Formal Analysis of Chaumian Mix Nets with Randomized Partial Checking*

Ralf Küsters
University of Trier
kuesters@uni-trier.de

Tomasz Truderung
University of Trier
truderung@uni-trier.de

Andreas Vogt
University of Applied Sciences and Arts, Northwestern Switzerland
andreas.vogt@fhnw.ch

Abstract

Mix nets with randomized partial checking (RPC mix nets) have been introduced by Jakobsson, Juels, and Rivest as particularly simple and efficient verifiable mix nets. These mix nets have been used in several implementations of prominent e-voting systems to provide vote privacy and verifiability. In RPC mix nets, higher efficiency is traded for a lower level of privacy and verifiability. However, these mix nets have never undergone a rigorous formal analysis. Recently, Kahazei and Wikström even pointed out several severe problems in the original proposal and in implementations of RPC mix nets in e-voting systems, both for so-called re-encryption and Chaumian RPC mix nets. While Kahazei and Wikström proposed several fixes, the security status of Chaumian RPC mix nets (with the fixes applied) has been left open; re-encryption RPC mix nets, as they suggest, should not be used at all.

In this paper, we provide the first formal security analysis of Chaumian RPC mix nets. We propose security definitions that allow one to measure the level of privacy and verifiability RPC mix nets offer, and then based on these definitions, carry out a rigorous analysis. Altogether, our results show that these mix nets provide a reasonable level of privacy and verifiability, and that they are still an interesting option for the use in e-voting systems.

1 Introduction

The concept of a mix net has been introduced by Chaum [5] as a tool for achieving anonymity. The main application is in electronic voting, but they have also found applications in other domains, such as multi-party computation, payment systems, and anonymous web browsing.

The mix nets proposed by Chaum, later called *Chaumian mix nets*, consist of a sequence $\mathsf{M}_0,\dots,\mathsf{M}_{m-1}$ of mix servers. Each server generates a public/private key pair (pk_j, sk_j) and publishes the public key pk_j. So-called senders choose plaintexts to be sent through the mix net. In the context of e-voting, the plaintexts might be the candidates chosen by the senders/voters. Every sender encrypts her plaintext, say m, under the public keys of all mix servers in the reverse order, i.e., a sender produces a ciphertext of the form: $\mathsf{Enc}_{pk_0}(\mathsf{Enc}_{pk_1}(\cdots \mathsf{Enc}_{pk_{m-1}}(m)\cdots))$. Now, first M_0 decrypts all ciphertexts and shuffles the results, then M_1 does the same with the ciphertexts received from M_0, and so on. Eventually, the last mix server, M_{m-1}, outputs all plaintexts (again after having shuffled them first). The goal of such a mix net is that it should

*This is a full and revised version of a paper that will appear at S&P 2014 [17].

not be possible to link the output to the input. In the context of voting, this is important for *vote privacy*.

Another common form of a mix net is a so-called *re-encryption mix net* [20]. Here the mix servers generate a single joint public key for a public-key encryption scheme that allows for random re-encryption and distributed verifiable decryption. Senders encrypt their messages with the public key and the mix servers do not decrypt the ciphertexts but only randomly re-encrypt and shuffle them. (Decryption is performed jointly at the very end of the mixing phase.)

In the context of e-voting, it is crucial that potential manipulations are detected. That is, if plaintexts (votes) have been dropped or manipulated, this should be detected. This property is called *verifiability*.

Many constructions have been proposed to obtain verifiable mix nets (see, e.g., [23, 19, 7, 25, 11, 12], some of which have been broken [24]). Most of the constructions are quite complex.

A particularly simple and efficient construction is the one proposed by Jakobsson, Juels, and Rivest [11]. They call the new technique they introduce *randomized partial checking* (RPC), which applies to both Chaumian mix nets and re-encryption mix nets. Roughly speaking, the idea behind RPC is that to check whether or not a mix server cheated, every mix server is supposed to reveal some partial information about the input/output relation. (Which information is to be revealed is randomly chosen and the mix servers should not know it beforehand.) Therefore, a cheating server is caught with some probability. From the design of RPC mix nets it is clear that they do not provide perfect security: there is some non-negligible probability that cheating goes undetected and some partial information about the input/output relation is revealed. As argued in [11], in the context of e-voting the penalties for cheating would be so severe that being caught with some (even small) probability should deter a mix server from cheating. Due to their simplicity and efficiency, RPC mix nets have been used in real implementations of several prominent e-voting systems, including Civitas [6] and Prêt à Voter [22]. Some systems, such as Scantegrity [4], have used a similar technique.

In [13], Kahazei and Wikström have pointed out several severe attacks on RPC mix nets as described in the original work [11] and as implemented in several e-voting systems. They suggest that re-encryption RPC mix nets should not be employed at all, but leave as an open problem to prove or disprove that, with the fixes they suggest, Chaumian RPC mix nets provide sufficient security. Kahazei and Wikström mention that carrying out such a proof and even coming up with useful security notions for privacy and verifiability is challenging, considering that in any case RPC mix nets can provide restricted forms of privacy and verifiability only.

Given the simplicity, efficiency, and importance of Chaumian RPC mix nets, this is an interesting and practically relevant open problem, for which we provide answers in this paper. More specifically, the contributions of this work are as follows.

Contribution of this paper. Based on work by Küsters et al. in [15, 16], we propose security definitions which allow one to precisely measure the level of privacy and verifiability Chaumian RPC mix nets provide. As mentioned before, being able to measure the level of security is crucial for RPC mix nets since they are not perfect. Our definitions should be applicable also to other kinds of mix nets.

Since mix nets are mainly used in the context of e-voting, our notion of privacy corresponds to one that has been used in the context of e-voting before [16]. It focuses on the level of privacy for individual senders/voters and basically requires that for every pair of messages an adversary should not be able to tell which of two messages a sender has sent.

We do not only study verifiability, but a stronger notion: accountability. Verifiability requires that misbehavior should be detectable. Accountability, in addition, requires that

specific misbehaving parties can be blamed. This property, which is expected from RPC mix nets, is important in order to deter parties from misbehaving.

