Optimal Randomized Partial Checking for Decryption Mix Nets

Thomas Haines[1] and Johannes Müller[2][0000−0003−2134−3099]

[1] Norwegian University of Science and Technology, Norway,
firstname.lastname@ntnu.no
[2] SnT, University of Luxembourg, Luxembourg,
firstname.lastname@uni.lu

Abstract. One of the most important verifiability techniques for mix nets is *randomized partial checking (RPC)*. This method is employed in a number of prominent secure e-voting systems, including Prêt à Voter, Civitas, and Scantegrity II, some of which have also been used for real political elections including in Australia.
Unfortunately, it turned out that there exists a significant gap between the intended and the actual verifiability tolerance of the original RPC protocol. This mismatch affects exactly the "Achilles heel" of RPC, namely those application scenarios where manipulating a few messages can swap the final result (e.g., in close runoff elections).
In this work, we propose the first RPC protocol which closes the aforementioned gap for decryption mix nets. We prove that our new RPC protocol achieves an optimal verifiability level, without introducing any disadvantages. Current implementations of RPC for decryption mix nets, in particular for real-world secure e-voting, should adopt our changes to improve their security.

1 Introduction

Mix nets are indispensable building blocks of many secure e-voting systems. Essentially, a mix net consists of a sequence of mix servers which take as input the encrypted messages provided by the senders (e.g., the voters' ballots), secretely shuffle them, and eventually output the permutated plain messages (e.g., votes). Unless all mix servers are corrupted, the mixing breaks the individual connections between the senders and their revealed messages in the output. In the context of e-voting, this property guarantees vote privacy.

For *secure* e-voting, it is also important to ensure that the voters' intent be reflected correctly in the election result, even if the mix servers are corrupted and actively try to tamper with the votes. For this purpose, a mix net must be *verifiable* to guarantee that manipulating the senders' input, and generally incorrect mixing, can be detected. In the literature,

numerous mix nets have been proposed that aim to achieve verifiability (see, e.g., [1, 2, 11, 14–16, 18, 18, 21–23]).

One of the most prominent verifiability techniques for mix nets is *randomized partial checking (RPC)*, originally introduced by Jakobsson, Juels, and Rivest [15]. RPC combines several advantageous features:

- *Wide field of applications*: RPC allows one to realize secure electronic elections even if the voters' choices are complex. This is the case for Instant Runoff Voting (IRV) or Single Transferable Vote (STV) which are commonly used in political elections all over the world, for example in Australia, India, Ireland and the UK. Such elections cannot be handled easily by homomorphic e-voting schemes.
- *Intuitive concept*: The main idea of RPC is exceptionally simple: Once a mix server has completed its mixing, the mix server is challenged to open the links between a number of randomly chosen output messages and the respective input ciphertexts. If the mix server manipulated (i.e., dropped or replaced) one of the associated input ciphertexts, then this will be detected.
- *Lightweight and simple crypto*: The computational overhead of RPC is small. Moreover, RPC requires well-studied black-box cryptographic primitives only. This is particularly advantageous when it comes to implementing a verifiable mix net correctly in practice. For example, the recently discovered attacks on the Internet voting scheme that was supposed to be employed in Swiss federal elections [12] mainly reduce to the fact that the sophisticated cryptographic components related to the underlying proof of shuffle were not implemented correctly.

Due to these features, RPC mix nets are used in several prominent secure e-voting systems, including Prêt à Voter [7], Civitas [8], and Scantegrity II [5]. Some of these systems have also been used for real political elections, for example in the Australian state of Victoria [3].

Unfortunately, it turned out that the verifiability tolerance of the original RPC protocol [15] is significantly worse than intended. Jakobsson et al. [15] stated that manipulating k messages in the original RPC protocol remains undetected with probability at most $(\frac{1}{2})^k$ but this claim was disproven subsequently. A number of pitfalls were discovered [17, 20] that allow for manipulating k messages in the original RPC protocol but which remain undetected with probability $(\frac{3}{4})^k$. This gap affects exactly the "Achilles heel" of RPC, namely those application scenarios where manipulating a few messages can swap the final result (e.g., in close runoff

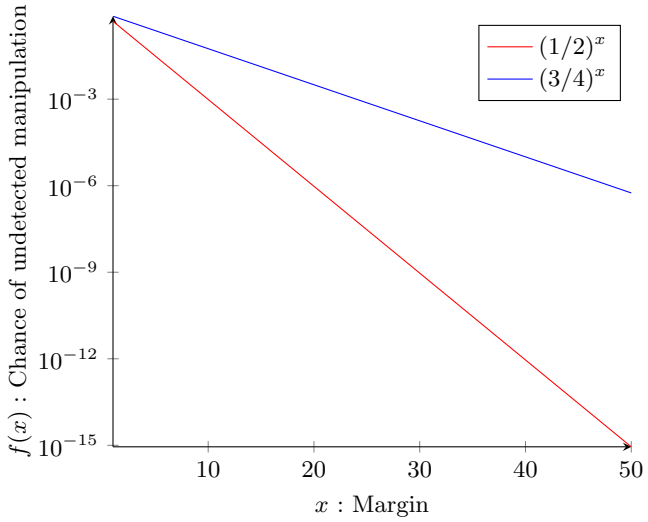

Fig. 1. Difference in concrete verifiability tolerances between original RPC (blue) and optimal RPC (red).

elections). In such cases, the asymptotic behaviour of the verifiability tolerance is rather irrelevant; instead, it is important that the base of the exponential function is small. We illustrate this in Fig. 1.

Elections with close margins are fairly common. For example, in the 2020 Queensland election, Bundaberg had a margin of 9 which the original RPC protocol would have allowed to be changed undetectably with probability 7.5%. On the contrary, in an RPC protocol with optimal verifiability tolerance, i.e., $(\frac{1}{2})^k$, swapping the election result would have been caught with 99.8% probability. Designing such an *optimal RPC* protocol and proving it secure is the main objective of this paper.

Contributions. In this paper, we provide the following contributions:

1. We propose an optimal RPC protocol for decryption mix nets. Our new protocol preserves all advantages of the original RPC protocol: it is widely applicable, intuitive, lightweight, and does not require any cryptographic primitives in addition to the basic ones employed in original RPC.
2. We formally prove that the new RPC protocol improves the verifiability tolerance of the original one from $(\frac{3}{4})^k$ down to $(\frac{1}{2})^k$ under the same cryptographic and trust assumptions. For this purpose, we

use the verifiability framework by Küsters, Truderung, and Vogt [19] which was already applied in [20] to analyze the original version of RPC decryption mix nets, as well as all other techniques for verifiable mix nets (see [13]). We emphasize that the attacks discovered on the original RPC protocol demonstrate the importance of such a formal treatment.

Current implementations of RPC for decryption mix nets, in particular for real-world secure e-voting, should adopt our changes to improve their security.

Structure of the paper. We discuss the relation between our new optimal RPC protocol and the other techniques for verifiable mix nets from the literature in Sec. 2. In Sec. 3, we explain how a decryption mix net works at a conceptual level, and in Sec. 4, we describe how it can be extended by the original RPC protocol. In Sec. 5, we recall the pitfalls of the original RPC protocol and how they can be exploited to attack it. In Sec. 6, we propose our new RPC protocol for decryption mix nets. In Sec. 7, we state that our new RPC protocol is indeed optimal. The complete formal analysis is provided in App. A. We conclude in Sec. 8.

2 Related Work

Many verifiable mix nets have been proposed in the literature. According to a recent systemization-of-knowledge [13], the underlying verifiability techniques can be classified as follows: message tracing [18,22], verification codes [18,21], original RPC [15], trip wires [2,16], message replication [16], and proofs of shuffle (e.g., [1, 11, 14, 23]). Optimal RPC and the other verifiability techniques relate as follows.[3] We will provide more details on the relation between original and optimal RPC in the subsequent sections.

Let us first elaborate on the *verifiability tolerances*, i.e., the probability that manipulating more than k messages remains undetected, of the different techniques and their relationships. Together with the message tracing, verification codes, original RPC, and trip wires technique, the optimal RPC technique belongs to a class of verifiability techniques which have a verifiability tolerance of the form f^{k+1} where f is some linear function. Compared to original RPC for which $f = \frac{3}{4}$ holds true,

[3] Since the optimal RPC technique proposed in this paper is tailored to decryption mix nets, we restrict our attention to verifiability techniques for these mix nets in what follows and refer to [13] for further details.

we have $f = \frac{1}{2}$ for optimal RPC. This shows that, on the one hand, the verifiability tolerance of the original and the new RPC protocol are asymptotically equivalent, but on the other hand, our new version significantly improves the "Achilles heel" of RPC, i.e., the range of small values of k, where some cheating may remain undetected with non-negligible probability (see Fig. 1). Compared to the remaining techniques in this class, f is constant both for optimal and original RPC. This property is, typically, superior to the verifiability tolerance of the message tracing and verification codes technique for which the base $f = (1 - p)$ depends on the senders' individual, and thus uncertain, verification probability p. Compared to the trip wire technique for which the base $f = n_{\mathsf{S}}^h/(n_{\mathsf{S}}^h + n_{\mathsf{tw}})$ can be decreased by increasing the number of trip wire messages n_{tw} for a given number of honest senders n_{S}^h, the verifiability tolerance of original and optimal RPC is inferior. However, tripwires unlike RPC allows manipulation of dishonest senders' messages without detection which is unacceptable in many circumstances.

Both original and optimal RPC for decryption mix nets employ *moderate cryptographic primitives* (black-box NIZKP of correct decryption), require a (temporarily) *trusted auditor* whose role can easily be distributed, and guarantee *individual accountability* (i.e., each misbehaving mix server can be identified individually).

