** sequences of commitments on B:
   (a) commitments to the values $\pi_{2j-1}(1), \ldots, \pi_{2j-1}(l)$,
   (b) commitments to the values $\pi_{2j}^{-1}(1), \ldots, \pi_{2j}^{-1}(l)$.

Note that RPC in **Posting commitments** phase in step 4a posts: $\pi_{2j-1}^{-1}(1), \ldots, \pi_{2j-1}^{-1}(l)$ and in step 4b posts: $\pi_{2j}(1), \ldots, \pi_{2j}(l)$. Similarly as in RPC, for clarity of presentation, we assume no duplicate elimination took place in mRPC, *i.e.,* $l = n$.

3.2 mRPC Audit

During the audit phase, each mix server M_j opens half of the commitments. A set $I_j \subset \{1, \ldots, n\}$ is computed by *e.g.,* xor-ing random bit strings provided by the auditors. Set I_j defines a corresponding challenge string (also called audit string) $B_j = b_{j,1} b_{j,2} \ldots b_{j,n}$ for $b_{j,i} \in \{0, 1\}$, where $b_{j,x} = 0$ if and only if $x \in I_j$.

AL If $x \in I_j$ *i.e.,* $b_{j,x} = 0$, then the mix server M_j is supposed to:
1 (bidirectional checking):
  (a) publish value y;
  (b) then open $z = \pi_{2j-1}(y)$ and check if $z = x$;
2 post a non-interactive zero-knowledge proof demonstrating that indeed $C_{2j-1}[x]$ is obtained from decrypting $C'_{2j-2}[y]$ using sk_{2j-1}.

AR If $x \notin I_j$ *i.e.,* $b_{j,x} = 1$, then the mix server M_j is supposed to:
1 (bidirectional checking):
  (a) publish value y;
  (b) open the commitment to $z = \pi_{2j}^{-1}(y)$ and check if $z = x$.
2 post a non-interactive zero-knowledge proof that $C_{2j}[y]$ is obtained from decrypting $C_{2j-1}[x]$ using sk_{2j}.

Example 2 (mRPC Audit). Let us assume that the jth server committed to the values presented in Fig. 8. The audit is presented in Fig. 9. During the audit phase, an audit string $b = 010 = b_1 b_2 b_3$ (defining the corresponding $I_j = \{1, 3\}$). The jth server needs to publish:

AL for $x \in \{1, 3\}$:
1. for $x = 1$, $y_1 = 2$, $z = \pi_{2j-1}(y_1) = 1 = x$, a non-interactive ZKP that $C_{2j-1}[1]$ is obtained from decrypting $C_{2j-2}[2]$;
2. for $x = 3$, $y_3 = 1$, $z = \pi_{2j-1}(y_3) = 3 = x$, a non-interactive ZKP that $C_{2j-1}[3]$ is obtained from decrypting $C_{2j-2}[1]$;

AR for $x \in I_j^c = \{2\}$:
1. for $x = 2$, $y_2 = 1$, $z = \pi_{2j}^{-1}(y_2) = 2 = x$, a non-interactive ZKP that $C_{2j}[1]$ is obtained from decrypting $C_{2j-1}[2]$.

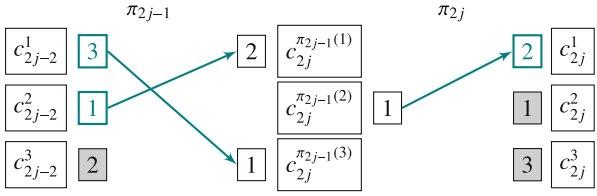


Fig. 9. mRPC audit example for server M_j with $B_j = 010$. Dashed edges and corresponding commitments remain hidden.

3.3 Attack Examples on mRPC

Attack by the Last Mix Server. Let us reconsider the attack described in Sect. 2.3. The dishonest "permutation" together with all commitments is depicted in Fig. 10.

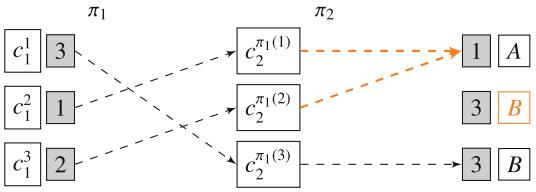
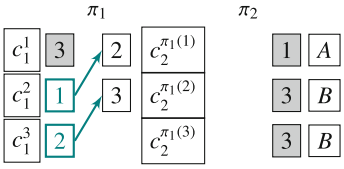


Fig. 10. Attack by the last mix server in mRPC.

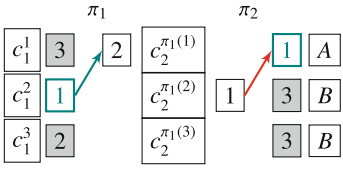
For $b_{r,1} = b_{r,2} = 1$, the cheating is detected (Fig. 11 (d)). However, this is not the only situation when the manipulation is detected.

- Assume that the server commits to $\pi'^{-1}_{2r} = (1, *, 3)$. Consider $b_{r,1} = 0, b_{r,2} = 1$. In RPC server M_r is asked to open $\pi'_{2r}(2) = y = 1$, in mRPC the server is additionally asked to open $\pi'^{-1}_{2j}(y) = \pi'^{-1}_{2j}(1)$, which is 1 and the cheating is detected – Fig. 11(b).
- Assume that the server commits to $\pi'^{-1}_{2r} = (2, *, 3)$. Consider $b_{r,1} = 1, b_{r,2} = 0$. Then M_r is asked to open $\pi'_{2r}(1) = y = 1$ and additionally $\pi'^{-1}_{2r}(1) = 2$. This case is presented in Fig. 15(c).

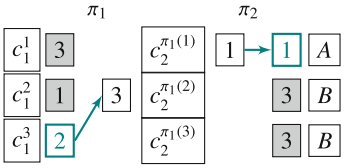
In any case (for manipulated permutations), the manipulation will caught for 2 audit strings $b_{r,1}b_{r,2}$ out of 4, thus with probability 1/2. All options are depicted in Fig. 11.



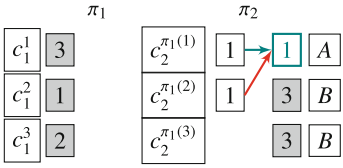
(a) Challenge string: 00∗. Audit passed.



(b) Challenge string: 01∗. For $x = 2$ value $y = 1$ is published but opened commitment $z = 1 \neq x$. Audit failed.



(c) Challenge: 10∗. Audit passed.



(d) Challenge 11∗. For $x = 2, y = 1$ is published but opened commitment is for $z = 1 \neq x$. Audit failed.