We study Chaumian RPC mix net both w.r.t. in-phase and post-phase auditing. Post-phase auditing means that it is checked at the very end of the mixing phase only whether or not the mix servers behaved correctly. For in-phase auditing, the auditing is done for every mix server immediately after it has produced its output.

In RPC mix nets, manipulation of inputs of honest senders might give adversaries (malicious mix servers) leverage for breaking privacy. But, as mentioned, if a mix server is caught cheating it might face severe penalties. To be able to study this trade-off (the risk of being caught and the information gain), besides general (venturesome) adversaries who do not mind being caught, we also introduce the concept of *risk-avoiding adversaries*, i.e., adversaries who cheat only if there is no risk of being caught.

For our analysis of accountability and privacy of Chaumian RPC mix nets we make standard cryptographic assumptions. We assume the public key encryption scheme to be IND-CCA2-secure [1] and the commitment scheme used in such mix nets to be perfectly hiding and computationally binding, with Pedersen commitments being an example [21]. (However, a computationally hiding scheme would be sufficient.) As usual for Chaumian RPC mixnets, we require that the public key encryption scheme allows for proofs of correct decryption.

In our analysis of accountability, we discovered an attack which does not seem to have been described in the literature before. While one of the most effective attacks on accountability/verifiability, it does not further decrease the overall level of security of RPC mix nets. We prove that altogether Chaumian RPC mix nets have a quite good level of accountability, no matter whether in-phase or post-phase auditing is performed. This proves, in particular, that there are no worse attacks on accountability/verifiability than those already known.

As for the level of privacy, it matters whether post-phase or in-phase auditing is performed and whether adversaries are venturesome or risk-avoiding. In the case of in-phase auditing, our results indicate that the level of privacy is very close to the ideal case (where an adversary only learns the plaintexts of the senders after ideal mixing), surprisingly even for venturesome adversaries that are prepared to be caught cheating for sure. In the case of post-phase auditing, such adversaries can, however, break privacy completely. It is quite unlikely though that adversaries are venturesome, at least in the context of e-voting, given the negative consequences such adversaries would have to face. Also, our results show that if an adversary cheats in order to (even only very slightly) increase his advantage in breaking privacy, he will be caught with probability $\frac{1}{4}$ and this probability grows rapidly if the adversary wants to further increase his chances of breaking privacy.

We note that no rigorous formal analysis of RPC mix nets has been carried out before. In particular, none that provides formal security guarantees for privacy or verifiability/accountability. As mentioned, Kahazei and Wikström [13] point out and discuss attacks. In [8], the authors study the distance between the probability distribution of the permutation links in RPC mix nets and the uniform distribution, however, as also pointed out in [13], this result does not capture privacy or verifiability of RPC mix nets.

Structure of this paper. In Section 2, we describe Chaumian RPC mix nets and present our formal model. Accountability and verifiability for such mix nets are defined in Section 3, with the formal analysis presented in Section 4. Our notion of privacy is introduced in Section 5. The formal analysis of the privacy of Chaumian RPC mix nets is then provided in Section 6. We conclude in Section 7, with full details provided in the appendix.

2 Chaumian RPC Mix Net

In this section, we recall the definition of a Chaumian mix net with randomized partial checking [11], a *Chaumian RPC mix net* (or an *RPC mix net* for short), and then provide a formal model of this protocol.

We focus here on a variant where duplicates are eliminated before every mixing stage. As noted in [13], duplicate elimination is necessary to prevent a serious attack on privacy. We therefore need to fix some details of the procedure of duplicate elimination (see below).

We will consider two variants of the protocol, already mentioned in [11]: i) *in-phase auditing*, a variant where auditing takes place as soon as a mix server produced its output and ii) *post-phase auditing*, where auditing takes place only at then end of the mixing phase, i.e., when the last mix server has output its result. While, as we will see, the two variants do not make a difference for verifiability/accountability, they differ in the level of privacy they provide.

2.1 Description of the Protocol

Set of participants. The set of participants of the protocol consists of a public, append-only *bulletin board* B, *n senders* $\mathsf{S}_1, \ldots \mathsf{S}_n$, *m mix servers* $\mathsf{M}_0, \ldots, \mathsf{M}_{m-1}$, and some number of *auditors*.

The role of the auditors is to provide randomness for the auditing phase. The auditors each output a random bit string (more precisely, they first commit to their random bit strings and later open the commitments). Honest auditors output a bit string chosen uniformly at random. These bit strings are combined to one bit string, say by XOR. So, if at least one auditor is honest, the resulting bit string is chosen uniformly at random. We will indeed assume, both for verifiability/accountability and for privacy, that at least one auditor is honest. We note that sometimes heuristics are implemented by which this assumption can be dropped (see [11]). However, as pointed out in [13], especially in the case of in-phase auditing, this leads to problems.

Typically, pairs of mix servers are audited. For the sake of presentation, it is therefore convenient to assume that one mix server performs two mixing steps. We will consider such mix servers in this paper.

Now, an RPC mix net consists of the following phases: setup, submit, mixing, and auditing, where as mentioned before, auditing might be in-phase. Chaumian RPC mix nets require a public-key encryption scheme and a commitment scheme. The precise assumptions required for these schemes are formulated later in this paper.

Setup phase. In this phase, every mix server M_j, $j \in \{0, \ldots, m-1\}$, invokes the key generation algorithm of a public key encryption scheme in order to generate two pairs of public/private keys. We denote the public keys by pk_{2j} and pk_{2j+1} and the corresponding private keys by sk_{2j} and sk_{2j+1}. The public keys are posted on the bulletin board B. Note that, altogether, the mix servers publish $2m$ public keys, $pk_0, \ldots, pk_{2m-1}$, on B.

Submit phase. In this phase, every (honest) sender S_i chooses her input plaintext m_i and performs the following computation. She first encrypts m_i under pk_{2m-1}, resulting in the ciphertext α^i_{2m-1}. Then, she encrypts α^i_{2m-1} under pk_{2m-2}, resulting in the ciphertext α^i_{2m-2}, and so on. In the last step, α^i_1 is encrypted under pk_0, resulting in the ciphertext α^i_0. This ciphertext is posted by the sender on B as her encrypted input.