3 Decryption Mix Nets

A decryption mix net [6] consists of a sequence of mix servers, denoted by $M_1, \ldots, M_{n_{\mathsf{MS}}}$. Each mix server M_j holds a public/private key pair $(\mathsf{pk}_j, \mathsf{sk}_j)$ of an IND-CCA2-secure public key encryption scheme $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$. Each sender S encrypts her message m under the mix servers' public keys in reverse order:

$$c = \mathsf{Enc}(\mathsf{pk}_1, \mathsf{Enc}(\ldots \mathsf{Enc}(\mathsf{pk}_{n_{\mathsf{MS}}}, m))).$$

The first mix server M_1 takes as input the senders' nested input ciphertexts, decrypts them using its secret key sk_1, permutes the result uniformly at random, and forwards the shuffled list to M_2. The remaining mix servers $M_2, \ldots, M_{n_{\mathsf{MS}}}$ repeat the same steps using their secret keys $\mathsf{sk}_2, \ldots, \mathsf{sk}_{n_{\mathsf{MS}}}$. Eventually, the last mix server $M_{n_{\mathsf{MS}}}$ returns the senders' original plain input messages in randomly permuted order.

The main purpose of a mix net is to ensure *message privacy* by "breaking" the individual links between the senders and their plain messages.

To this end, at least one mix server should not be corrupted but keep its secret key as well as its internal permutation secret.

Note that if a number of senders were dishonest and aimed for breaking message privacy of some honest sender, then the dishonest senders could simply duplicate the honest sender's input ciphertext multiple times. By this, the targeted honest sender's message would be amplified in the final outcome. This means that an honest sender's message privacy could be undermined even if all mix servers are perfectly honest. In order to protect against such *replay attacks*, each mix server removes all duplicates (except for one per duplicate group) from its input vector.

4 Original RPC

We recall the general idea of the original RPC protocol for decryption mix nets as proposed by Jakobsson, Juels, and Rivest [15].

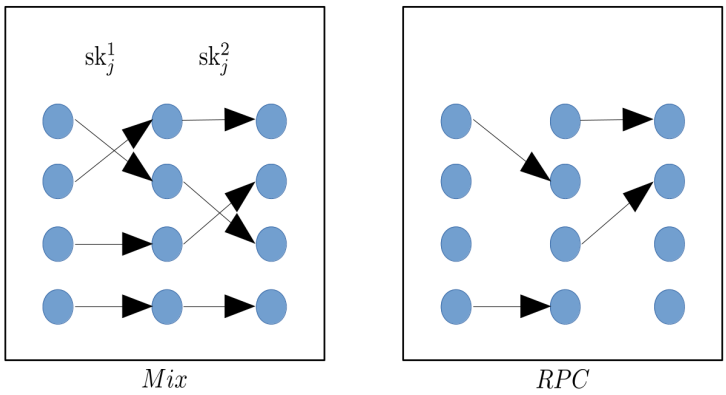

Fig. 2. *Examplified illustration of RPC*: The left box shows the internal view of mix server M_j during the mixing process. The right box shows the public view of the links revealed by M_j during RPC for challenge $\boldsymbol{\alpha}_j = (1, -1, 1, -1)$.

If at least one mix server in a "plain" decryption mix net (as described in Sec. 3) is corrupted and actively deviates from its protocol specification, then it is not possible to verify (without further means) whether the final outcome consists of the senders' original messages. In order to extend a "plain" decryption mix net so that the correctness of the final outcome can publicly be verified, Jakobsson, Juels, and Rivest [15] proposed the concept of *randomized partial checking (RPC)*.

The main idea of RPC is to challenge each mix server M_j as follows. After the mixing phase, an auditor A chooses a fraction of M_j's input ciphertexts uniformly at random. For each chosen ciphertext c, the mix server has to "open" the link between c and its decryption c' in its output. For this purpose, the mix server M_j generates a non-interactive zero-knowledge proof (NIZKP) which proves the respective decryption relation w.r.t. M_j's public key pk_j. Then, the auditor (and everybody else) can verify the NIZKPs returned by M_j. If the check for one of the chosen ciphertexts fails, then M_j is held accountable and the final outcome is rejected. Because the mix server does not know in advance which links it needs to open during audit, the probability that the mix server can manipulate some messages undetectably decreases with the number of links to be opened.

Typically, pairs of mix servers are audited to ensure that traces through the mix net are not revealed completely.[4] We will therefore assume that each mix server M_j has two public/private key pairs $(\mathsf{pk}_j^1, \mathsf{sk}_j^1), (\mathsf{pk}_j^2, \mathsf{sk}_j^2)$ and performs two mixing steps. We denote M_j's input by $\boldsymbol{c}_j^0$, its intermediate ciphertext vector by $\boldsymbol{c}_j^1$, and its output by $\boldsymbol{c}_j^2$. Now, the idea is that for any randomly chosen intermediate ciphertext $c^1 \in \boldsymbol{c}_j^1$, the mix server has to open either its link to $c^0 \in \boldsymbol{c}_j^0$ or its link to $c^2 \in \boldsymbol{c}_j^2$. In this way, one of c^1's links, the one to c^0 or the one to c^2, remains secret. We denote the auditor's *challenge vector* for M_j by $\boldsymbol{\alpha}_j$, where $\boldsymbol{\alpha}_j[i] = -1$ if the left link of $\boldsymbol{c}_j^1[i]$, $\boldsymbol{\alpha}_j[i] = 1$ if the right link of $\boldsymbol{c}_j^1[i]$, and $\boldsymbol{\alpha}_j[i] = 0$ if none of the links of $\boldsymbol{c}_j^1[i]$ is supposed to be opened. We illustrate this approach in Fig. 2.

5 Attacks on Original RPC

Jakobsson, Juels, and Rivest [15] claimed that the original RPC technique (Sec. 4) provides the following verifiability guarantee (if always the left or the right link of an intermediate ciphertext is supposed to be opened).

Claim ([15]). Suppose that the adversary alters elements in the mix net such that the observed election tally differs by k votes from the honest one. Then the probability that the adversary goes undetected is $\leq (\frac{1}{2})^k$.

This claim was disproven subsequently. Khazaei and Wikström [17] as well as Küsters, Truderung, and Vogt [20] discovered attacks on original

[4] This could happen if the links of an input ciphertext need to be opened for each mix server. In this case, the sender's message privacy would be broken.

RPC which allow for manipulating k messages in such a way that the tampering remains undetected with probability $(\frac{3}{4})^k$.

In this work, we describe for the first time how to do RPC protocol such that optimal verifiability tolerance $(\frac{1}{2})^k$ is achieved. Our RPC protocol solves the vulnerabilities of the original RPC protocol that allow for the attacks mentioned above. In this way, not only these specific but all possible attacks are prevented that would go undetected with probability $> (\frac{1}{2})^k$. To illustrate our solution, we recall the two attacks with full technical details in what follows.

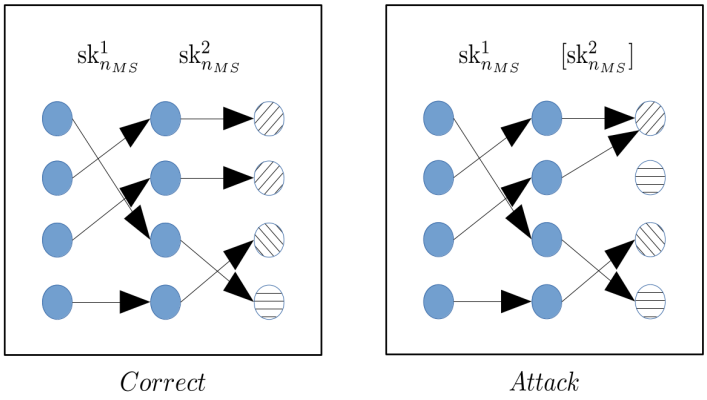

Fig. 3. *Cheating of the last mix server*: The left box shows the correct execution of mix server $M_{n_{\mathsf{MS}}}$. The right box shows an attack of $M_{n_{\mathsf{MS}}}$ where the first two output messages are identical but the second one is replaced by a different output message. The remaining output message is linked to both ciphertexts of the identical output messages. By this, the second output message is effectively dropped. The attack is detected if and only if $\boldsymbol{\alpha}_{n_{\mathsf{MS}}} = (1, 1, \star, \star)$.