Fig. 11. A view of a bulletin board after the audit step of mRPC.

Attacks by Any Mix Server. Let us continue the setup Sect. 2.3. The attack, in the presence of additional commitments is presented in Fig. 12.

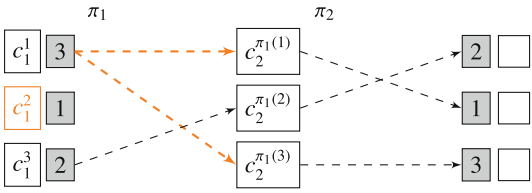
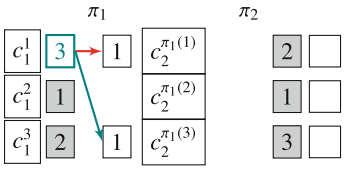


Fig. 12. Attack on any layer in mRPC.

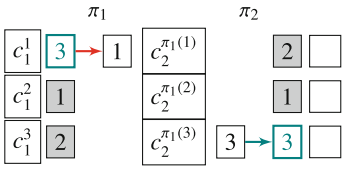
Again, once $b_{j,1} = b_{j,3} = 0$, the manipulation is detected. Consider cases:

- Assume that server commits to $\pi'_{2j-1} = (3, *, 2)$. Consider $b_{j,1} = 0, b_{j,3} = 1$. In RPC server M_j is asked to open $\pi'^{-1}_{2j-1}(1) = y = 1$, in mRPC server is additionally asked to open $\pi'_{2j-1}(1)$ which is 3 and the manipulation is detected.
- Assume now that the commitment is $\pi'_{2j-1} = (1, *, 2)$. Then in case $b_{j,1} = 1, b_{j,3} = 0$ the server must open $\pi'^{-1}_{2j-1}(3)$ which commits to 1.

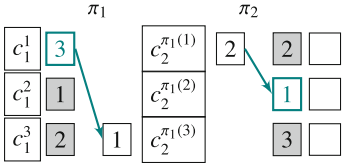
In any case (for manipulated permutations), the manipulation will be detected in two out of four possibilities for $b_{j,1}$ and $b_{j,3}$, i.e., with probability 1/2. All the situations (for $\pi'_{2j-1} = (3, 1, 2)$) are presented in Fig. 13, other cases are "symmetric" (see Fig. 15).



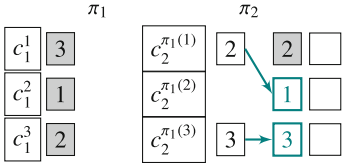
(a) Challenge: $0 * 0$.
Adit failed.



(b) Challenge: $0 * 1$.
Audit failed.



(c) Challenge: $1 * 0$.
Audit passed.



(d) Challenge $1 * 1$.
Audit passed.

Fig. 13. mRPC ggdetects a single message manipulation with probability $\frac{1}{2}$. For the same settings, RPC succeeds in detecting only with probability $\frac{1}{4}$.

The same attack as in Sect. 2.3 (Attacks by the last mix server) is presented, except in the first row of the last column, commitment is to row 2 from the middle column. The attack is then detected for different challenge strings but still with 50% probability (Fig. 14).

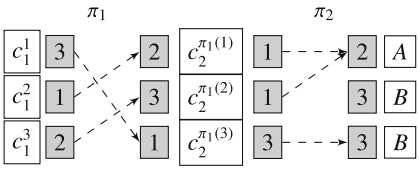
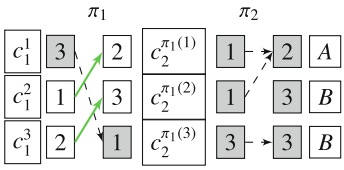
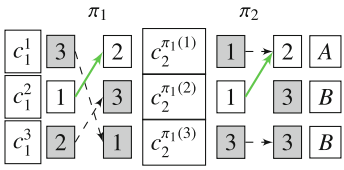


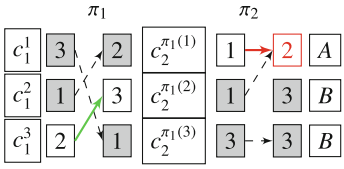
Fig. 14. Attack by the last mix server in mRPC.



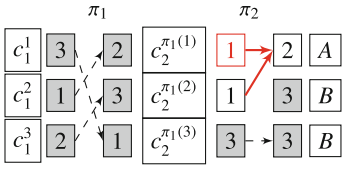
(a) Challenge: 00∗.
Audit passed.



(b) Challenge: 01∗.
Audit passed.



(c) Challenge: 10∗.
Audit fails.



(d) Challenge 11∗.
Audit fails.

Fig. 15. The same attack as in Sect. 3.4 is presented but with attacker committing to different values.

3.4 Security of mRPC

Lemma 1. *For mRPC, the probability of undetectable modification of k entries by any mix server M_j, during one mixing step is upper bounded by* $\frac{1}{2^k}$.

Proof. We will show the proof for an odd $(2j-1)$ mixing step, the reasoning for an even $(2j)$ mixing step is similar. The input to the mix server M_j is a list of ciphertexts $C_{2j-1} = \left\langle c_{2j-1}^1, \ldots, c_{2j-1}^n \right\rangle$ published by the server M_{j-1} (or the users/voters for $j = 1$).

During the *Mixing phase*, M_j posts:

- the result of the *First mixing*, ciphertexts: $C_{2j} = \left\langle c_{2j}^1, \ldots, c_{2j}^n \right\rangle$,
- commitments $t_1, \ldots, t_n$.

If M_j is honest then for some $\pi_{2j-1} \in S_n$ and $y = \pi_{2j-1}(x)$ for all $x \in [n]$ the following equations hold:

$$t_x = \mathsf{Comm}(y), \tag{1}$$

$$c_{2j}^y = \mathsf{Dec}_{sk_{2j-1}}(c_{2j-1}^x). \tag{2}$$

During the audit step if $b_{j,y} = 0$ ($y \in I_j$) the following steps are performed:

AL.1 (bidirectional checking):
1. M_j publishes z,
2. M_j opens commitment $t_z = \mathsf{Comm}(y')$ to y',
3. auditor checks if $y = y'$.

AL.2 (proof of correct decryption):
1. M_j publishes the proof that Eq. 2 holds for y and z,
2. auditor verifies the proof.

If M_j is dishonest and decides to manipulate k entries from positions in a set $A \subset \{1, \ldots, n\}$ ($|A| = k$) it means that M_j will not be able to pass $AL.2$ part of the audit for $x \in A$.