Mixing phase. The sequence $C_0 = \alpha^1_0, \ldots, \alpha^n_0$ of the encrypted messages posted by the senders on B is the input to the mixing phase. In what follows, we refer to α^i_0 by $C_0[i]$; similarly for other sequences. These ciphertexts are fetched by the first mix server M_0 which processes

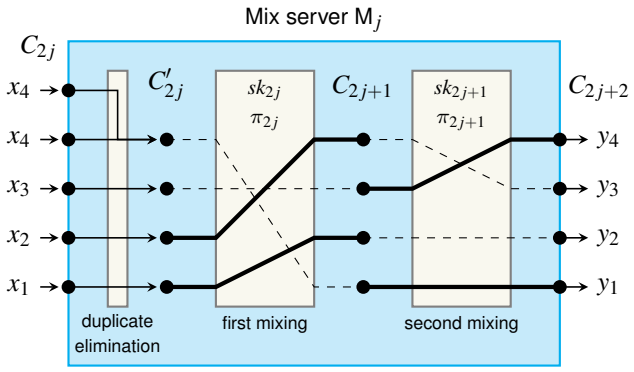

Figure 1: Mixing by M_j. Solid bold lines represent audited links and dashed lines represent not audited links.

them, as described below, and posts its output (which, again, is a sequence of ciphertexts) on B. This output becomes the input to the next mix server M_1, and so on. We will denote the input to the j-th mix server by C_{2j} and its output by C_{2j+2}, reserving C_{2j+1} for intermediate output (see Figure 1). Recall that one mix server performs two mixing steps.

The output C_{2m} of the last mix server M_{m-1} is the output of the protocol. It is supposed to contain the unencrypted input messages $m_1, \ldots, m_n$ (in random order).

The steps taken by every mix server M_j are as follows (see also Figure 1):

1. *Duplicate elimination.* M_j removes all duplicates from its input C_{2j}, leaving only one copy each. The mix server also removes all messages $\perp$, indicating decryption failures, from its input. We denote the resulting sequence by C'_{2j}. In what follows, we denote by $l \leq n$ the number of messages left in C'_{2j}.

2. *First mixing.* M_j uniformly at random chooses a permutation π_{2j} of $\{1, \ldots, l\}$ and posts the sequence C_{2j+1} on B, where $C_{2j+1}[i]$ is the result of the decryption of $C'_{2j}[\pi_{2j}(i)]$ under the private key sk_{2j}. Note that, depending on the encryption scheme, decryption may fail, if the input is not a valid ciphertext. Hence, $C_{2j+1}[i]$ might be $\perp$.

3. *Second mixing.* M_j, again, uniformly at random chooses a permutation π_{2j+1} of $\{1, \ldots, l\}$ and posts the sequence C_{2j+2} on B, where $C_{2j+2}[i]$ is the result of the decryption of $C_{2j+1}[\pi_{2j+1}(i)]$ under the private sk_{2j+1}. The sequence C_{2j+2} is posted by M_j on B. It is the input to the next mix server.

4. *Posting commitments.* M_j posts two sequences of commitments on B: commitments to the values $\pi_{2j}(1), \ldots, \pi_{2j}(l)$ and commitments to the values $\pi^{-1}_{2j+1}(1), \ldots, \pi^{-1}_{2j+1}(l)$ (in this order).

We note that the duplicate elimination is performed only on the input C_{2j} to the mix server, not on the intermediate sequence C_{2j+1}. This simplifies the auditing.

Auditing phase. The outputs of the mix servers are (partially) audited in order to detect potential misbehavior. As mentioned before, we consider in-phase and post-phase auditing. In-phase auditing is performed for every mix server M_j immediately after it has posted its output C_{2j+2} and the commitments on B. In the case that misbehavior is detected, the output of the malicious mix server is not forwarded to the next server and the mixing process is stopped altogether. Conversely, post-phase auditing is performed only when the mixing phase

is finished. However, the steps taken for every individual mix server are the same for both types of auditing. We now describe the auditing for the mix server M_j.

First, using the randomness produced by the auditors, for an initial empty set I_j and for every $i \in \{1,\ldots,l\}$ it is randomly decided, independently of other elements, whether i is added to $I_j \subseteq \{1,\ldots,l\}$ or not. Provided that the random bit strings jointly produced by the auditors are distributed uniformly at random, the probably that i belongs to I_j is $\frac{1}{2}$.

Now, for every $i \in \{1,\ldots,l\}$ the mix server M_j does the following, depending on whether i belongs to I_j or not:

If $i \in I_j$, then the mix server M_j is supposed to open (by posting appropriate information on B) 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 $\pi_{2j}(i)$. The mix server also has to post a (non-interactive zero-knowledge) proof demonstrating that indeed $C_{2j+1}[i]$ is obtained by decrypting $C'_{2j}[\pi_{2j}(i)]$ using sk_{2j}.

If $i \notin I_j$, then, symmetrically, the mix server is supposed to open the right link for i, i.e., M_j is supposed to open its i-th commitment from its second sequence of commitments, which should be a commitment on the value $\pi^{-1}_{2j+1}(i)$. As before, the mix server also has to post a proof that allows an observer to verify that indeed $C_{2j+2}[\pi^{-1}_{2j+1}(i)]$ is obtained by decrypting $C_{2j+1}[i]$ using sk_{2j+1}.

An observer (or a judge) can now verify correctness of the data output by M_j in the audit phase. First, the observer verifies that indeed all duplicates have been removed from the input (by checking whether the number of messages output by M_j is as expected). Second, one verifies that commitments are opened correctly. Third, one verifies that the opened indices (both from the first and the second sequence) do not contain duplicates (if they do, this means that the mix server has not committed to a permutation, but to some other, non-bijective function). Finally, one verifies the decryption proofs. As pointed out in [13], the third step, which often has been omitted in implementations and is not mentioned in [11], is crucial for verifiability and privacy.