Cheating of the last mix server. This attack by [20] is illustrated in Fig. 3. Recall that the final outcome $\boldsymbol{c}^2_{n_{\mathsf{MS}}}$ is returned by the last mix server $M_{n_{\mathsf{MS}}}$. Assume that the adversary controls the last mix server $M_{n_{\mathsf{MS}}}$ and, say, favors candidate A over candidate B. If there are two distinct ciphertexts $\boldsymbol{c}^1_{n_{\mathsf{MS}}}[i], \boldsymbol{c}^1_{n_{\mathsf{MS}}}[i']$ in the intermediate ciphertext vector $\boldsymbol{c}^1_{n_{\mathsf{MS}}}$ which both decrypt to candidate B under $\mathsf{sk}^2_{n_{\mathsf{MS}}}$, then the malicious mix server $M_{n_{\mathsf{MS}}}$ replaces one of them by candidate B in its output $\boldsymbol{c}^2_{n_{\mathsf{MS}}}$. If the mix server is supposed to open the right link of $\boldsymbol{c}^1_{n_{\mathsf{MS}}}[i]$ or $\boldsymbol{c}^1_{n_{\mathsf{MS}}}[i']$, then it opens the link to B in both cases. Effectively, $M_{n_{\mathsf{MS}}}$'s manipulation is detected if

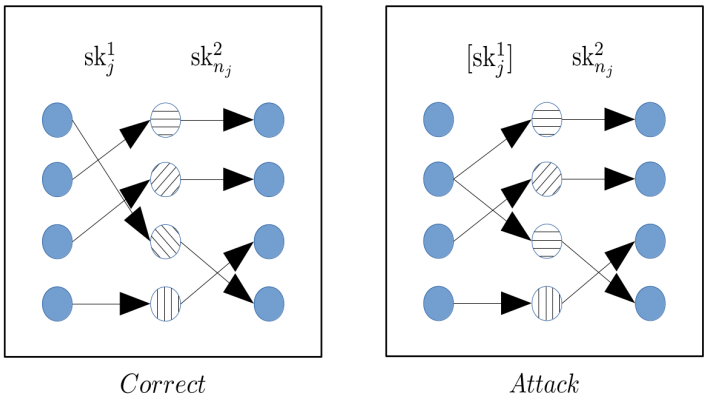

Fig. 4. *Cheating of an arbitrary mix server*: The left box shows the correct execution of mix server M_j. The right box shows an attack of M_j where the first intermediate ciphertext is replaced by a duplicate of the third intermediate ciphertext. In the cleaning phase of the next mix server M_{j+1}, one of these two duplicates will be removed. By this, the message contained in the first intermediate ciphertext vector is effectively dropped. The attack is detected if and only if $\boldsymbol{\alpha}_j = (-1, \star, -1, \star)$.

and only if the right links of both $\boldsymbol{c}^1_{n_{\mathsf{MS}}}[i]$ *and* $\boldsymbol{c}^1_{n_{\mathsf{MS}}}[i']$ are to be opened. The probability of this event is $\frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$. Hence, the attack remains undetected with probability $\frac{3}{4}$.

Cheating of an arbitrary mix server. This attack by [17] is illustrated in Fig. 4. Assume that the adversary controls an arbitrary mix server M_j. Let $\boldsymbol{c}^0_j[i]$ and $\boldsymbol{c}^0_j[i']$ be two arbitrary elements of M_j's input vector $\boldsymbol{c}^0_j$. Assume that $\boldsymbol{c}^0_j[i]$ decrypts to $\tilde{c}^1$ under sk^1_j and that $\tilde{c}^1$ decrypts to $\tilde{c}^2$ under sk^2_j. Assume that $\boldsymbol{c}^0_j[i']$ decrypts to c^1 under sk^1_j and that c^1 decrypts to c^2 under sk^2_j. Now, the malicious mix server M_j replaces $\tilde{c}^1$ by c^1 in its intermediate ciphertext vector $\boldsymbol{c}^1_j$, and $\tilde{c}^2$ by c^2 in its output ciphertext vector $\boldsymbol{c}^2_j$. In this way, the choice in $\tilde{c}^0$ is effectively dropped. The choice in c^0 is temporarily copied but one of these copies will be removed again due to the duplicate removal of the next mix server M_{j+1}. If M_j is supposed to open the left link of one of the two identical intermediate ciphertexts $c^1 \in \boldsymbol{c}^1_j$, then in both cases it opens the link to c^0. By this, M_j's manipulation is detected if and only if the left links of both copies of c^1 are to be opened. The probability of this event is $\frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$. Hence, the attack remains undetected with probability $\frac{3}{4}$.

6 Optimal RPC

We propose an RPC protocol for decryption mix nets which achieves optimal verifiability tolerance $(\frac{1}{2})^k$. We first explain the general idea of our solution (Sec. 6.1) and then describe the optimal RPC protocol with full technical details (Sec. 6.2).

6.1 Idea

Recall that the attacks described in Sec. 5 exploit the following two properties of the original RPC protocol:

1. It is possible that there exist duplicate plaintext messages in the final outcome $\boldsymbol{c}^2_{n_{\mathsf{MS}}}$.
2. If a duplicate ciphertext appears during mixing, then this ciphertext is removed but it is not necessarily checked whether it was injected by a malicious mix server.

Based on these observations, we designed a new RPC protocol which extends the original RPC protocol as follows:

1. *Adding innermost encryption layer*:[5] The auditor A creates a public/secret key pair $(\mathsf{pk}_A, \mathsf{sk}_A)$. Each sender S encrypts her message m first under the auditor's public key pk_A and afterwards under the mix servers' public keys. By this, the mixing is performed over encrypted rather than plain messages. Due to the IND-CCA2-security of the PKE scheme, the probability that there exist (honestly generated) duplicate entries in the outcome of the mixing phase $\boldsymbol{c}^2_{n_{\mathsf{MS}}}$ is negligible. Once the auditing phase succeeded, the auditor reveals its secret key sk_A so that all ciphertexts in $\boldsymbol{c}^2_{n_{\mathsf{MS}}}$ can be decrypted (publicly).[6]
2. *Opening duplicate links*: If during the mixing of an arbitrary mix server M_j, a duplicate ciphertext appears, then M_j explicitly opens the complete local trace of the two (or more) identical ciphertexts through its mix. All duplicate ciphertexts are removed and not taken into account for the RPC in the auditing phase. In this way, we ensure that, in contrast to the original RPC protocol, the (deterministic) mixing function is bijective. As a result, for each intermediate ciphertext $c^1 \in \boldsymbol{c}^1_j$,

[5] Even though this idea was already mentioned in prior work [20], it was dismissed because it does not improve verifiability/accountability by itself.

[6] Not releasing the secret key until after auditing provides an extra degree of privacy protection if any mixer server was dishonest but the secrecy of the key is not required for integrity.

there exists exactly one correct link to an element in $\boldsymbol{c}_j^0$ and exactly one correct link to an element in $\boldsymbol{c}_j^2$.

Due to these two extensions, the resulting RPC decryption mix net achieves optimal verifiability tolerance without affecting privacy.

Impact on verifiability. Our formal verifiability analysis (Sec. A) will demonstrate that these two modifications in combination protect against *all possible* attacks for manipulating k messages that would remain undetected with probability $> (\frac{1}{2})^k$. While it is easy to see that the two attacks described in Sec. 5 are prevented by the two modifications above, proving that this holds true for all possible attacks of the same kind is more challenging (see App. A.4).

Impact on privacy. At a first sight, one may think that opening the local links of *all* identical ciphertexts undermines privacy of honest senders. That is, in collaboration with a dishonest mix server, a dishonest sender could simply duplicate an honest sender's intermediate ciphertext. In this way, the honest sender's local link would be revealed. However, observe that for privacy to be guaranteed, all mix nets assume that at least one mix server is honest. Due to the IND-CCA2-security of the underlying PKE scheme, a dishonest sender can only duplicate honest ciphertexts *outside* the honest mix server's encryption layer. In such a case, the local link of an honest sender's ciphertext trace may only be revealed *prior* to the honest mixing phase. This argument demonstrates that privacy of the original RPC protocol is preserved.

6.2 Protocol

We describe the optimal RPC protocol with full technical details.

Remark. Due to the IND-CCA2-security of the underlying public-key encryption scheme, permuting the decrypted ciphertexts uniformly at random is equivalent from a privacy perspective to sorting them lexicographically. Unlike the original RPC protocol, we chose the latter version because it makes commitments dispensable. This will simplify the protocol description without affecting security.

Parameters and algorithms. We use the following parameters and algorithms:

- $p \in (0, 1]$: probability for opening either a left or right link (as opposed to opening neither of them).
- $\lambda > 1$: security parameter.
- Algorithm App: Takes as input a vector $\boldsymbol{c}$ and element c. Appends element c to vector $\boldsymbol{c}$.
- Algorithm Ins: Takes as input a lexicographically sorted vector $\boldsymbol{c}$ and element c. Inserts element c into vector $\boldsymbol{c}$ according to its lexicographic position.

Cryptographic primitives. We use the following cryptographic primitives:

- An IND-CCA2-secure public-key encryption scheme $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$.
- A NIZKP proof of correct decryption $(\mathsf{Prove}, \mathsf{Verify})$ for $\mathcal{E}$. The underlying relation is

$$R = \{((c, m, \mathsf{pk}), \mathsf{sk}) \colon m = \mathsf{Dec}(\mathsf{sk}, c) \wedge (\exists r \colon (\mathsf{pk}, \mathsf{sk}) = \mathsf{KeyGen}(r))\}.$$

To instantiate these primitives, one can combine for example the IND-CCA2-secure PKE by Cramer-Shoup [10] with the NIZKP by Camenisch-Shoup [4].

Protocol participants. The protocol is run among the following participants:

- Bulletin board B (append-only).
- Senders $S_1, \ldots, S_{n_\mathsf{S}}$.
- Mix servers $M_1, \ldots, M_{n_\mathsf{MS}}$.
- Auditor A.[7]

We assume that there exist mutually authenticated channels between the bulletin board B and all other participants.

Setup phase. Each mix server M_j creates two public/secret key pairs as follows:

1. $(\mathsf{pk}_j^1, \mathsf{sk}_j^1) \leftarrow \mathsf{KeyGen}(1^\lambda)$

[7] The role of the auditor can easily be distributed. For example, each auditor could first commit to its randomness (using a non-malleable commitment scheme), and once all auditors have published their commitments, they open them and combine the results using XOR. For the sake of simplicity, we consider a single auditor only.