M_j may try to post commitments to different positions but since the commitment check **AL.1** is bidirectional, a single entry from C_{2j-1} can be mapped only to a single entry in C_{2j}. And since C_{2j} lacks entries corresponding to ciphertexts from positions in A it will be detected in the **AL.2** part of the check whenever for such an entry a challenge bit 0 will be chosen.

Since there are k positions with that property, the probability of not detecting that k entries were dropped (replaced) is equal to $1/2^k$.

The main theorem is a direct conclusion from Lemma 1.

Theorem 1 (mRPC security). *For mRPC, the probability of undetectable modification of k entries by any mix server is upper bounded by* $\frac{1}{2^k}$.

4 Privacy Guarantees of RPC and mRPC

4.1 Constant Number of Mix-Servers

In [11] it was shown that for a scenario when votes are cast only on one of the two candidates, the constant number of mix servers is enough.

Here we show that for arbitrary messages (*e.g.,* for Australian-type ballots), a constant number of mix-servers is not enough.

Example 3 (Bulletin board leaks information). RPC auditing process may reveal a lot of information about voters' preferences. Figure 16 presents an extreme example. There are two candidates A, B and 8 voters $v_1, \ldots, v_8$. With only $r = 2$ mixing servers, a lot of information may be available to an adversary, even when he just observes publicly accessible information.

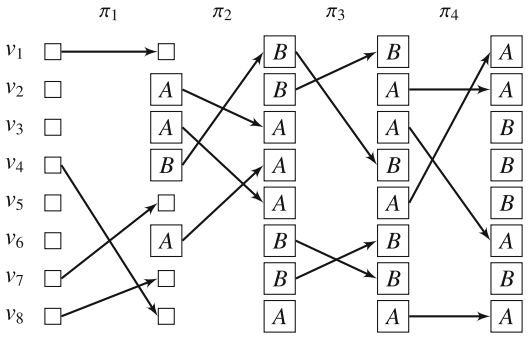


Fig. 16. RPC for small number of mix servers may reveal a lot of information. Just by observing bulletin board, an adversary may say that voters $\{v_1, v_4, v_7, v_8\}$ cast votes $\{A, B, B, B\}$ while voters $\{v_2, v_3, v_5, v_6\}$ cast votes: $\{A, A, A, B\}$.

The situation becomes much worse if an adversary knows exactly how some voters voted. For the example at Fig. 16, knowlede that v_4 voted for A reveals that v_1, v_7, v_8 voted for B.

Example 4 (Anonymity for arbitrary messages.). For a general case, when there are more types of messages (*e.g.,* ballots in Australia), senders' privacy is still at risk. In the most general case, every message is unique. The insight behind privacy definition is achieved in the following way: for an adversary, every permutation should be possible with almost the same probability – *i.e.,* distribution on permutations generated by the RPC process should be close to the uniform distribution.

By π we denote a permutation obtained by applying permutations $\pi_1, \ldots, \pi_4$, and revealing parts of them during the audit phase (see Fig. 17). It is easy to see that $P[\pi = (*, *, 5, *, *, 7, *, *)] = 0 = P[\pi = (*, *, 7, *, *, 5, *, *)]$ (it is impossible that message 5 was sent by v_3 at the same time when message 7 was sent by v_6, and vice versa). There are a lot of other permutations that are impossible to achieve.

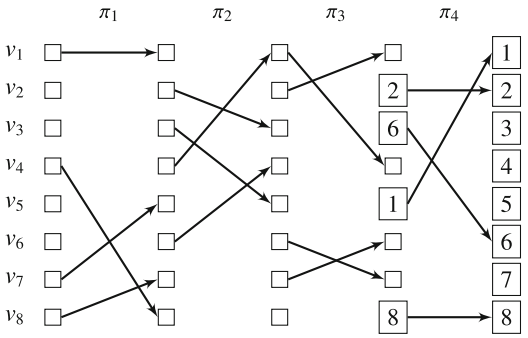


Fig. 17. RPC for a small number of mix servers may reveal a lot of information. The probability that senders v_2, v_3 sent messages $5, 7$ respectively is equal 0. One can exclude a lot of other combinations. A similar analysis can be applied to linkability of many crypto-currencies, *e.g.,* Zerocash [23].

Definition 1. *For mix-server entries x_i, x_j, we denote by $M_{at}(x_i, x_j) = t$ if M_t was the first mix-server for which entries were both audited during the same step. By audited by server M_k we mean that both x_i and x_j were assigned the same audit bits.*

Note that if entires x_i, x_j were audited in M_t then the entries were *mixed* – someone observing only revealed links does not know their relative ordering.

Lemma 2. *For any entries x_i, x_j the probability that they will be mixed for the first time in the k-th mix-server is equal to $\frac{1}{2^k}$, i.e., $P\left(M_{at}(x_i, x_j) = k\right) = \frac{1}{2^k}$.*

Lemma 3. *Let r be the number of mix-servers, and n be the number of processed entries. If $\binom{n}{2} \geq 2^r$ then with high probability there exists a pair of entries $i, j \in \{1, \ldots, n\}$ such that $M_{at}(x_i, x_j) > r$.*

The proof of Lemma 3 follows from a birthday paradox argument.

By $\mathsf{RPC}_{r,n}$ we mean a random permutation obtained by processing n messages through an RPC cascade of r mix-servers having the knowledge on so far opened links. By $\mathcal{L}(\mathsf{RPC}_{k,n})$ we denote the distribution of the scheme at step $k \leq r$ and by $\mathcal{U}(\mathcal{S}_n)$ we denote the uniform distribution, both on $\mathcal{S}_n$ – a set of permutations of n elements. We will use a *total variation distance* between two distributions μ, ν on a common finite state space $\mathbb{E}$ as

$$\mathrm{TVD}[\mu, \nu] = \frac{1}{2} \sum_{e \in \mathbb{E}} |\mu(e) - \nu(e)|.$$

The conclusion of Lemma 4 is that a constant number of mix-servers is not enough to privately process arbitrary messages.

Lemma 4. *Let r be the number of mix-servers, and n be the number of processed entries. If $\binom{n}{2} \geq 2^r$ and in the last server m left links are open, then*

$$\mathrm{TVD}\left[\mathcal{U}(\mathcal{S}_n), \mathcal{L}(\mathit{RPC}_{r,\sqrt{2^r}})\right] > 1 - \frac{2\,m(n-m)}{n(n-1)} \overset{(*)}{\geq} \frac{1}{2} - \frac{1}{2(n-1)},$$

where equality in $(*)$ is achieved for $m = n/2$ (say n is even).

The proof of Lemma 4 is in Appendix A.1, the bound is depicted in Fig. 18.