The auditing described above guarantees that for a message from the sequence C_{2j+1} either the connection to some message from C_{2j} or to some message from C_{2j+2} is revealed, but never both. Otherwise, an observer could follow the path of an input message to the corresponding output message (see also Figure 1 for an illustration). Nevertheless, some information about the link between the input and the output is revealed. For example, in Figure 1 an observer knows that the input values x_1, x_2 map to y_2, y_3 in some way and that x_3, x_4 map to y_1, y_4 in some way, and hence, for instance, she learns that x_4 does not map to y_2 or y_3.

2.2 Modeling Chaumian RPC Mix Nets

We now provide a formal model of Chaumian RPC mix nets, based on a computational model with interactive Turing machines. The computational model follows the one used in [15, 16], which we briefly recall before presenting the model of RPC mix nets and which in turn is based on the IITM model [14, 18].

2.2.1 The Computational Model

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 channels of the same name but opposite directions (input/output) are connected by such channels. A process may have external input/output channels, those that are not connected internally. In a run of a process, at any time only one program is active. 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. A 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 $\pi = p_1 \parallel \cdots \parallel p_l$, where $p_1, \ldots, p_l$ are programs. If π_1 and π_2 are processes, then $\pi_1 \parallel \pi_2$ 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).

A process π where all programs are given the security parameter ℓ is denoted by $\pi^{(\ell)}$. The processes we consider are such that 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.

Based on the notion of programs and processes, protocols and instances of protocols are defined as follows.

A *protocol* P specifies a set of agents (also called parties or protocol participants) and the channels these agents can communicate over. Moreover, P specifies, for every agent a, a set Π_a of all programs the agent a may run and a program $\hat{\pi}_a \in \Pi_a$, *the honest program of* a, i.e., the program that a runs if a follows the protocol.

Let P be a protocol with agents $a_1, \ldots, a_n$. An *instance of* P is a process of the form $\pi = (\pi_{\mathsf{a}_1} \parallel \ldots \parallel \pi_{\mathsf{a}_n})$ with $\pi_{\mathsf{a}_i} \in \Pi_{a_i}$. An agent a_i is *honest* in the instance π, if $\pi_{\mathsf{a}_i} = \hat{\pi}_{\mathsf{a}_i}$. A *run of* P (with security parameter ℓ) is a run of some instance of P (with security parameter ℓ). An agent a_i is honest in a run r, if r is a run of an instance of P with honest a_i.

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

As usual, a function f from the natural numbers to the interval $[0,1]$ is *negligible* if, for every $c > 0$, there exists ℓ_0 such that $f(\ell) \leq \frac{1}{\ell^c}$, for all $\ell > \ell_0$. The function f is *overwhelming* if the function $1 - f$ is negligible. A function f is λ-*bounded* if, for every $c > 0$ there exists ℓ_0 such that $f(\ell) \leq \lambda + \frac{1}{\ell^c}$, for all $\ell > \ell_0$.

2.2.2 Chaumian RPC Mix Nets Modeled as Protocols

We model an RPC mix net as a protocol in the sense of Section 2.2.1. The set of agents of such a protocol is as introduced in Section 2.1 plus two additional agents, the *judge* J and the scheduler Sch.

The programs of all agents are defined to have channels between each pair of agents. While not all channels are necessarily used by honest agents, they may be used by dishonest agents.

Scheduler. The honest program $\hat{\pi}_{\mathsf{Sch}}$ of the scheduler will be the master program. It triggers all agents in the appropriate order, according to the phases. It is part of every instance of the protocol and we assume that it is given information about which agents are honest and which are dishonest in order to schedule the agents in the appropriate way. In particular, the scheduler can schedule agents in a way advantageous for the adversary (dishonest agensts) so that we obtain stronger security guarantees. For example, the scheduler would first schedule honest senders to post their inputs on the bulletin board and then schedule dishonest senders. By this, the input of dishonest senders (the adversary) may depend on the input of honest senders. We also let $\hat{\pi}_{\mathsf{Sch}}$ create a common reference string (CRS), by calling the Setup algorithm of the non-interactive zero-knowledge proof system used, and provide it to all parties. The CRS is used by agents for non-interactive zero-knowledge proofs of correct decryption (see also Section 4.2 and Appendix A.3).

The bulletin board B. The honest program of B accepts messages from all agents. A message

received from an agent is stored in a list along with the identifier of the agent who posted the message. On request, B sends this list to an agent.

Auditors. For simplicity of presentation, we will simply assume one honest auditor A. The honest program $\hat{\pi}_{\mathsf{A}}$ of A, whenever triggered by the scheduler posts its random output on the bulletin board, as described in Section 2.1.

Sender. The honest program $\hat{\pi}_{\mathsf{S}}$ of a sender S implements the procedure described in Section 2.1: when triggered by the scheduler it first randomly picks a plaintext p according to some fixed probability distribution μ and then encrypts p as described and posts the resulting ciphertext on the bulletin board.[1] The honest program that is executed once p has been chosen is denoted by $\hat{\pi}_{\mathsf{S}}(p)$. As we will see, μ does not play any role for accountability, in which case we could simply assume the input to be provided by the adversary; this distribution, however, matters for our privacy result. It models prior knowledge of the adversary about the distribution of messages that honest senders send. In reality, in the context of e-voting, the adversary might not know this distribution precisely (only estimates according to election forecasts, for example). But assuming that the adversary knows this distribution precisely only makes the security guarantees that we prove stronger.

Mix server. The honest program $\hat{\pi}_{\mathsf{M}_j}$ of a mix server implements the procedure describe in Section 2.1. When triggered for the first time by the scheduler, it performs the described mixing procedure. It then waits to be triggered again by the scheduler to run the described audit procedure.

Judge. The honest program of the judge $\hat{\pi}_{\mathsf{J}}$ whenever triggered by the scheduler, reads data from the bulletin board and verifies it as described in Section 2.1. If a mix server M_i provides wrong output or if it simply declines to output the required data, the judge posts a message $\mathsf{dis}(\mathsf{M}_i)$, asserting that M_i misbehaved, i.e., M_i has not followed the prescribed protocol.

Trust assumptions. We assume that the scheduler, the bulletin board, the auditor, and the judge are honest. Formally, this means that the set Π_a of each such agent a consist of only the honest program $\hat{\pi}_a$ of that agent. All the other agents can (possibly) be dishonest. For a dishonest agent a, the set of its programs Π_a contains all probabilistic polynomially-bounded programs.