2. $(\mathsf{pk}_j^2, \mathsf{sk}_j^2) \leftarrow \mathsf{KeyGen}(1^\lambda)$
3. Send $(\mathsf{pk}_j^1, \mathsf{pk}_j^2)$ to B

The auditor A creates a public/secret key pair as well:

1. $(\mathsf{pk}_A, \mathsf{sk}_A) \leftarrow \mathsf{KeyGen}(1^\lambda)$
2. Send pk_A to B

Submission phase. Each sender S takes as input the mix servers' public keys $(\mathsf{pk}_j^1, \mathsf{pk}_j^2)_{j=1}^{n_{\mathsf{MS}}}$ as well as the auditor's public key pk_A and iteratively encrypts m as follows

1. $c_{n_{\mathsf{MS}}}^2 \leftarrow \mathsf{Enc}(\mathsf{pk}_A, m)$
2. for $j = n_{\mathsf{MS}}$ to 1:
   (a) $c_j^1 \leftarrow \mathsf{Enc}(\mathsf{pk}_j^2, c_j^2)$
   (b) $c_{j-1}^2 \leftarrow \mathsf{Enc}(\mathsf{pk}_j^1, c_j^1)$
3. Send c_0^2 to B

We denote the (initially empty) vector of input ciphertexts by $\boldsymbol{c}_0^2$. For each incoming ciphertext c_0^2 from some sender S, the bulletin board B performs the following steps to ensure that S can neither submit multiple nor duplicated inputs:

1. if S already submitted $c \in \boldsymbol{c}_0^2$, then abort
2. elseif $c_0^2 \in \boldsymbol{c}_0^2$, then abort
3. else $\boldsymbol{c}_0^2 \leftarrow \mathsf{App}(\boldsymbol{c}_0^2, c_0^2)$

The ciphertext vector $\boldsymbol{c}_0^2$ is the senders' joint input to the subsequent mixing phase.

Mixing phase. Starting with M_1, each mix server M_j takes $\boldsymbol{c}_{j-1}^2$ as input and decrypts these input ciphertexts first under sk_j^1 and then under sk_j^2. The main idea of the optimal RPC protocol is the following one: If, during the mixing process, a mix server M_j decrypts a ciphertext to a duplicate message, then the mix server explicitly opens the complete local links (i.e., the left and right side) of both identical messages. These opened links are stored in $\boldsymbol{d}_j$ and the duplicate message is discarded. Similarly, the mix server also opens the links of all invalid messages, stores the opened links in $\boldsymbol{i}_j$, and discards the invalid message.

In what follows, $\boldsymbol{c}_j^0, \boldsymbol{c}_j^1, \boldsymbol{c}_j^2$ denote the (initially empty) ciphertext vectors, $\boldsymbol{p}_j$ denotes the (initially empty) list of local traces, $\boldsymbol{i}_j$ denotes the (initially empty) list of invalid messages, and $\boldsymbol{d}_j$ denotes the (initially empty) list of duplicate messages. Now, each mix server M_j executes the following steps for all $c^0 \in \boldsymbol{c}_{j-1}^2$:

1. *Decrypt*:
   (a) $c^1 \leftarrow \mathsf{Dec}(\mathsf{sk}_j^1, c^0)$
   (b) $c^2 \leftarrow \mathsf{Dec}(\mathsf{sk}_j^2, c^1)$
2. *Open invalids*: if $c^1 = \perp$ or $c^2 = \perp$ then
   (a) $\pi^1 \leftarrow \mathsf{Prove}((\mathsf{pk}_j^1, \mathsf{sk}_j^1), c^0, c^1)$
   (b) $\pi^2 \leftarrow \mathsf{Prove}((\mathsf{pk}_j^2, \mathsf{sk}_j^2), c^1, c^2)$
   (c) $\boldsymbol{i}_j \leftarrow \mathsf{App}(\boldsymbol{i}_j, (c^0, c^1, c^2, \pi^1, \pi^2))$
3. *Open intermediate duplicates*: elseif $c^1 \in \boldsymbol{c}_j^1$ then
   (a) $\pi^1 \leftarrow \mathsf{Prove}((\mathsf{pk}_j^1, \mathsf{sk}_j^1), c^0, c^1)$
   (b) $\pi^2 \leftarrow \mathsf{Prove}((\mathsf{pk}_j^2, \mathsf{sk}_j^2), c^1, c^2)$
   (c) $\tilde{c}^0, \tilde{c}^2$ s.t. $(\tilde{c}^0, c^1, \tilde{c}^2) \in \boldsymbol{p}_j$
   (d) $\tilde{\pi}^1 \leftarrow \mathsf{Prove}((\mathsf{pk}_j^1, \mathsf{sk}_j^1), \tilde{c}^0, c^1)$
   (e) $\tilde{\pi}^2 \leftarrow \mathsf{Prove}((\mathsf{pk}_j^2, \mathsf{sk}_j^2), c^1, \tilde{c}^2)$
   (f) $\boldsymbol{d}_j \leftarrow \mathsf{App}(\boldsymbol{d}_j, ((c^0, c^1, c^2, \pi^1, \pi^2), (\tilde{c}^0, c^1, \tilde{c}^2, \tilde{\pi}^1, \tilde{\pi}^2)))$
4. *Open outcome duplicates*: elseif $c^2 \in \boldsymbol{c}_j^2$ then
   (a) $\pi^1 \leftarrow \mathsf{Prove}((\mathsf{pk}_j^1, \mathsf{sk}_j^1), c^0, c^1)$
   (b) $\pi^2 \leftarrow \mathsf{Prove}((\mathsf{pk}_j^2, \mathsf{sk}_j^2), c^1, c^2)$
   (c) $\tilde{c}^0, \tilde{c}^1$ s.t. $(\tilde{c}^0, \tilde{c}^1, c^2) \in \boldsymbol{p}_j$
   (d) $\tilde{\pi}^1 \leftarrow \mathsf{Prove}((\mathsf{pk}_j^1, \mathsf{sk}_j^1), \tilde{c}^0, \tilde{c}^1)$
   (e) $\tilde{\pi}^2 \leftarrow \mathsf{Prove}((\mathsf{pk}_j^2, \mathsf{sk}_j^2), \tilde{c}^1, c^2)$
   (f) $\boldsymbol{d}_j \leftarrow \mathsf{App}(\boldsymbol{d}_j, ((c^0, c^1, c^2, \pi^1, \pi^2), (\tilde{c}^0, \tilde{c}^1, c^2, \tilde{\pi}^1, \tilde{\pi}^2)))$
5. *Store links and insert*: else
   (a) $\boldsymbol{p}_j \leftarrow \mathsf{App}(\boldsymbol{p}_j, (c^0, c^1, c^2))$
   (b) $\boldsymbol{c}_j^0 \leftarrow \mathsf{Ins}(\boldsymbol{c}_j^0, c^0)$
   (c) $\boldsymbol{c}_j^1 \leftarrow \mathsf{Ins}(\boldsymbol{c}_j^1, c^1)$
   (d) $\boldsymbol{c}_j^2 \leftarrow \mathsf{Ins}(\boldsymbol{c}_j^2, c^2)$

Eventually, M_j sends $(\boldsymbol{c}_j^0, \boldsymbol{c}_j^1, \boldsymbol{c}_j^2, \boldsymbol{i}_j, \boldsymbol{d}_j)$ to B. Observe that, in contrast to the original RPC protocol, the elements in the final mix server's ciphertext vector $\boldsymbol{c}_{n_{\mathsf{MS}}}^2$ are still encrypted under the auditor's public key pk_A. If all checks in the subsequent auditing phase are successful, then A publishes its secret key sk_A on the bulletin board so that $\boldsymbol{c}_{n_{\mathsf{MS}}}^2$ can be decrypted publicly.

Auditing phase. For each mix server M_j, the auditor A checks whether invalid and duplicate messages were correctly discarded and whether the respective links were opened correctly. Precisely, A runs the following program:

1. *Initialize*:

(a) $b \leftarrow 1$

2. *Verify*:
   (a) *Consistency*:
      i. if $\neg(|\boldsymbol{c}_j^0| = |\boldsymbol{c}_j^1| = |\boldsymbol{c}_j^2|)$, then $b \leftarrow 0$
      ii. if $\boldsymbol{c}_{j-1}^2 \neq \boldsymbol{c}_j^0 \cup (c^0)_{(c^0,\ldots)\in \boldsymbol{i}_j} \cup (c^0)_{((c^0,\ldots),\ldots)\in \boldsymbol{d}_j}$ as multisets, then $b \leftarrow 0$
   (b) *Invalids removal*:
      i. if $\bot \in \boldsymbol{c}_j^1$, then $b \leftarrow 0$
      ii. if $\bot \in \boldsymbol{c}_j^2$, then $b \leftarrow 0$
   (c) *Duplicate removal*:
      i. if $\exists i \neq i' \colon \boldsymbol{c}_j^1[i] = \boldsymbol{c}_j^1[i']$, then $b \leftarrow 0$
      ii. if $\exists i \neq i' \colon \boldsymbol{c}_j^2[i] = \boldsymbol{c}_j^2[i']$, then $b \leftarrow 0$
   (d) *Invalids links*: for all $(c^0, c^1, c^2, \pi^1, \pi^2) \in \boldsymbol{i}_j$:
      i. if $c^1 \neq \bot$ and $c^2 \neq \bot$, then $b \leftarrow 0$
      ii. if $\mathsf{Verify}(\mathsf{pk}_j^1, c^0, c^1, \pi^1) = 0$, then $b \leftarrow 0$
      iii. if $\mathsf{Verify}(\mathsf{pk}_j^2, c^1, c^2, \pi^2) = 0$, then $b \leftarrow 0$
   (e) *Duplicate links*: for all $((c^0, c^1, c^2, \pi^1, \pi^2), (\tilde{c}^0, \tilde{c}^1, \tilde{c}^2, \tilde{\pi}^1, \tilde{\pi}^2)) \in \boldsymbol{d}_j$:
      i. if $c^1 \notin \boldsymbol{c}_j^1$ and $c^2 \notin \boldsymbol{c}_j^2$, then $b \leftarrow 0$
      ii. if $c^1 \neq \tilde{c}^1$ and $c^2 \neq \tilde{c}^2$, then $b \leftarrow 0$
      iii. if $\mathsf{Verify}(\mathsf{pk}_j^1, c^0, c^1, \pi^1) = 0$, then $b \leftarrow 0$
      iv. if $\mathsf{Verify}(\mathsf{pk}_j^2, c^1, c^2, \pi^2) = 0$, then $b \leftarrow 0$
      v. if $\mathsf{Verify}(\mathsf{pk}_j^1, \tilde{c}^0, \tilde{c}^1, \tilde{\pi}^1) = 0$, then $b \leftarrow 0$
      vi. if $\mathsf{Verify}(\mathsf{pk}_j^2, \tilde{c}^1, \tilde{c}^2, \tilde{\pi}^2) = 0$, then $b \leftarrow 0$
3. Return b