4.2 Mixing Time

In this section, we show the required number of mix-servers to achieve high level of privacy. The main result concerning the privacy of RPC and mRPC is the following.

Lemma 5. *Let n be the number of processed entries by a r-server RPC mix or mRPC. If*

$$r = r(n, \varepsilon) \geq \log_2 \left[\binom{n}{2} / \varepsilon \right]$$

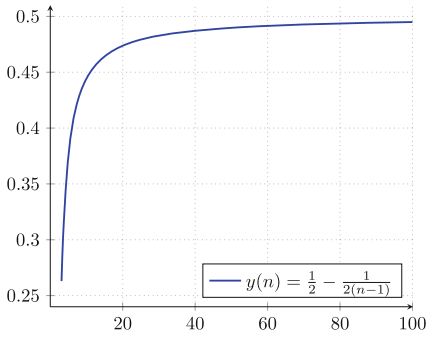


Fig. 18. Lower bound on $\mathrm{TVD}\left[\mathcal{U}(\mathcal{S}_n), \mathcal{L}(\mathsf{RPC}_{r,n})\right] \geq \frac{1}{2} - \frac{1}{2(n-1)}$.

then

$$\mathrm{TVD}\left[\mathcal{U}(\mathcal{S}_n), \mathcal{L}(RPC_{r,n})\right] \leq \varepsilon.$$

We start – Lemma 6 – with the RPC/mRPC Scheme One, i.e., the case when each server is asked to open left/right connections independently. Moreover, we assume that each entry is opened with some predefined probability $p \in (0,1)$. Note that it is equivalent to actually considering $2r$ servers, each performing a single permutation – we consider however r servers, each performing two permutations, to be consistent with lemmas related to Scheme Two.

Afterwards, in Lemma 7 we show the result for Scheme Two. Lemma 5 is a direct consequence of the latter (substitute $p = 1/2$).

Lemma 6. *Let n be the number of processed entries by a r-server RPC or mRPC mix Scheme One. Each server is asked to open any connection independently with probability $p \in (0,1)$. If*

$$r = r(n,\varepsilon) \geq \frac{1}{2} \log_{\frac{1}{1-(1-p)^2}} \left[\binom{n}{2} / \varepsilon \right]$$

then

$$\mathrm{TVD}\left[\mathcal{U}(\mathcal{S}_n), \mathcal{L}(RPC_{r,n})\right] \leq \varepsilon.$$

The proof of Lemma 6 is in Appendix A.2, it is based on strong stationary times (SST, introduced in [1,2]), a tool from a Markov chain theory.

Remark. It is worth mentioning that SST T from Lemma 6 (see its proof) resembles SST constructed in [19] for riffle shuffle scheme. Note that the RPC and the riffle shuffle are quite different – in RPC full permutation is applied in each step and each connection is revealed with probability p, whereas in riffle shuffle only the specific type of permutation in each step is performed and p corresponds to revealing some bits used to perform it. Note also that it takes $\frac{1}{2} \log_{\frac{1}{1-(1-p)^2}} \left[\binom{n}{2} / \varepsilon \right]$ for RPC to mix, whereas it takes $\log_{\frac{2}{1-(1-p)^2}} \left[\binom{n}{2} / \varepsilon \right]$ for riffle shuffle to mix.

Let us consider the following example.

Example 5. Consider $n = 6$ an assume that $B_1 = 001010, B_2 = 010011$, *i.e.*, in first steps outgoing connections from nodes $1, 2, 4$ and 6 are revealed and in the second step the outgoing connections from nodes $1, 3$, and 4. With this knowledge, the adversary knows that with equal probability one of the permutations is possible:

$$(4,1,6,3,2,5), (4,1,2,3,6,5), (4,6,1,3,2,5), (4,2,1,3,6,5),$$
$$(4,6,2,3,1,5), (4,2,6,3,1,5), (4,1,6,5,2,3), (4,1,2,5,6,3),$$
$$(4,6,1,5,2,3), (4,2,1,5,6,3), (4,6,2,5,1,3), (4,2,6,5,1,3).$$

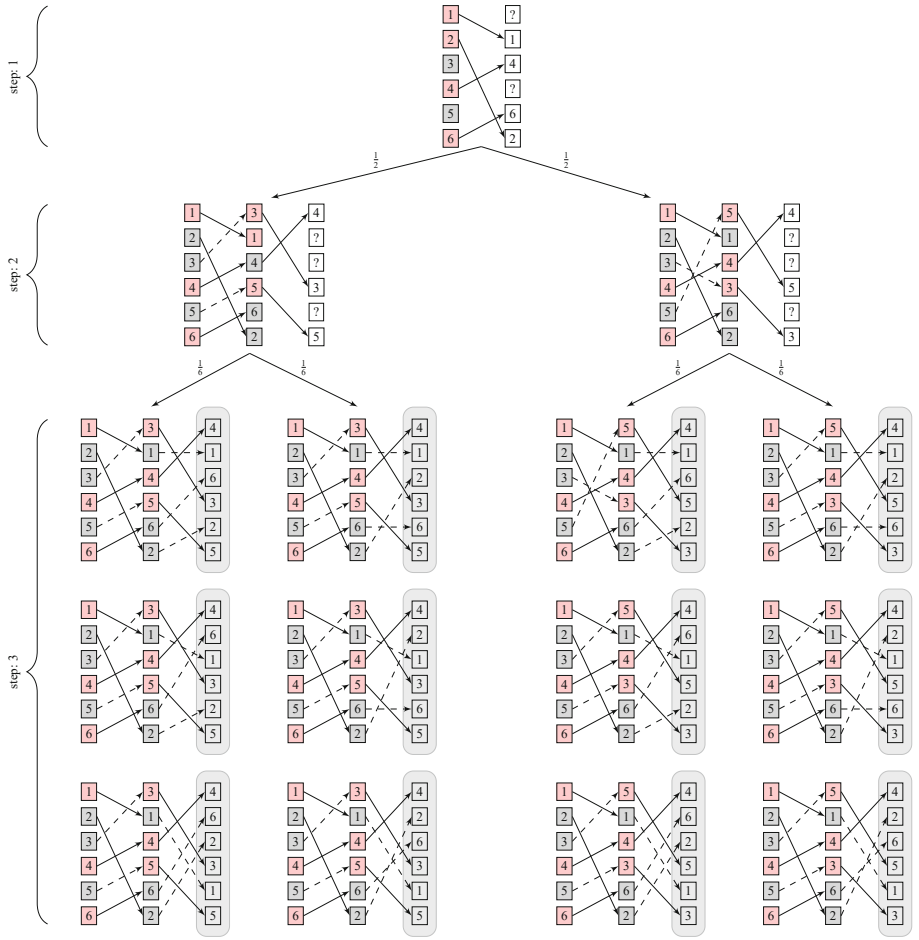
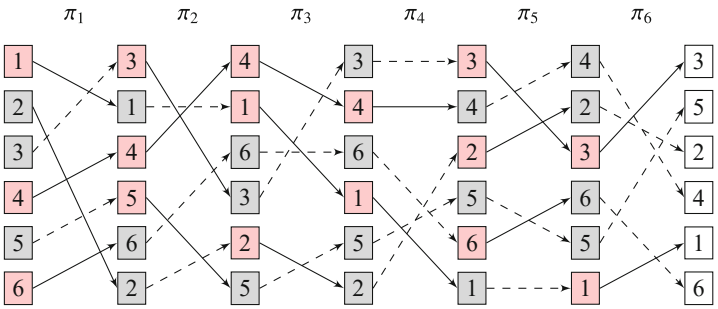