We denote RPC mix nets modeled as above with m mix servers and n senders that use a probability distribution μ to determine their choices by $\mathsf{P}_{\mathsf{mix}}(n,m,\mu)$. To study privacy, by $\mathsf{P}^{j}_{\mathsf{mix}}(n,m,\mu)$ we denote the variant of the protocol, where the j-th mix server is assumed to be honest (which, again, formally means that the set of all programs of M_j contains its honest program only).

3 Defining Accountability and Verifiability of RPC Mix Nets

In this section, we provide a definition of accountability for RPC mix nets, followed by a definition of verifiability. These notions are instantiations of the general, domain independent definitions of accountability and verifiability proposed in [15]. They should also apply to other forms of mix nets.

The (general) definition of accountability of a protocol from [15] is stated with respect to a property γ of the protocol, called the *goal*, a parameter $\lambda \in [0,1]$, and an agent J of the protocol who is supposed to blame protocol participants in case of misbehavior (when the goal γ is not achieved). The agent J, sometimes referred to as a *judge*, can be a "regular"

[1] We will always assume that all plaintexts chosen by (honest) senders have the same length. This assumption is needed in order to prove privacy; it is not needed for accountability.

protocol participant or an (external) judge, who is provided with information by other, possibly untrusted, protocol participants. Informally speaking, accountability requires two conditions to be satisfied:

(i) (*fairness*) J (almost) never blames protocol participants who are honest, i.e., run their honest program.

(ii) (*completeness*) If, in a run, the desired goal γ of the protocol is not met—due to the misbehavior of one or more protocol participants—, then J blames those participants who misbehaved, or at least some of them individually. The probability that the desired goal is not achieved but J nevertheless does not blame misbehaving parties should be bounded by λ.

To instantiate this definition for RPC mix nets, we first specify the goal γ and the parties who should be blamed in case γ is not achieved in a run. We then present the formal definition of accountability for mix nets, along the lines of the general definition of accountability from [15] sketched above.

The goal. As far as accountability (also verifiability) is concerned, we expect from an RPC mix net that the output strictly corresponds to the input, i.e., the plaintexts in the input ciphertexts and the plaintext in the output of the mix net should be the same multisets of plaintexts. This, of course, can be guaranteed for honest senders only, as dishonest parties may not follow the protocol and it might not even be clear what their input plaintexts are. Below, we formally describe this goal as a set of runs γ_0. Moreover, we generalize this goal by considering a family of goals γ_k, for $k \geq 0$, where γ_k is achieved if the output corresponds to the input, up to k changed entries. In other words, for the goal γ_k we tolerate up to k changes. This is useful for the study of RPC mix nets because, due to the nature of random partial checking, changing a small number of entries can go unnoticed with some probability. However, this probability should decrease very quickly with an increasing number of manipulated entries.

To formally specify the goal γ_k, let us consider a run r of an instance π of an RPC mix net P with n senders. Let $s_1, \ldots, s_l$ (for $l \leq n$) be those senders that are honest in r, $\vec{x} = x_1, \ldots, x_l$ be the input of these senders in r, and $\vec{y} = y_1, \ldots, y_p$ (with $p \leq n$) be the output of the mix net in r (if any), i.e., the sequence of plaintexts posted by the last mix server. We define r to belong to γ_k (in other words, γ_k is achieved in r), if there exists a subsequence $\vec{x}'$ of the honest input $\vec{x}$ of size $l-k$ such that $\vec{x}'$, treated as a multiset, is contained in $\vec{y}$ (again, treated as a multiset), i.e., for each element a of $\vec{x}'$, the count of a in $\vec{x}'$ is less than or equal to the count of a in $\vec{y}$. Hence, we require the output to contain $l-k$ elements from the honest input, while the remaining plaintexts, up to $n-(l-k)$, can be provided by the adversary. If in r no final output was produced (possibly because a mix server refused to produce output or the process was stopped because in in-phase auditing some mix server was blamed to have misbehaved), then r does not belong to γ_k, i.e., r does not achieve γ_k.

Parties to be blamed. We require that if the goal γ_k is not achieved, then the judge should blame at least one mix server, i.e., post $\mathsf{dis}(\mathsf{M}_i)$ for at least one i. By the fairness property for accountability, it follows that at least this mix server definitely misbehaved. By this, every mix server risks to be blamed in the case it misbehaves, i.e., does not follow the prescribed protocol. Note that we do not require the judge to blame *all* misbehaving servers. This requirement would be too strong, because not all misbehavior (i.e., deviations from the prescribed protocol) can be detected by the judge. However, the above guarantees that at least one mix server is (rightly) blamed in the case that γ_k is not achieved. The above requirement also implies that a sender cannot spoil the goal γ_k: if γ_k is not achieved, this must be due to a misbehaving mix server.

In the following definition of accountability for mix nets we say that if the judge posts $\mathsf{dis}(a)$, for some agent a, that the judge stated the *verdict* $\mathsf{dis}(a)$. Moreover, given an instance

π of a protocol P, we say that a verdict $\mathsf{dis}(a)$ *is true in* π if and only if a is not honest in π (in the sense of Section 2.2.1).

Now formally, accountability for RPC mix nets is defined as follows. We note that while this definition is formulated for RPC mix nets as introduced in Section 2.2.2, it should be useful also for other forms of mix nets. We write $\mathsf{Pr}[\pi^{(\ell)} \mapsto J : \mathsf{dis}(a)]$ to denote the probability that in a run of $\pi^{(\ell)}$ the judge J states the verdict $\mathsf{dis}(a)$. We write $\mathsf{Pr}[\pi^{(\ell)} \mapsto \neg\gamma_k \wedge \neg(J : \mathsf{dis}(\mathsf{M}_i) \text{ for some } i)]$ to denote the probability that in a run of $\pi^{(\ell)}$ the goal γ_k is not satisfied, i.e., the run does not belong to γ_k, and nevertheless J does not state a verdict $\mathsf{dis}(\mathsf{M}_i)$ for any i. Both probabilities are taken over the runs of $\pi^{(\ell)}$, i.e., the random coins used by the agents in π.