The remaining part of the auditing phase (i.e., generating the challenges $\boldsymbol{\alpha}_j$, creating the proofs $\boldsymbol{\pi}_j$, and verifying them) works as in the original RPC. If all of these checks are successful, then A publishes its secret key sk_A on the bulletin board so that the mix net's outcome ciphertext vector $\boldsymbol{c}_{n_{\mathsf{MS}}}^2$ can be decrypted publicly. Otherwise, if one of the previously described checks fails, then the auditor outputs $\mathsf{dis}(M_j)$ to state that M_j misbehaved.

7 Formal Verifiability Analysis

We formally analyze verifiability of the optimal RPC protocol (Sec. 6) using the same generic verifiability framework [19] that was previously applied by Küsters, Truderung, and Vogt [20] to analyze the original RPC protocol (Sec. 4). We summarize our formal result in what follows and refer to App. A for full details and our formal proof.

Assumptions. We make the following assumptions:

(V1) The public-key encryption scheme $\mathcal{E}$ is IND-CPA-secure.[8]
(V2) $(\mathsf{Prove}, \mathsf{Verify})$ is a non-interactive proof (NIP) of correct decryption.[9]
(V3) The bulletin board B and the auditor are honest.

Note that these assumptions are the same as for the original RPC protocol.

Result. Under the assumptions above, we obtain the following verifiability result for the optimal RPC protocol. We refer to Theorem 2 (App. A) for the completely formal statement.

Theorem 1 (Verifiability (informal)). *Under the assumptions (V1) to (V3) stated above, the probability that in a run of the optimal RPC protocol with verification probability p more than k inputs of honest senders have been manipulated but the auditing procedure is nevertheless successfull is bounded by* $(1 - \frac{p}{2})^{k+1}$.

8 Conclusion

We proposed a new RPC protocol for decryption mix nets. We proved that our new version improves the verifiability level of the original RPC protocol from $(\frac{3}{4})^k$ down to $(\frac{1}{2})^k$ which is optimal for RPC. By this, we improve the "Achilles heel" of RPC, i.e., the range of small values of k, where some cheating may remain undetected with non-negligible probability. Current implementations of RPC for decryption mix nets, in particular for real-world secure e-voting, should adopt our changes to improve their security.

Acknowledgements

Thomas Haines was supported by Research Council of Norway and the Luxembourg National Research Fund (FNR), under the joint INTER project SURCVS (INTER/RCN/17/11747298/SURCVS/Ryan). Johannes Mueller was supported by the Luxembourg National Research Fund (FNR), under the CORE Junior project FP2 (C20/IS/14698166/FP2/Mueller).

[8] For verifiability/accountability, IND-CPA-security is sufficient. For privacy, we need the stronger notion of IND-CCA-security.
[9] The zero-knowledge property is necessary for privacy but not for verifiability/accountability.

References

1. S. Bayer and J. Groth. Efficient Zero-Knowledge Argument for Correctness of a Shuffle. In D. Pointcheval and T. Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 7237 of *Lecture Notes in Computer Science*, pages 263–280. Springer, 2012.
2. X. Boyen, T. Haines, and J. Müller. A Verifiable and Practical Lattice-Based Decryption Mix Net with External Auditing. In L. Chen, N. Li, K. Liang, and S. A. Schneider, editors, *Computer Security - ESORICS 2020 - 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14-18, 2020, Proceedings, Part II*, volume 12309 of *Lecture Notes in Computer Science*, pages 336–356. Springer, 2020.
3. C. Burton, C. Culnane, J. Heather, T. Peacock, P. Y. Ryan, S. Schneider, S. Srinivasan, V. Teague, R. Wen, and Z. Xia. Using Prêt à Voter in Victorian State elections. In *Proc. USENIX EVT/WoTE*, 2012.
4. J. Camenisch and V. Shoup. Practical Verifiable Encryption and Decryption of Discrete Logarithms. In *CRYPTO 2003, Proceedings*, volume 2729 of *LNCS*, pages 126–144. Springer, 2003.
5. R. Carback, D. Chaum, J. Clark, J. Conway, A. Essex, P. S. Herrnson, T. Mayberry, S. Popoveniuc, R. L. Rivest, E. Shen, A. T. Sherman, and P. L. Vora. Scantegrity II Municipal Election at Takoma Park: The First E2E Binding Governmental Election with Ballot Privacy. In *19th USENIX Security Symposium, Washington, DC, USA, August 11-13, 2010, Proceedings*, pages 291–306. USENIX Association, 2010.
6. D. Chaum. Untraceable Mail, Return Addresses and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.
7. D. Chaum, P. Y. A. Ryan, and S. A. Schneider. A Practical Voter-Verifiable Election Scheme. In S. D. C. di Vimercati, P. F. Syverson, and D. Gollmann, editors, *ESORICS*, volume 3679 of *Lecture Notes in Computer Science*, pages 118–139. Springer, 2005.
8. M. R. Clarkson, S. Chong, and A. C. Myers. Civitas: Toward a Secure Voting System. In *2008 IEEE Symposium on Security and Privacy (S&P 2008), 18-21 May 2008, Oakland, California, USA*, pages 354–368. IEEE Computer Society, 2008.
9. V. Cortier, D. Galindo, R. Küsters, J. Müller, and T. Truderung. SoK: Verifiability Notions for E-Voting Protocols. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, pages 779–798. IEEE Computer Society, 2016.
10. R. Cramer and V. Shoup. A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer, 1998.
11. P. Fauzi, H. Lipmaa, J. Siim, and M. Zajac. An Efficient Pairing-Based Shuffle Argument. In *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II*, pages 97–127, 2017.
12. T. Haines, S. J. Lewis, O. Pereira, and V. Teague. How Not to Prove Your Election Outcome. In *2020 IEEE SP 2020*, pages 644–660. IEEE, 2020.
13. T. Haines and J. Müller. SoK: Techniques for Verifiable Mix Nets. In *33rd IEEE Computer Security Foundations Symposium, CSF 2020, Boston, MA, USA, June 22-26, 2020*, pages 49–64. IEEE, 2020.

14. C. Hébant, D. H. Phan, and D. Pointcheval. Linearly-Homomorphic Signatures and Scalable Mix-Nets. In *Public-Key Cryptography - PKC 2020 - International Conference on Practice and Theory in Public-Key Cryptography, 2020. Proceedings.*, 2020.
15. M. Jakobsson, A. Juels, and R. L. Rivest. Making Mix Nets Robust for Electronic Voting by Randomized Partial Checking. In *USENIX Security Symposium*, pages 339–353. USENIX, 2002.
16. S. Khazaei, T. Moran, and D. Wikström. A Mix-Net from Any CCA2 Secure Cryptosystem. In X. Wang and K. Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 607–625. Springer, 2012.
17. S. Khazaei and D. Wikström. Randomized Partial Checking Revisited. In *CT-RSA*, volume 7779 of *Lecture Notes in Computer Science*, pages 115–128. Springer, 2013.
18. R. Küsters, J. Müller, E. Scapin, and T. Truderung. sElect: A Lightweight Verifiable Remote Voting System. In *IEEE 29th Computer Security Foundations Symposium, CSF 2016, Lisbon, Portugal, June 27 - July 1, 2016*, pages 341–354, 2016.
19. R. Küsters, T. Truderung, and A. Vogt. Accountability: Definition and Relationship to Verifiability. In E. Al-Shaer, A. D. Keromytis, and V. Shmatikov, editors, *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*, pages 526–535. ACM, 2010.
20. R. Küsters, T. Truderung, and A. Vogt. Formal Analysis of Chaumian Mix Nets with Randomized Partial Checking. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 343–358, 2014.
21. B. Schneier. *Applied Cryptography - Protocols, Algorithms, and Source Code in C, 2nd Edition*. Wiley, 1996.
22. D. Wikström. A Sender Verifiable Mix-Net and a New Proof of a Shuffle. In *Advances in Cryptology - ASIACRYPT 2005, 11th International Conference on the Theory and Application of Cryptology and Information Security, Chennai, India, December 4-8, 2005, Proceedings*, pages 273–292, 2005.
23. D. Wikström. A Commitment-Consistent Proof of a Shuffle. In *Information Security and Privacy, 14th Australasian Conference, ACISP 2009, Brisbane, Australia, July 1-3, 2009, Proceedings*, pages 407–421, 2009.

A Formal Verifiability Analysis

In this section, we state and prove our verifiability result of the optimal RPC protocol with full details. In what follows, we describe the computational model of our verifiability analysis (App. A.1), and then recall the verifiability framework that we will use (App. A.2). After that, we will state the formal verifiability result of the optimal RPC protocol (App. A.3) and prove it formally (App. A.4).