Fig. 19. 3 steps of execution of RPC for $n = 6$. All options for unrevealed nodes shown. Revealed connections depicted as solid lines (corresponding nodes are red), unrevealed ones as dashed lines (corresponding nodes are gray). After these three steps Y_3 has the uniform distribution on 12 permutations emphasized by gray regions. (Color figure online)

All possible situations are depicted in Fig. 19. In Fig. 20, one realization of this example is depicted – then all the pairs are mixed, thus the resulting permutation is random. (Note that for $n = 6$ such a situation happens on average after processing by $\frac{1}{2}\log_{\frac{4}{3}}\binom{6}{2} = 4.7$ servers).



	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6
Mixed pairs	{**3,5**}	{**1,2**},{**1,6**} {**2,6**}	{3,5},{**3,6**} {**5,6**}	{**2,4**},{**2,5**} {2,6},{**4,5**} {**4,6**},{5,6}	{**1,3**},{**1,5**} {3,5}	{1,2},{1,3} {**1,4**},{**2,3**} {2,4},{**3,4**}
Total nr of mixed pairs	1	4	6	10	12	15 STOP

Fig. 20. Sample execution of mixing of $n = 6$ elements. Newly mixed pairs are in **bold**. In this example after 6 steps an adversay has no knowledge on the final permutation (all $\binom{6}{2} = 15$ pairs are mixed). After three steps his knowledge is depicted in Fig. 19

In the following Lemma 7 we show the result for Scheme Two.

Lemma 7. *Let n be the number of processed entries by a r-server RPC or mRPC mix Scheme Two. Each server is asked to open any left link independently with probability $p \in (0, 1)$, then the right links corresponding to non-opened left ones, are open.*

$$r = r(n, \varepsilon) \geq \log_{\frac{1}{2p(1-p)}} \left[\binom{n}{2} / \varepsilon \right]$$

then

$$\mathrm{TVD}\left[\mathcal{U}(\mathcal{S}_n), \mathcal{L}(RPC_{r,n})\right] \leq \varepsilon.$$

The proof of Lemma 7 is in Appendix A.3. Note that for p close to 0 or 1 there will be many pairs mixed in step $2k - 1$ or $2k$. The worst situation i.e., the smallest number of mixed pairs (on average) will be for $p = 1/2$. For cases $n \in \{100, 10\,000, 1\,000\,000\}$ the average number of steps, as a function of p is depicted in Fig. 21.

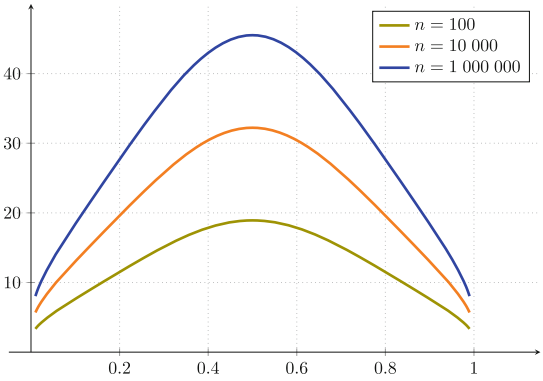


Fig. 21. Average number of RPC servers needed to mix $n \in \{100, 10\,000, 1\,000\,000\}$ entries (for $\varepsilon = \frac{1}{100}$). In the worst case $p = 1/2$ we need 19 servers on average.

5 Application: CryptoCurrency Unlinkability

In most popular cryptocurrencies, payments are performed between pseudonyms. Since transactions are published on a public ledger, payment transactions remain traceable. There were a couple of approaches that introduce untraceability to blockchain cryptocurrencies: Zerocoin [20], Zerocash [23] (used in ZCash), CryptoNote [25] (used in Monero).

In this section, we want to show a link between Lemma 6 and the anonymity guarantees of various cryptocurrencies. We assume that the anonymization protocol is similar to the one that is used in Zerocash. If one wants to measure the anonymity level of a given system in total variation distance then it corresponds to the mixing time of RPC Scheme One which is expressed in Lemma 6. Instead of applying the following equation:

$$r(n, p, \varepsilon) \geq \frac{1}{2} \log_{\frac{1}{1-(1-p)^2}} \left[\binom{n}{2} / \varepsilon \right] = -\frac{\log n + \log(n-1) - \log(2\varepsilon)}{2\log(1-(1-p)^2)}$$

it seems simpler is simpler think about the function of $q = 1 - p$ – here q corresponds to the fraction of entries being in a mix. Then let $rb(n, q, \varepsilon) = r(n, p, \varepsilon)$:

$$rb(n, q, \varepsilon) \geq \frac{1}{2} \log_{\frac{1}{1-q^2}} \left[\binom{n}{2} / \varepsilon \right] = -\frac{\log n + \log(n-1) - \log(2\varepsilon)}{2\log(1-q^2)}$$

One needs to approximate n, *e.g.,* by applying the simplifications (1) and (2) below; then $nq = n(1-p)$ would be the average number of transactions (or from/to addresses) in a single block. The number $rb(n, q, \varepsilon)$ denotes the required number of mix-servers so the resulting permutation is ε close in total variation distance to the uniform distribution on n elements. Since each server performs two independent permutations, the required number of steps is equal to $2rb(n, q, \varepsilon)$ (Fig. 22).