Definition 1. (Accountability for RPC mix nets) Let P be an RPC mix net protocol with an agent J (the judge), $\lambda \in [0,1]$, and $k \geq 0$. We say that P *provides λ-accountability with tolerance k (and w.r.t. J)*, if the following two conditions are satisfied.

(i) (*Fairness*) For all instances π of P and all verdicts $\mathsf{dis}(a)$ which are not true in π, the probability $\mathsf{Pr}[\pi^{(\ell)} \mapsto J : \mathsf{dis}(a)]$ is a negligible function in ℓ.

(ii) (*Completeness*)
For every instance π of P, the probability $\mathsf{Pr}[\pi^{(\ell)} \mapsto \neg\gamma_k \wedge \neg(J : \mathsf{dis}(\mathsf{M}_i) \text{ for some } i)]$ is a λ-bounded function in ℓ.

The above definition requires that the judge never (more precisely, only with negligible probability) blames mix servers that behave honestly, i.e., run their honest program. It also requires that the probability that the goal γ_k is not satisfied, and hence, more than k inputs of honest senders have been manipulated, but the judge nevertheless does not blame any single mix server, is bounded by λ.

Of course, mix nets where 0-accountability with tolerance 0 is achieved are desirable, i.e., mix nets, where even one manipulation of an honest input goes unnoticed with only negligible probability. While such mix nets can be obtained with more complex cryptographic constructions (some of which have been mentioned in the introduction), it is in the nature of RPC mix nets that there is some probability that manipulation goes unnoticed. One main contribution of this work is to precisely measure the level of accountability/verifiability RPC mix nets provide, and hence, with regard to the above definition, to find the optimal values of λ for the goals γ_k. This analysis is carried out Section 4.

Verifiability. Accountability and verifiability are tightly related as shown in [15]. Accountability is a stronger property than verifiability and subsumes it. While for verifiability one requires protocol participants to be able to see whether something went wrong or not, accountability additionally demands that if something went wrong, it is possible to blame specific misbehaving parties. This is an important security property in practice. Mix nets and e-voting systems should strive for accountability rather than only for verifiability. Nevertheless, traditionally, in the context of e-voting, the focus has been on verifiability, which is why here we also present a definition of verifiability for mix nets, based on the general, domain independent definition proposed in [15]. Similarly to the definition of accountability, the definition of verifiability is parametrized by the goal γ_k (defined just as in the case of accountability) and assumes a judge J which in the case of verifiability merely outputs accept or reject, depending on whether she thinks that the goal is achieved or not.

Definition 2. (Verifiability for RPC mix nets) Let P be an RPC mix net protocol with an agent J (the judge), $\lambda \in [0,1]$, and $k \geq 0$. We say that P *is λ-verifiable w.r.t. J and tolerance k*, if the following two conditions are satisfied.

(i) For all instances π of P where all mix servers are honest, the probability $\mathsf{Pr}[\pi^{(\ell)} \mapsto J : \mathsf{accept}]$ is an overwhelming function in ℓ.

Figure 2: Examples of cheating by the last mix server (left-hand side) and by any mix server (right-hand side).

(ii) For every instance π of P, the probability $\mathsf{Pr}[\pi^{(\ell)} \mapsto \neg\gamma_k \wedge J : \mathsf{accept}]$ is a λ-bounded function in ℓ.

Condition (ii) say that the probability that in a run the goal is not satisfied but J nevertheless accepts the run should be small (bounded by λ). This condition can easily be satisfied by judges that do not accept any run. Condition (i) therefore requires that if all mix servers are honest, then the judge should accept the run, which together with Condition (ii) implies that (with high probability) the goal is satisfied. It follows from results shown in [15] that if an RPC mix net provides λ-accountability with tolerance k and w.r.t. a judge J, then the RPC mix net also provides λ-verifiable with tolerance k and w.r.t. J', where J' outputs reject if J would blame some party.

We note that for RPC mix nets every party (also external observers) can play the role of the judge, who needs to examine publicly available information only.

4 Analysis of Accountability of the Chaumian RPC Mix Nets

In this section, we provide formal results for the level of accountability (and hence, verifiability) Chaumian RPC mix nets provide. As already mentioned in the introduction, this is the first rigorous analysis of accountability/verifiability for Chaumian RPC mix nets in the literature.

We start, in Section 4.1, with a description of some attacks on the accountability/verifiability of Chaumian RPC mix nets. We then present our formal results, which show that these mix nets have a reasonable level of accountability/verifiability. In particular, they show that there are no worse attacks than those described in Section 4.1.

4.1 Attacks

The most obvious way in which a mix server can cheat is when it replaces the result of the decryption of an input ciphertext by another ciphertext (or another plaintext in the case that the last mix server cheats in its second mixing step). If the mix server does not lie about the permutation it used, then this kind of cheating is (not) detected with probability $\frac{1}{2}$. If the mix server cheats in this way for $k+1$ input ciphertexts at the same time (and hence, would violate γ_k), its probability of not being caught is $(\frac{1}{2})^{k+1}$. Of course, all dishonest mix servers could cheat in this way.

However, there are more subtle ways of cheating which result in dishonest mix servers being caught less likely.

Cheating by the last mix server. This attack does not seem to have been described before in the literature. The attack can be applied by the last mix server in its second mixing step. Note

that the last mix server outputs the final plaintexts. The idea of the attack is that if after the first mixing step performed by the last mix server for two different positions p and q (marked gray on the left-hand side of Figure 2) the ciphertexts $C_{2m-1}[p]$ and $C_{2m-1}[q]$ decrypt to the same plaintexts (the last mix server knows this), it, instead of committing to $\pi^{-1}_{2m-1}(p)$ and $\pi^{-1}_{2m-1}(q)$, respectively, commits to, say, $\pi^{-1}_{2m-1}(p)$, for both p and q. Then, the plaintext at position q can be replaced by any other plaintext, and hence, the mix server can replace one plaintext by a plaintext of it's choice. This way of cheating is detected only if both $p \notin I_{m-1}$ and $q \notin I_{m-1}$, because in this case only it would be visible that the mix server did not commit to a permutation in its second sequence of commitments. The probability for this is $\frac{1}{4}$. Hence, the probability that this way of cheating goes undetected is $\frac{3}{4}$. Of course, the last mix server can apply this attack on different pairs of ciphertexts that decrypt to the same plaintext in order to replace many plaintexts. Applied to $k+1$ different pairs of ciphertexts results in the violation of γ_k and this remains undetected with probability $(\frac{3}{4})^{k+1}$.