A.1 Computational model

We start by formally modeling RPC using a general and established computational framework (see, e.g., [9, 19]) that we use for analyzing verifiability. The computational model introduces the notion of a process which can be used to model protocols. Essentially, a process $\hat{\pi}_P$ modeling some protocol P is a set of interacting ppt Turing machines which capture the honest behavior of protocol participants. The protocol P runs alongside an adversary $\mathcal{A}$, modeled via another process $\pi_{\mathcal{A}}$, which controls the network and may corrupt protocol participants; here we assume static corruption. We write $\pi = (\hat{\pi}_P \| \pi_{\mathcal{A}})$ for the combined process.

In what follows, we explain the computational model for our verifiability analysis (Section A) in more detail.

Process. A *process* is a set of probabilistic polynomial-time interactive Turing machines (ITMs, also called *programs*) which are connected via named tapes (also called *channels*). Two programs with a channel of the same name but opposite directions (input/output) are connected by this channel. A process may have external input/output channels, those that are not connected internally. At any time of a process run, one program is active only. The active program may send a message to another program via a channel. This program then becomes active and after some computation can send a message to another program, and so on. Each process contains a *master program*, which is the first program to be activated and which is activated if the active program did not produce output (and hence, did not activate another program). If the master program is active but does not produce output, a run stops.

We write a process π as $\pi = p_1 \| \cdots \| p_l$, where $p_1, \ldots, p_l$ are programs. If π_1 and π_2 are processes, then $\pi_1 \| \pi_2$ is a process, provided that the processes are connectible: two processes are *connectible* if common external channels, i.e., channels with the same name, have opposite directions (input/output); internal channels are renamed, if necessary. A process π where all programs are given the security parameter 1^ℓ is denoted by $\pi^{(\ell)}$. In the processes we consider, the length of a run is always polynomially bounded in ℓ. Clearly, a run is uniquely determined by the random coins used by the programs in π.

Protocol. A protocol P is modeled via a process, where different participants and components are represented via one ITM each. Typically, a protocol contains a *scheduler* $\mathcal{S}$ as one of its participants which acts as the master program of the protocol process (see below). The task of the

scheduler is to trigger the protocol participants and the adversary in the appropriate order.

The honest programs of the agents of P are typically specified in such a way that the adversary $\mathcal{A}$ can corrupt the programs by sending the special message corrupt. Upon receiving such a message, the agent reveals all or some of its internal state to the adversary and from then on is controlled by the adversary. Some agents, such as the scheduler, will typically not be corruptible, i.e., they would ignore corrupt messages. Also, agents might only accept corrupt messages upon initialization, modeling static corruption. This is the case for our security analysis of RPC.

We say that an agent a is *honest in a protocol run* r if the agent has not been corrupted in this run, i.e., has not accepted a corrupt message throughout the run. We say that an agent a is *honest* if for all adversarial programs $\pi_{\mathcal{A}}$ the agent is honest in all runs of $\hat{\pi}_P \| \pi_{\mathcal{A}}$, i.e., a always ignores all corrupt messages.

Property. A *property* γ *of* P is a subset of the set of all runs of P.[10] By $\neg\gamma$ we denote the complement of γ.

A.2 Verifiability framework

We recall the verifiability framework by Küsters, Truderung, and Vogt [19] that we use to analyze the optimal RPC protocol.

Intuitively, a mix net is verifiable if an incorrect final outcome is accepted only with small probability $\delta \in [0, 1]$.

Judge. To model whether the final outcome of a protocol run should be accepted, the verifiability definition by Küsters, Truderung, and Vogt assumes an additional protocol participant J, called the *judge*. The judge can be thought of as a "virtual" entity; in reality, the program of J can be carried out by any party, including external observers or the senders themselves, since its input is merely public information. On a high level, the judge performs certain checks to ensure the correctness of the final outcome (e.g., verifying all NIZKPs). Typically, the program of J follows immediately from the protocol description. In the case of RPC, the judge performs the auditor's checks. Formally, to either accept or reject a protocol run, the judge writes accept or reject on a dedicated channel.

[10] Recall that the description of a run r of P contains the description of the process $\hat{\pi}_P \| \pi_{\mathcal{A}}$ (and hence, in particular the adversary) from which r originates. Therefore, γ can be formulated independently of a specific adversary.

Goal. To specify which runs are "correct" in some protocol-specific sense, Küsters, Truderung, and Vogt use the notion of a goal γ. Formally, a goal γ is simply a set of protocol runs. For mix nets, γ would contain those runs where the announced mix net result corresponds to the actual messages of the senders.

In what follows, we describe the goal $\gamma(k, \varphi)$ that we use to analyze all the different mix nets. This goal was previously applied to analyze and compare all verifiable mix nets from the literature (see [13] for a systematic overview). The parameter φ is a Boolean formula that describes which protocol participants are assumed to be honest in a run, i.e., not corrupted by the adversary. On a high level, the parameter k denotes the maximum number of messages submitted by the honest senders that the adversary is allowed to manipulate. So, roughly speaking, the goal $\gamma(k, \varphi)$ consists of those runs of a mix net protocol P where either φ is false, or φ holds true and the adversary manipulated at most k messages of honest senders. More formally, the goal $\gamma(k, \varphi)$ is defined as follows.

Definition 1 (Goal $\gamma(k, \varphi)$). *Let P be a mix net protocol with n_{S} senders. Let π be an instance of P, and let r be a run of π. Let $S_1, \ldots, S_{n_{\mathsf{S}}^h}$ be those senders that are honest in r. Let $\boldsymbol{m} = m_1, \ldots, m_{n_{\mathsf{S}}^h}$ be the plaintext inputs of these senders in r. Then, $\gamma(k, \varphi)$ is satisfied in r if either (a) the trust assumption φ does not hold true in r, or if (b) φ holds true in r and there exist messages $\tilde{m}_1, \ldots, \tilde{m}_{n_{\mathsf{S}}}$ (including $\perp$) such that the following conditions hold true:*

- *The multiset $\{\tilde{m}_1, \ldots, \tilde{m}_{n_{\mathsf{S}}}\}$ contains at least $n_{\mathsf{S}}^h - k$ elements of the multiset $\{m_1, \ldots, m_{n_{\mathsf{S}}^h}\}$.*
- *The mix net outcome as published in r (if any) equals to a permutation of $\{\tilde{m}_1, \ldots, \tilde{m}_{n_{\mathsf{S}}}\}$.*

If φ does not hold true in r and no outcome is published in r, then $\gamma(k, \varphi)$ is not satistied in r.

Verifiability. Now, the idea behind the verifiability definition is simple. The judge J should accept a protocol run only if the goal γ is met: as discussed, if we use the goal $\gamma(k, \varphi)$, then this essentially means that the published mix net result corresponds to the actual messages of the senders up to k messages of honest senders. More precisely, the definition requires that the probability (over the set of all protocol runs) that the goal γ is not satisfied but the judge nevertheless accepts the run is δ-bounded.[11]

[11] A function f is δ-*bounded* if, for every $c > 0$, there exists ℓ_0 such that $f(\ell) \leq \delta + \ell^{-c}$ for all $\ell > \ell_0$.

Certainly, $\delta = 0$ would be desirable but this perfect tolerance cannot be achieved by any known verifiable decryption mix net (see [13]). The parameter δ is called the *verifiability tolerance* of the protocol.

To define this notion formally, we denote by $\Pr[\pi^{(\ell)} \mapsto \neg\gamma, (J\colon \mathsf{accept})]$ the probability that π, with security parameter 1^ℓ, produces a run which is not in γ but nevertheless accepted by J. Now, we formally define verifiability as follows.

Definition 2 (Verifiability). *Let P be a protocol with the set of agents Σ. Let $\delta \in [0,1]$ be the tolerance, $J \in \Sigma$ be the judge, and γ be a goal. Then, we say that* the protocol P is (γ, δ)-verifiable by the judge J *if for all adversaries $\pi_{\mathcal{A}}$ and $\pi = (\hat{\pi}_P \| \pi_{\mathcal{A}})$, the probability $\Pr[\pi^{(\ell)} \mapsto \neg\gamma, (J\colon \mathsf{accept})]$ is δ-bounded as a function of ℓ.*

We note that the original definition in [19] also captures *soundness*: if the protocol runs with a benign adversary, which, in particular, would not corrupt parties, then the judge accepts all runs. This kind of soundness can be considered to be a sanity check of the protocol, including the judging procedure, and is typically easy to check. For brevity of presentation, we omit this condition here.

A.3 Formal result

We are now able to precisely state the verifiability property of the optimal RPC protocol according to Definition 2. We prove the verifiability result for the goal $\gamma(k, \varphi)$ (Definition 1).

Assumptions. We make the following assumptions:

(V1) The public-key encryption scheme $\mathcal{E}$ is IND-CPA-secure.[12]
(V2) $(\mathsf{Prove}, \mathsf{Verify})$ is a non-interactive proof (NIP) of correct decryption.[13]
(V3) The bulletin board B, the judge J, and the auditor are honest, i.e., $\varphi = \mathsf{hon}(J) \wedge \mathsf{hon}(B) \wedge \mathsf{hon}(A)$.

Note that these assumptions are the same as for the original RPC protocol.

[12] For verifiability/accountability, IND-CPA-security is sufficient. For privacy, we need the stronger notion of IND-CCA-security.
[13] The zero-knowledge property is necessary for privacy but not for verifiability/accountability.

Result. Let $p \in (0, 1]$ be the probability for opening either a left or right link (as opposed to opening neither of them). Now, intuitively, the following theorem states that the probability that in a run of the optimal RPC protocol with verification probability p more than k inputs of honest senders have been manipulated but the judge J nevertheless accepts the run is bounded by $(1 - \frac{p}{2})^{k+1}$.