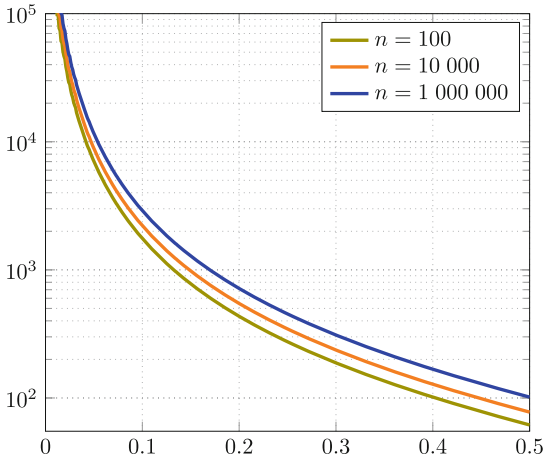


Fig. 22. Number of rounds (for cryptocurrencies the number of blocks) $2rb(n, q, \varepsilon)$ needed to be processed for a system with n entries to be close to the uniform distribution, as a function of q, ε. The x-axis of the plot is q – the probability of selecting an element to the mix $\varepsilon = 1/100$.

To give insights, we make a series of further simplifications:

(1) each transaction is of a nominal value (*e.g.,* 1 BTC/1 ETH/...),
(2) each pseudonym (public key) is linked with a single nominal value,
(3) each coin is selected to be used in a payment transaction independently, uniformly at random with probability $1 - p$ (in the Lemma 6 p corresponds to the opened links).

Then, assuming that all transactions are "shielded", with the data of 5/1/2022 (source: https://bitinfocharts.com) the results are following:

– for Ethereum $1.63 \cdot 10^{13}$ blocks (For $n = 120\,606\,657$, $p = \frac{186}{n}$, $\varepsilon = \frac{1}{10}$) would be needed ($2.56 \cdot 10^9$ days),
– for Bitcoin $6.03 \cdot 10^9$ blocks ($3.54 \cdot 10^7$ days).

6 Conclusions

We presented Mirrored Randomized Partial Checking (mRPC), the protocol that eliminates attacks on Randomized Partial Checking. Proposed mRPC makes minimal changes to the original protocol but allows for upper bounding probability of successful attack by an adversary to $(\frac{1}{2})^k$ - compared to $(\frac{3}{4})^k$ in the original RPC. The presented approach can be applied to fix Civitas and Scantegrity II voting systems.

We also provided an analysis of privacy guarantees offered by RPC. Our analysis gives also insights into the level of anonymity of cryptocurrencies. We conclude that due to the need for many steps (high value of $rb(n, q, \varepsilon)$ for small values of q) and the need for speedy transactions (that enforce low values of q), de-anonymization will be open to some attacks due to insufficient mixing.

A Proofs

A.1 Proof of Lemma 4

Proof. Recall that I_j is a subset of $[n]$ for which left link is revelead (challenge bit is set to 0). Let us denote $S_{j,0} = I_j$ and $S_{j,1} = [n] \setminus I_j$, *i.e.*, those messages for which right link is revealed (challenge bit 1). In terms of an audit string $B_j = b_{j,1}b_{j,2}\ldots b_{j,n}$, we may rewrite $S_{j,b} = \{i : b_{j,i} = b\}$.

If two elements x, y are not mixed in the M_j mix, it means that $x \in S_{j,b}$ and $y \in S_{j,1-b}$ for $b \in \{0, 1\}$.

Let us compare distance between the uniform distribution $\mathcal{U}(\mathcal{S}_n)$ on n-element permutations to the distribution $\mathcal{L}(\mathsf{RPC}_{r,n})$ when $n \geq \sqrt{2^r}$.

From Lemma 3 there exists two mix entries x, y that are not yet mixed after r steps, with high probability. It means that $x \in S_{1,b_1}, S_{2,b_2}, \ldots, S_{r,b_r}$ and $y \in S_{1,1-b_1}, S_{2,1-b_2}, \ldots, S_{r,1-b_r}$ for $b_1, \ldots, b_c \in \{0, 1\}$.

Let $\mathcal{S}_n^0$ be the set of all permutations for which $x \in S_{r,b}$ and $y \in S_{r,1-b}$ for $b = 0, 1$. From the assumptions we have that $|S_{r,0}| = m$. From Lemma 3, with high probability, only permutations from $\mathcal{S}_n^0$ have nonzero probabilities in distribution $\mathcal{L}(\mathsf{RPC}_{r,\sqrt{2^r}})$. In other words, we can write that the probability of σ under $\mathcal{L}(\mathsf{RPC}_{r,\sqrt{2^r}})$ is $f(\sigma)$ such that

$$f(\sigma) \quad \begin{cases} > 0 \ \text{ if } \sigma \in \mathcal{S}_n^0, \\ = 0 \ \text{otherwise}, \end{cases}$$

for some distribution f on $\mathcal{S}^0$ (Fig. 23).

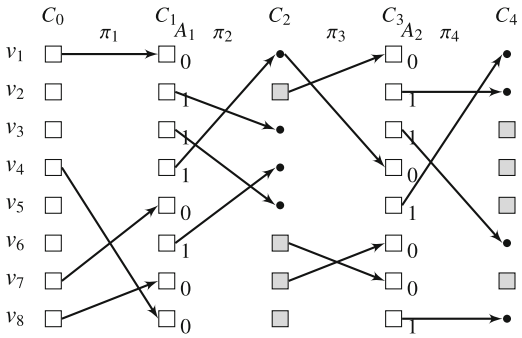


Fig. 23. Representation of sets $S_{1,0}, S_{1,1}$ for M_1 and sets $S_{2,0}, S_{2,1}$ for M_2. Audit/challenge bits A_1, A_2 for M_1, M_2 are presented next to columns C_1, C_3. Sets $S_{j,0}$ are denoted by ■ and sets $S_{j,1}$ are denoted by •.

Now, let us compute the distance between uniform distribution and the distribution $\mathcal{L}(\mathsf{RPC}_{r,n})$ for a set of permutations $\mathcal{S}_n^0$ such that m left links were opened, *i.e.*, $|I_r| = |S_{r,0}| = m$.