This problem could be fixed, for example, as follows: i) require honest senders to add a nonce to their plaintext in order to avoid clashes between plaintexts, or ii) add another level of encryption (where for this last layer of encryption no mixing is performed, only decryption). However, these fixes do not improve the level of accountability RPC mix nets provide, as there are other, equally harmful attacks (like the following one).

Cheating by any mix server. This attack, which seems to have been sketched already in [13] and is illustrated on the right-hand side of Figure 2, can be applied to the first mixing step of any mix server. The idea is that a mix server M_j for two positions p and q in its intermediate sequence C_{2j+1} of ciphertexts sets both $C_{2j+1}[p]$ and $C_{2j+1}[q]$ to be the decryption of $C'_{2j}[\pi_{2j}(p)]$ (an honest M_j would set $C_{2j+1}[q]$ to be the decryption of $C'_{2j}[\pi_{2j}(q)]$). Moreover, in its first sequence of commitments, both at positions p and q it commits to the value $\pi_{2j}(p)$ (an honest M_j would at position q commit to $\pi_{2j}(q)$).

Now in the duplicate elimination phase, performed by the next mix server after the second mixing step of M_j, if $j < m-1$, one of the two copies of the result of the decryption of $C_{2j+1}[p]$ will be removed (on the right-hand side of Figure 2, one of the gray nodes in the rightmost column). As a result, one of the input ciphertexts from C'_{2j} is dropped (the black node in the example).

Analogously to the previous attack, this attack can be detected with probability $\frac{1}{4}$, because detection requires that both p and q belong to I_j. As before, one mix server can apply this attack for multiple pairs of positions and it can also be performed by many mix servers in order to manipulate (in this case drop) many input ciphertexts. Performing the attack on $k+1$ different pairs of ciphertexts (by the same mix server or different mix servers) results in the violation of γ_k and this remains undetected with probability $\left(\frac{3}{4}\right)^{k+1}$.

This seems to be an inherent problem for RPC mix nets, without an obvious fix.

4.2 Formal Analysis of Accountability of the Mix net

We now state and prove the precise level of accountability/verifiability Chaumian RPC mix nets have. While from the above it is clear that λ_k, i.e., the probability of more than k manipulations going unnoticed, may be as high as $(\frac{3}{4})^{k+1}$, we prove that the probability is not higher, and hence, there are no worse attacks.

Recall from Section 2.2.2 that we assume that the scheduler, the judge, the auditor, and the bulletin board are honest. However, none of the mix servers nor the senders are assumed to be honest.

Security assumptions. We make the following assumptions about the cryptographic primitives used. We assume the commitment scheme to be computationally binding and perfectly

hiding, with Pedersen commitments being an example [21]. (However, a scheme that is only computationally binding and hiding would do as well.) For the public-key encryption scheme, we assume it to be randomized such that for every plaintext in the domain of the scheme the probability of producing two identical ciphertext when encrypting the plaintext twice under the same public-key is negligible. This property is satisfied, for example, by all IND-CPA secure schemes. (Later, for privacy, Section 6, we will require an IND-CCA2 secure public key encryption scheme. But for accountability IND-CCA2 security is not necessary.) For the public key encryption scheme we, as usual, also assume that mix servers can provide proofs of correct decryption. More specifically, we require a non-interactive zero-knowledge (NIZK) proof of correct decryption, i.e., a NIZK proof that, for input of the form (m, c, pk), proves the statement $\exists sk : (pk, sk) \in K \wedge \mathsf{Dec}_{sk}(c) = m$, where K denotes the set of all public/private key pairs the key generation algorithm of the public key scheme can produce. For this, as already mentioned in Section 2.2, all parties are provided with a CRS by the scheduler. Note that an honest mix server knows the secret key sk related to its public key pk, and hence, can prove the above statement. Also observe that m might be $\perp$, in which case the statement states failure of decryption. To prove accountability, the zero-knowledge property is actually not needed (only completeness and soundness). But to prove privacy, we need the ZK property as well.

Now, the following theorem holds for Chaumian RPC mix nets as modeled in Section 2.2 for both in-phase and post-phase auditing.

Theorem 1. *Under the above assumptions concerning the cryptographic primitives, Chaumian RPC mix nets provide λ_k-accountability with tolerance k, where $\lambda_k = \left(\frac{3}{4}\right)^{k+1}$, and they do not provide λ-accountability for any $\lambda < \lambda_k$, i.e., λ_k is optimal.*

This theorem implies that even if all mix servers are dishonest, the probability that more than k inputs of honest voters have been manipulated, but the judge nevertheless does not blame any mix server, is bounded by $\left(\frac{3}{4}\right)^{k+1}$. For example, the probability that more than 5 manipulations go undetected is less than 18% and 10 manipulations go undetected in less than 4.5% of the cases. Moreover, if manipulation is detected, at least one mix server is blamed (and rightly so) for its misbehavior. As explained in Section 3, Theorem 1 also immediately implies that Chaumian RPC mix nets enjoy λ_k-verifiability with tolerance k.

In the proof of Theorem 1 (see Appendix B for a proof sketch), we show that the attack described above (*cheating by any mix server*) is an optimal strategy. Every other way of cheating does not give a better or more effective way of manipulating the result.

5 Defining Privacy of RPC Mix Nets

In this section, we propose a definition of privacy that is suitable for RPC mix nets in that it allows one to measure the level of privacy a protocol provides. The ability to measure the level of privacy is important in the context of RPC mix nets because such protocols do not achieve perfect privacy: the adversary can learn information from a protocol run and therefore it is essential to be able to precisely tell how much he can learn.