Theorem 2 (Verifiability). *Under the assumptions (V1) to (V3) stated above, the decryption mix net protocol with optimal RPC is* $(\gamma(k, \varphi), \delta_k(p))$*-verifiable by the judge* J*, where*

$$\delta_k(p) = \left(1 - \frac{p}{2}\right)^{k+1}.$$

Our proof will demonstrate that, just as the original RPC protocol, our optimal RPC version even provides the stronger notion of *(individual) accountability* [19] but with tighter tolerance $\delta_k(p)$. Individual accountability not only guarantees that the correctness of the mix net's outcome can be verified but also that misbehaving mix servers (if any) can be identified and thus held accountable.

Proof sketch. Our complete formal proof of Theorem 2 is provided in App. A.4. In what follows, we describe its high-level idea.

We first observe that it is not possible to undetectably "drop" an honest sender's input ciphertext by claiming that it was invalid. More precisely, the probability that an honest sender S outputs a ciphertext trace for which $(c_j^0, \ldots) \in \boldsymbol{i}_j$ holds true but the auditor nevertheless does not output $\mathsf{dis}(M_j)$ is negligible. This follows from the correctness of the PKE scheme, the soundness of the NIP scheme, and the fact that honest senders run the (correct) encryption algorithm of the PKE scheme.

Since the ciphertext traces of two distinct honest senders "clash" only with at most negligible probability, it follows that an honest sender's plain input message cannot be "dropped" by claiming that it was a duplicate. More precisely, if an honest sender S outputs a ciphertext trace for which $((\tilde{c}_j^0, \tilde{c}_j^1, \tilde{c}_j^2, \ldots), (c_j^0, c_j^1, c_j^2, \ldots)) \in \boldsymbol{d}_j$ holds true, then the remaining ciphertext $\tilde{c}_j^2$ is part of a dishonest sender's ciphertext trace and $\tilde{c}_j^2$ contains the honest sender's plain input message. This follows (with overwhelming probability) from the IND-CPA-security of the PKE scheme (which guarantees that ciphertexts equal with at most negligible probability), the soundness of the NIP scheme, and the fact that honest senders run the (correct) encryption algorithm of the PKE scheme.

In combination, these two observations yield that all honest senders' plain input messages are contained in the input RPC vector $\boldsymbol{c}_1^0$. Therefore, if an adversary wants to manipulate an honest sender's input message, then the adversary has to manipulate at least one RPC vector $\boldsymbol{c}_j^1$ or $\boldsymbol{c}_j^2$ of a malicious mix server M_j.

However, if the auditor does not output $\mathsf{dis}(M_j)$, then we have that each of the RPC vectors $\boldsymbol{c}_j^0$, $\boldsymbol{c}_j^1$, and $\boldsymbol{c}_j^2$ contains distinct ciphertext elements. Due to the soundness of the NIP, for each intermediate element $c^1 \in \boldsymbol{c}_j^1$, there exists at most one element $c^0 \in \boldsymbol{c}_j^0$ for which M_j can prove that c^0 decrypts to c^1; analogously for each element $c^2 \in \boldsymbol{c}_j^2$. Furthermore, if the auditor does not output $\mathsf{dis}(M_j)$, then $|\boldsymbol{c}_j^0| = |\boldsymbol{c}_j^1|$ holds true. Therefore, if the decryption c^1 of some $c^0 \in \boldsymbol{c}_j^0$ is missing in $\boldsymbol{c}_j^1$, then there exists $\tilde{c}^1 \in \boldsymbol{c}_j^1$ for which there does not exist $\tilde{c}^0 \in \boldsymbol{c}_j^0$ so that M_j can provide a valid proof that $\tilde{c}^0$ decrypts to $\tilde{c}^1$. This remains undetected only if the left link of $\tilde{c}^1$ is not supposed to be opened. The probability of the latter event is bounded $(1 - \frac{p}{2})$, where p is the probability that either the left or right link of a given intermediate ciphertext is to be opened. Analogously for the case that an expected c^2 is missing in $\boldsymbol{c}_j^2$.

Since all elements in $\boldsymbol{c}_j^0$, $\boldsymbol{c}_j^1$, and $\boldsymbol{c}_j^2$ are distinct if the auditor does not output $\mathsf{dis}(M_j)$, we can repeat the above argument independently k times for the case that more than k expected elements are missing in $\boldsymbol{c}_j^1$ or $\boldsymbol{c}_j^2$. The probability of this event is therefore bounded by $(1 - \frac{p}{2})^{k+1}$.

Now, in combination with the observation above that all honest senders' inputs are contained in $\boldsymbol{c}_1^0$, we can conclude that manipulating more than k honest plain input messages remains undetected with probability at most $(1 - \frac{p}{2})^{k+1}$. Furthermore, since the auditor also checks for each mix server M_j whether $|\boldsymbol{c}_j^0| = |\boldsymbol{c}_j^1| = |\boldsymbol{c}_j^2|$ holds true, it is not possible to undetectably "stuff" dishonest input messages. Altogether, we can conclude that the probability of the event that $\gamma(k, \varphi)$ does not hold true in a run r of an arbitrary instance π of the optimal RPC protocol (where assumptions (V1) to (V3) are satisfied) but the auditor nevertheless does not output $\mathsf{dis}(M_j)$ for some M_j is bounded by $(1 - \frac{p}{2})^{k+1}$. This proves Theorem 2.

A.4 Proof

We provide a formal proof of Theorem 2 which establishes the verifiability result of the new RPC protocol described in Sec. 6.2.

We will first show that it is not possible to undetectably "drop" an honest sender's input ciphertext by claiming that it was invalid.

Lemma 1. *Let r be an arbitrary run of π in which the auditor does not output $\mathsf{dis}(M)$. Then there does not exist an honest sender S which outputs ciphertext c^0 such that $(c^0, \ldots) \in \boldsymbol{i}$ holds true (with overwhelming probability).*

Proof. Assume that $\boldsymbol{i} \neq \emptyset$. Let $(c^0, c^1, c^2, \pi^1, \pi^2) \in \boldsymbol{i}$ be an arbitrary entry of $\boldsymbol{i}$.

Since the auditor does not output $\mathsf{dis}(M)$, we have that

- $\mathsf{Verify}(\mathsf{pk}^1, c^0, c^1, \pi^1) = 1$, and
- $\mathsf{Verify}(\mathsf{pk}^1, c^1, c^2, \pi^2) = 1$

hold true.

Due to the soundness of the non-interactive proof, we have that with overwhelming probability

- $c^1 \in \mathsf{Enc}(\mathsf{pk}^1, c^0)$, and
- $c^2 \in \mathsf{Enc}(\mathsf{pk}^2, c^1)$

hold true, and that there exist $\mathsf{sk}^1, \mathsf{sk}^2$ such that

- $(\mathsf{pk}^1, \mathsf{sk}^1) \in \mathsf{KeyGen}(1^\lambda)$, and
- $(\mathsf{pk}^2, \mathsf{sk}^2) \in \mathsf{KeyGen}(1^\lambda)$

hold true as well.

Since the auditor does not output $\mathsf{dis}(M)$, we have that $c^1 = \perp$ or $c^2 = \perp$ hold true.

Now, assume that there exists an honest sender S which output c^0. Then, there exists a message m such that

- $\perp \in \mathsf{Enc}(\mathsf{pk}_A, m)$, or
- $\perp \in \mathsf{Enc}(\mathsf{pk}^2, \mathsf{Enc}(\mathsf{pk}_A, m))$

hold true. This would be a contradiction to the correctness of the public-key encryption scheme since there exist $\mathsf{sk}^2, \mathsf{sk}_A$ such that

- $(\mathsf{pk}^2, \mathsf{sk}^2) \in \mathsf{KeyGen}(1^\lambda)$, and
- $(\mathsf{pk}_A, \mathsf{sk}_A) \in \mathsf{KeyGen}(1^\lambda)$

hold true.

We now observe that the traces of two honest senders "clash" only with at most negligible probability.

Lemma 2. *Let r be an arbitrary run of π. The probability that two honest senders S and $\tilde{S}$ generate ciphertext traces (c^0, c^1, c^2) and $(\tilde{c}^0, \tilde{c}^1, \tilde{c}^2)$ such that*

– $c^0 = \tilde{c}^0$, *or*
– $c^1 = \tilde{c}^1$, *or*
– $c^2 = \tilde{c}^2$

holds true is negligible.

Proof. This follows from the IND-CPA-security of the PKE scheme.

The following result states that an honest sender's plain input message cannot be "dropped" by claiming that it was a duplicate.

Lemma 3. *Let r be an arbitrary run of π in which the auditor does not output $\mathsf{dis}(M)$ and $\boldsymbol{d} \neq \emptyset$. Let $((c^0, c^1, c^2, \pi^1, \pi^2), (\tilde{c}^0, \tilde{c}^1, c^2, \tilde{\pi}^1, \tilde{\pi}^2)) \in \boldsymbol{d}$ be an arbitrary entry of $\boldsymbol{d}$. If there exist an honest sender $\tilde{S}$ which output $\tilde{c}^0$, then there exists a dishonest sender S which output c^0 (with overwhelming probability).*

Proof. Since the auditor does not output $\mathsf{dis}(M)$, we have that

– $\mathsf{Verify}(\mathsf{pk}^1, c^0, c^1, \pi^1) = 1$, and
– $\mathsf{Verify}(\mathsf{pk}^2, c^1, c^2, \pi^2) = 1$, and
– $\mathsf{Verify}(\mathsf{pk}^1, \tilde{c}^0, \tilde{c}^1, \tilde{\pi}^1) = 1$, and
– $\mathsf{Verify}(\mathsf{pk}^2, \tilde{c}^1, \tilde{c}^2, \tilde{\pi}^2) = 1$

hold true.