$$\begin{aligned}
&\mathrm{TVD}\left[\mathcal{U}(\mathcal{S}_n), \mathcal{L}(\mathsf{RPC}_{r,n}) \,\middle|\, |S_{r,0}| = m\right] = \\
&= \frac{1}{2}\left(\sum_{\substack{\sigma\in\mathcal{S}_n^0\\ |S_{r,0}|=m}} \left|f(\sigma) - \frac{1}{n!}\right| + \sum_{\substack{\sigma\notin\mathcal{S}_n^0\\ |S_{r,0}|=m}} \frac{1}{n!}\right) \geq \frac{1}{2}\left(\sum_{\substack{\sigma\in\mathcal{S}_n^0\\ |S_{r,0}|=m}} \left(f(\sigma) - \frac{1}{n!}\right) + \sum_{\substack{\sigma\notin\mathcal{S}_n^0\\ |S_{r,0}|=m}} \frac{1}{n!}\right) \\
&= \frac{1}{2} + \frac{1}{2}\left(\sum_{\substack{\sigma\notin\mathcal{S}_n^0\\ |S_{r,0}|=m}} \frac{1}{n!} - \sum_{\substack{\sigma\in\mathcal{S}_n^0\\ |S_{r,0}|=m}} \frac{1}{n!}\right) \qquad = \frac{1}{2} + \frac{1}{2n!}\left(n! - 2|\{\sigma \in \mathcal{S}_n^0 : |S_{r,0}| = m\}|\right) \\
&\qquad\qquad = 1 - \frac{|\{\sigma \in \mathcal{S}_n^0 : |S_{r,0}| = m\}}{n!}
\end{aligned}$$

Noting that

$$|\{\sigma \in \mathcal{S}_n^0 : |S_{r,0}| = m\}| = 2m(n-m)(n-2)!$$

we have

$$\mathrm{TVD}\left[\mathcal{U}(\mathcal{S}_n), \mathcal{L}(\mathsf{RPC}_{r,n}) \,\middle|\, |S_{r,0}| = m\right] \geq 1 - \frac{2\,m(n-m)}{n(n-1)}.$$

The worst-case is exactly half left links are open (say n is even), *i.e.*, $m = n/2$, then

$$\begin{aligned}
\mathrm{TVD}\left[\mathcal{U}(\mathcal{S}_n), \mathcal{L}(\mathsf{RPC}_{r,n}) \,\middle|\, |S_{r,0}| = m\right] &\geq \mathrm{TVD}\left[\mathcal{U}(\mathcal{S}_n), \mathcal{L}(\mathsf{RPC}_{r,n}) \,\middle|\, |S_{r,0}| = n/2\right] \\
&\geq 1 - \frac{2\frac{n}{2}\frac{n}{2}}{n(n-1)} = \frac{1}{2} - \frac{1}{2(n-1)}.
\end{aligned}$$

A.2 Proof of Lemma 6

Proof. We will use some tools from Markov chain theory. We will consider two chains $\{X_t\}_{t\geq 0}, \{Y_t\}_{t\geq 0}$ on $\mathcal{S}_n$. We set $X_0 = Y_0$ to be the identity permutation (note that $\mathsf{RPC}_{0,n}$ is the identity permutation).

Recall that server j performs permutations π_{2j-1} and π_{2j}, in total $2r$ permutations are performed.

Concerning X_{t+1}: it is X_t to which we apply a uniformly random permutation $\pi_t = (\pi_t(1), \ldots, \pi_t(n))$ (note that then $X_t \sim \mathcal{U}(\mathcal{S}_n)$ for any $t \geq 1$).

Note that in Scheme One each server performs independently identical (in distribuion) steps. That is why we will look at the distribution after each application of π_t.

Concerning Y_t, this is X_t with the following extra knowledge. Let $B_t = b_{t,1}, \ldots, b_{t,n}$ be the n random bits chosen independently from the distribution $P(b_{t,i} = 0) = p = 1 - P(b_{t,i} = 1)$.

Now assume that the entries $S_{j,0} = \{j : b_{t,j} = 0\}$ from the permutation π_t are opened. Y_t has distribution of X_t provided we have a knowledge of $B_1, \ldots, B_t$. This corresponds to $\mathsf{RPC}_{t,n}$. Since $\{Y_t\}_{t\geq 0}$ is ergodic and aperiodic, the uniform distribution is the stationary distribution. By $\mathcal{L}(Y_t)$ we denote the distribution of Y_t.

We will use the strong stationary times (SST) approach (introduced in [1,2]). We say that T is an SST for $\{Y_k\}$ if for any permutation σ we have $P(Y_t = \sigma | T = t) = 1/n!$. For such SST we have $\mathrm{TVD}\left[\mathcal{L}(Y_k), \mathcal{U}(\mathcal{S}_n)\right] \leq P(T > t)$ (see, *e.g.*, Theorem 6 in [1]).

Let us define

$$T_{ij} = \min\{t : b_{t,i} = b_{t,j} = 1\},$$

i.e., this is the first time that both elements i and j were not opened. At this time the relative ordering of i and j is random (since π_k is uniformly random). Note that the probability that this will not happen in one step is $1-(1-p)^2$ (at least one entry was opened), thus $P(T_{ij} > t) = (1-(1-p)^2)^t$.

Now, let T be the first time when all the pairs of elements were not opened in at least one step. It means that all $\binom{n}{2}$ pairs are in random relative order – and that means that the permutation itself is random (since π_t's are uniformly random). In other words, T is an SST for $\{Y_t\}$. We may compute

$$\begin{aligned}\mathrm{TVD}\left[\mathcal{L}(Y_k), \mathcal{U}(\mathcal{S}_n)\right] \leq \quad & P(T > t) \quad = P\left(\bigcup_{1\leq i<j\leq n}\{T_{ij} > t\}\right) \\ \leq \sum_{1\leq i<j\leq n} P(T_{ij} > t) &= \sum_{1\leq i<j\leq n}\left(1-(1-p)^2\right)^t = \binom{n}{2}\left(1-(1-p)^2\right)^t.\end{aligned}$$

Taking $t = \log_{1-(1-p)^2}\left[\binom{n}{2}/\varepsilon\right]$, we have $\mathrm{TVD}\left[\mathcal{L}(Y_k), \mathcal{U}(\mathcal{S}_n)\right] \leq \varepsilon$. In total there are $t = 2r$ permutations, thus the proof is completed.

A.3 Proof of Lemma 7

Proof. The proof is similar to the proof of Lemma 6. The t-th server applies two permutations π_{2t-1} and π_{2t}, then each left link is opened independently with probability p, *i.e.*, $B_{2t-1} = b_{2t-1,1}, \ldots, b_{2t-1,n}$ with i.i.d. $P(b_{2t-1,j} = 0) = p = P(b_{2t-1,j} = 1), j = 1, \ldots, n$. However the audit string B_{2t} is uniquely determined:

$$B_{2t} = (b_{2t,1}, \ldots, b_{2t,n}) = (1 - b_{2t-1,1}, \ldots, 1 - b_{2t-1,n}).$$

The situation is depicted in Fig. 24. Again, let

$$T_{ij} = \min\{t : b_{t,i} = b_{t,j} = 1\},$$

i.e., this is the first moment that elements i and j were not opened in the same permutation. Consider steps $2t-1$ and $2t$: the elements i and j will be both opened in the same step if *i*) they are both revealed in step $2t-1$; *ii*) they are both not opened in step $2t-1$ (since then they surely will be in next step). Thus, the pair will not be mixed in steps $2t-1$ and $2t$ with probability $2p(1-p)$. We have

$$P(T_{ij} > 2t) = (2p(1-p))^t.$$

Again, since all permutations π_t's are random, the first moment T when all the pairs are mixed is an SST, and we have (consider t even)

$$\begin{aligned}&\mathrm{TVD}\left[\mathcal{L}(Y_t), \mathcal{U}(\mathcal{S}_n)\right] \\ &\leq P(T > t) = P\left(\bigcup_{1\leq i<j\leq n}\{T_{ij} > t\}\right) \\ &\leq \sum_{1\leq i<j\leq n} P(T_{ij} > 2t/2) = \sum_{1\leq i<j\leq n}(2p(1-p))^{\frac{t}{2}} \\ &= \binom{n}{2}(2p(1-p))^{\frac{t}{2}}.\end{aligned}$$

Taking the last step, *i.e.*, $t = 2r$ we have that $r = \log_{\frac{1}{2p(1-p)}} \left[\binom{n}{2}/\varepsilon\right]$ what completes the proof.

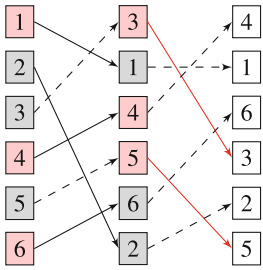


Fig. 24. Situation similar to Fig. 20: π_1 and π_2 and $B_1 = 001010$ are the same as there, but now B_2 is determined by B_1, namely $b_i^2 = 1 - b_i^1$ – opened connections depicted in red.

References

1. Aldous, D., Diaconis, P.: Shuffling cards and stopping times. Am. Math. Mon. **93**(5), 333–348 (1986)
2. Aldous, D., Diaconis, P.: Strong uniform times and finite random walks. Adv. Appl. Math. **8**(1), 69–97 (1987)
3. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among notions of security for public-key encryption schemes. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 26–45. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0055718
4. Burton, C., Culnane, C., Heather, J.: Thea Peacock, Peter YA Ryan, Steve A Schneider, Vanessa Teague, Roland Wen, Zhe Xia, and Sriramkrishnan Srinivasan. Using prêt à voter in victoria state elections. EVT/WOTE, 2 (2012)
5. Carback, R.T., et al.: The scantegrity voting system and its use in the takoma park elections. In: Real-World Electronic Voting, pp. 253–292. Auerbach Publications (2016)
6. Chaum, D., Ryan, P.Y.A., Schneider, S.: A practical voter-verifiable election scheme. In: di Vimercati, S.C., Syverson, P., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 118–139. Springer, Heidelberg (2005). https://doi.org/10.1007/11555827_8
7. Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM **24**(2), 84–90 (1981)
8. Chen, C., Asoni, D.E., Barrera, D., Danezis, G., Perrig, A.: Hornet: high-speed onion routing at the network layer. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 1441–1454 (2015)
9. Clarkson, M.R., Chong, S., Myers, A.C.: Civitas: toward a secure voting system. In: 2008 IEEE Symposium on Security and Privacy (S&P 2008), pp. 354–368. IEEE (2008)
10. Gjøsteen, K.: The Norwegian internet voting protocol. In: Kiayias, A., Lipmaa, H. (eds.) Vote-ID 2011. LNCS, vol. 7187, pp. 1–18. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32747-6_1
11. Gomułkiewicz, M., Klonowski, M., Kutyłowski, M.: Rapid mixing and security of Chaum's visual electronic voting. In: Snekkenes, E., Gollmann, D. (eds.) ESORICS 2003. LNCS, vol. 2808, pp. 132–145. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39650-5_8

12. Groth, J., Ishai, Y.: Sub-linear zero-knowledge argument for correctness of a shuffle. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 379–396. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_22
13. Haenni, R., Koenig, R.E., Locher, P., Dubuis, E.: Chvote system specification (2017)
14. Haines, T., Müller, J.: Optimal randomized partial checking for decryption mix nets. In: Baek, J., Ruj, S. (eds.) ACISP 2021. LNCS, vol. 13083, pp. 277–292. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-90567-5_14
15. Haines, T., Müller, J.: Sok: techniques for verifiable mix nets. In: 2020 IEEE 33rd Computer Security Foundations Symposium (CSF), pp. 49–64 (2020)
16. Jakobsson, M., Juels, A., Rivest, R.L.: Making mix nets robust for electronic voting by randomized partial checking. In: USENIX Security Symposium, San Francisco, USA, pp. 339–353 (2002)
17. Khazaei, S., Wikström, D.: Randomized partial checking revisited. In: Dawson, E. (ed.) CT-RSA 2013. LNCS, vol. 7779, pp. 115–128. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36095-4_8
18. Küsters, R., Truderung, T., Vogt, A.: Formal analysis of Chaumian mix nets with randomized partial checking. In: 2014 IEEE Symposium on Security and Privacy, pp. 343–358. IEEE (2014)
19. Lorek, P., Kulis, M., Zagórski, F.: Leakage-resilient riffle shuffle. In: Blömer, J., Kotsireas, I.S., Kutsia, T., Simos, D.E. (eds.) MACIS 2017. LNCS, vol. 10693, pp. 395–408. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-72453-9_32
20. Miers, I., Garman, C., Green, M., Rubinm, A.D.: Zerocoin: anonymous distributed e-cash from bitcoin. In: 2013 IEEE Symposium on Security and Privacy, pp. 397–411. IEEE (2013)
21. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-46766-1_9
22. Piotrowska, A.M., Hayes, J., Elahi, T., Meiser, S., Danezis, G.: The loopix anonymity system. In: 26th {USENIX} Security Symposium ({USENIX} Security 17), pp. 1199–1216 (2017)
23. Sasson, E.B., et al.: Zerocash: decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy, pp. 459–474. IEEE (2014)
24. Toledo, R.R., Danezis, G., Echizen, I.: Mix-ORAM: using delegated shuffles. In: Proceedings of the 2017 on Workshop on Privacy in the Electronic Society, pp. 51–61 (2017)
25. van Saberhagen, N.: Cryptonote v 1.0 (2012). https://cryptonote.org/whitepaperv1.pdf (2021)
26. Wikström, D.: A commitment-consistent proof of a shuffle. In: Boyd, C., González Nieto, J. (eds.) ACISP 2009. LNCS, vol. 5594, pp. 407–421. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02620-1_28
27. Douglas Wikström. Verificatum (2018). https://www.verificatum.org/