Due to the soundness of the non-interactive proof, we have that with overwhelming probability

– $c^0 \in \mathsf{Enc}(\mathsf{pk}^1, c^1)$, and
– $c^1 \in \mathsf{Enc}(\mathsf{pk}^2, c^2)$, and
– $\tilde{c}^0 \in \mathsf{Enc}(\mathsf{pk}^1, \tilde{c}^1)$, and
– $\tilde{c}^1 \in \mathsf{Enc}(\mathsf{pk}^2, \tilde{c}^2)$

hold true, and that there exist $\mathsf{sk}^1, \mathsf{sk}^2$ such that

– $(\mathsf{pk}^1, \mathsf{sk}^1) \in \mathsf{KeyGen}(1^\lambda)$, and
– $(\mathsf{pk}^2, \mathsf{sk}^2) \in \mathsf{KeyGen}(1^\lambda)$

hold true as well.

Since the auditor does not output $\mathsf{dis}(M)$, we have that

– $c^1 = \tilde{c}^1$, or
– $c^2 = \tilde{c}^2$

hold true.

Hence, there exist senders S and $\tilde{S}$ with ciphertext traces (c^0, c^1, c^2) and $(\tilde{c}^0, \tilde{c}^1, \tilde{c}^2)$. If $\tilde{S}$ is honest in r, then it follows from Lemma 2 that sender S is dishonest.

The following result states that all honest senders' plain input messages are contained in the input RPC vector $\boldsymbol{c}^0$.

Lemma 4. *Let r be an arbitrary run of π in which the auditor does not output $\mathsf{dis}(M)$. Let $(S_i)_{i \in I_h}$ be the honest senders in r and $(m_i)_{i \in I_h}$ be their plain input messages. Then, for each $i \in I_h$, there exists a unique $c_i^0 \in \boldsymbol{c}^0$ such that*

$$c_i^0 \in \mathsf{Enc}(\mathsf{pk}^2, \mathsf{Enc}(\mathsf{pk}^1, \mathsf{Enc}(\mathsf{pk}_A, m_i)))$$

holds true (with overwhelming probability).

Proof. Since the auditor does not output $\mathsf{dis}(M)$, we have that

$$\boldsymbol{c}_0^2 = \boldsymbol{c}^0 \cup (c^0)_{(c^0,\ldots) \in \boldsymbol{i}} \cup (c^0)_{((c^0,\ldots),\ldots) \in \boldsymbol{d}}$$

as multisets holds true.

If an honest sender's input ciphertext $c^0 \in \boldsymbol{c}_0^2$ is in $\boldsymbol{c}^0$, then the claim is trivial.

If an honest input ciphertext $c^0 \in \boldsymbol{c}_0^2$ (for some plain message m) is not in $\boldsymbol{c}^0$, then it follows from Lemma 1 and Lemma 3 that there exists a unique dishonest sender $\tilde{S}$ with input ciphertext $\tilde{c}^0 \in \boldsymbol{c}_0^2$ such that

- $\tilde{c}^0 \in \boldsymbol{c}^0$, and
- $(\tilde{c}^0, \ldots), (c^0, \ldots)) \in \boldsymbol{d}$

hold true. In this case, the claim follows for $c_i^0 = \tilde{c}^0$.

We now define an honest sender's valid ciphertext trace through the (local) mix as either the one that the sender create itself (in case the sender's input ciphertext is in $\boldsymbol{c}^0$) or the one of the associated dishonest sender (in case the sender's input ciphertext is not in $\boldsymbol{c}^0$).

Definition 3 (Valid traces). *Let r be an arbitrary run of π in which the auditor does not output $\mathsf{dis}(M)$. Let S be an honest sender in r. Let c^0, c^1, c^2 be the ciphertexts created by S during the submission phase for plain input message m. Then, we define the* valid trace of S *through the (local) mix as*

– (c^0, c^1, c^2) *if* $c^0 \in \boldsymbol{c}^0$*, and*
– $(\tilde{c}^0, \tilde{c}^1, \tilde{c}^2)$ *if* $c^0 \notin \boldsymbol{c}^0$ *where* $((\tilde{c}^0, \tilde{c}^1, \tilde{c}^2, \ldots), (c^0, c^1, c^2, \ldots)) \in \boldsymbol{d}$ *as in Lemma 4.*

Due to Lemma 2, each honest sender's valid trace is unique in both cases. Furthermore, we have that in both cases, the final ciphertext of the sender's valid trace decrypts to m *under* sk_A*.*

We now observe that manipulating a single honest sender's valid trace remains undetected with probability at most $(1 - \frac{p}{2})$, where p is the verification probability of the optimal RPC protocol.

Lemma 5. *Let* r *be an arbitrary run of* π *in which the auditor does not output* $\mathsf{dis}(M)$*. Let* S *be an arbitrary honest sender in* r *with valid trace* (c^0, c^1, c^2)*. Then, the probability that* $c^2 \notin \boldsymbol{c}^2$ *holds true is bounded by* $(1 - \frac{p}{2})$*.*

Proof. If $(\ldots, (c^0, c^1, c^2, \ldots)) \in \boldsymbol{d}$, then $c^2 \in \boldsymbol{c}^2$ with overwhelming probability because the auditor does not output $\mathsf{dis}(M)$.

Now, assume that $(\ldots, (c^0, c^1, c^2, \ldots)) \notin \boldsymbol{d}$. We first consider the case $c^1 \notin \boldsymbol{c}^1$:

Since the auditor does not output $\mathsf{dis}(M)$, it follows from the proof of Lemma 1 that there exists sk^1 such that $(\mathsf{pk}^1, \mathsf{sk}^1) \in \mathsf{KeyGen}(1^\lambda)$. We denote by $\boldsymbol{c}^1_{\mathsf{exp}}$ the result of decrypting (and then sorting) all ciphertexts in $\boldsymbol{c}^0$ under sk^1.

Since the auditor does not output $\mathsf{dis}(M)$, we have that $|\boldsymbol{c}^0_j| = |\boldsymbol{c}^1_j|$ holds true. Therefore, there exists $\tilde{c}^1 \in \boldsymbol{c}^1$ such that

– $\tilde{c}^1 \neq c^1$ and
– $\boldsymbol{c}^1 \setminus \tilde{c}^1 = \boldsymbol{c}^1_{\mathsf{exp}} \setminus c^1$

hold true.

Since the auditor does not output $\mathsf{dis}(M)$, we have that

$$\boldsymbol{c}^1_j[i] \neq \boldsymbol{c}^1_j[i']$$

holds true for all $i \neq i'$. Due to the correctness of the PKE scheme $\mathcal{E}$ and the soundness of the NIZKP $(\mathsf{Prove}, \mathsf{Verify})$, there does not exist $\tilde{c}^0 \in \boldsymbol{c}^0$ and proof π^1 such that $\mathsf{Verify}(\mathsf{pk}^1, \tilde{c}^0, \tilde{c}^1, \pi^1) = 1$ holds true.

Hence, the probability of the event that $c^1 \notin \boldsymbol{c}^1$ holds true and the auditor does not output $\mathsf{dis}(M)$ is bounded by the probability that the left link of $\tilde{c}^1$ is not to be opened.

The case $c^2 \notin \boldsymbol{c}^2$ is analogous.

We now state that (valid) traces are unique.

Lemma 6. *Let r be an arbitrary run of π in which the auditor does not output $\mathsf{dis}(M)$. Then, for each $c^0 \in \boldsymbol{c}^0$, there exists at most one $c^1 \in \boldsymbol{c}^1$ and at most one $c^2 \in \boldsymbol{c}^2$ such that there exists proofs π^1, π^2 so that $\mathsf{Verify}(\mathsf{pk}^1, c^0, c^1, \pi^1) = 1$ and $\mathsf{Verify}(\mathsf{pk}^2, c^1, c^2, \pi^2) = 1$ hold true.*

Proof. Since the auditor does not output $\mathsf{dis}(M)$, we have that

- $\forall i \neq i' \colon \boldsymbol{c}^1[i] \neq \boldsymbol{c}^1[i']$, and
- $\forall i \neq i' \colon \boldsymbol{c}^2[i] \neq \boldsymbol{c}^2[i']$

hold true. Due to the correctness of the PKE scheme $\mathcal{E}$ and the soundness of the non-interactive proof $(\mathsf{Prove}, \mathsf{Verify})$, the claim follows.

The following result states that manipulating the valid traces of more than k honest senders remains undetected with probability at most $(1 - \frac{p}{2})^{k+1}$.

Lemma 7. *Let r be an arbitrary run of π in which the auditor does not output $\mathsf{dis}(M)$. Let $(S_i)_{i \in I_h}$ be the honest senders in r with valid traces $(c_i^0, c_i^1, c_i^2)_{i \in I_h}$. Then, the probability that $c_i^2 \notin \boldsymbol{c}^2$ holds true for more than k honest senders S_i is bounded by $(1 - \frac{p}{2})^{k+1}$.*

Proof. This follows from a combination of Lemma 6 and Lemma 5.

From Lemma 7 and the fact that the auditor checks whether

$$|\boldsymbol{c}^0| = |\boldsymbol{c}^1| = |\boldsymbol{c}^2|$$

holds true (which guarantees that no dishonest messages can be "stuffed"), we can conclude that if the goal $\gamma(k, \varphi)$ is violated in a run r of an arbitrary instance π of the optimal RPC protocol (where assumptions (V1) to (V3) are satisfied), then the auditor does not output $\mathsf{dis}(M)$ with probability at most $(1 - \frac{p}{2})^{k+1}$. This proves Theorem 2.