Since the main application of RPC mix nets is e-voting, our definition of privacy for RPC mix nets resembles the definition of privacy for e-voting protocols proposed by Küsters et al. in [16]. In their definition, privacy is formalized as the inability of an observer to distinguish whether some voter v (called the voter under observation) voted for candidate j or candidate j', when running her *honest* voting program (as specified by the voting protocol). Analogously, here we formalize privacy of RPC mix nets as the inability of an adversary to distinguish whether some sender under observation submitted plaintexts p or p', when running

her honest program. While this definition is quite strong (see, e.g., the discussion in [2]), simulation-based definitions [12] are stronger (see also [3] for a related game-based definition). Roughly speaking, simulation-based definitions imply that an adversary should not be able to distinguish between two (different) vectors of honest inputs. However, as explained at the end of Section 2.1, in the case of RPC mix nets an adversary obtains partial information about how the input is mapped to the output, and hence, RPC mix nets do not satisfy such simulation-based definitions.[2] Nevertheless, this does not necessarily mean that RPC mix nets do not do a reasonably good job in hiding which specific message an individual honest sender sent. This security requirement corresponds to the central property in the context of e-voting, already sketched above, and it is what our privacy notion for RPC mix nets, which we define precisely below, is therefore supposed to capture.[3]

In the analysis of privacy of RPC mix nets, it turns out that it is useful to distinguish between *risk-avoiding* and *venturesome* adversaries, i.e., between adversaries that try to avoid being caught (i.e., blamed by the judge for misbehavior) and those that do not care. The class of venturesome adversary is simply the class of all probabilistic polynomial-time adversaries. In Section 5.3, we introduce and precisely define the concept of risk-avoiding adversaries.

5.1 Definition of Privacy w.r.t. Venturesome Adversaries

As already mentioned in Section 2.2, for studying privacy, we consider the protocol $\mathsf{P}^{j}_{\mathsf{mix}}(n,m,\mu)$, where the j-th mix server is assumed to be honest, all other mix servers may be dishonest. Among the n senders, we consider one sender s to be under observation. (The task of the adversary is to figure out whether this sender sent plaintext p or p'.)

Now, given a sender s and a plaintext p, the protocol $\mathsf{P}^{j}_{\mathsf{mix}}(n,m,\mu)$ induces a set of instances of the form $(\hat{\pi}_{\mathsf{s}}(p) \parallel \pi^*)$ where $\hat{\pi}_{\mathsf{s}}(p)$ is the honest program of the sender s under observation that takes p as its unencrypted input (as defined in Section 2.2) and π^* is the composition of programs of the remaining parties (scheduler, auditor, judge, senders, mix servers), one program $\pi \in \Pi_a$ for each party a. Recall that according to the definition of $\mathsf{P}^{j}_{\mathsf{mix}}(n,m,\mu)$, if a is the scheduler, the auditor, the judge, or the j-th mix server, then Π_a contains only the honest program of that party, as they are assumed to be honest. All other parties may run arbitrary (adversarial) probabilistic polynomial-time programs. Since we do not restrict these programs to avoid accusations by the judge, this models venturesome adversaries.

Privacy for Chaumian RPC mix nets (w.r.t. venturesome adversaries) is now defined as follows, where we use the following notation: $\mathsf{Pr}[(\hat{\pi}_{\mathsf{s}}(p) \parallel \pi^*)^{(\ell)} \mapsto 1]$ denotes the probability that the adversary (i.e., some dishonest agent) writes the output 1 on some dedicated channel in a run of $\hat{\pi}_{\mathsf{s}}(p) \parallel \pi^*$ with security parameter ℓ and some plaintext p. The probability is over the random coins used by the agents in $\hat{\pi}_{\mathsf{s}}(p) \parallel \pi^*$.

Definition 3. For $\mathsf{P}^{j}_{\mathsf{mix}}(n,m,\mu)$ as before let s be the sender under observation, $l < n-1$, and $\delta \in [0,1]$. We say that $\mathsf{P}^{j}_{mix}(n,m,\mu)$ *with l honest senders achieves δ-privacy*, if

$$\mathsf{Pr}[(\hat{\pi}_{\mathsf{s}}(p) \parallel \pi^*)^{(\ell)} \mapsto 1] - \mathsf{Pr}[(\hat{\pi}_{\mathsf{s}}(p') \parallel \pi^*)^{(\ell)} \mapsto 1] \tag{1}$$

[2] These security notions imply that the success probability of an adversary trying to distinguish between the two input vectors is bounded by $\frac{1}{2}$ up to a negligible value. It is easy to see that for a fixed number of honest mix servers (our results on privacy assume even only one honest mix server), the probability of distinguishing between the two input vectors will exceed $\frac{1}{2}$ by a non-negligible value. RPC mix nets might satisfy the definition if the number of (honest) mix servers grows in the length of the security parameter, but this is unrealistic. It might be interesting future work to see how many (honest) mix servers are needed to decrease the success probability of the adversary for the stronger notions to a reasonable level. However, this number might be unrealistically big.

[3] Alternatively to the definition of privacy used here, one could think of a definition where the adversary is asked to directly link a specific output to the input. However, such a link might not always be well-defined and such a definition seems to require a specific mix net structure.

is δ-bounded as a function of the security parameter ℓ, for all valid input plaintexts[4] p, p' and all programs π^* of the remaining parties such that (at least) l senders are honest in π^*.

Since δ typically depends on the number l of honest senders, privacy is formulated w.r.t. this number. Note that a smaller δ means a higher level of privacy. However, δ cannot be 0, not even in an ideal protocol, as detailed in the following subsection: there is, for example, a non-negligible chance that all honest senders sent the same message. In this case, the adversary knows the message sender s has sent, and hence, can easily distinguish between s having sent p or p'.

5.2 Privacy for the Ideal Mix Net Protocol

Before we introduce the notion of privacy w.r.t. risk-avoiding adversaries, we first study the level of privacy (w.r.t. venturesome adversaries) for the ideal mix net. More specifically, we determine the optimal $\delta^{id}_{l,\mu}$ in this case. This is useful because i) this value constitutes a lower bound for all kinds of mix net protocols and ii) the level of privacy for Chaumian RPC mix nets can be expressed in terms of this value.

In the ideal mix net, the senders submit their input plaintexts on a direct channel to the ideal mix net. The ideal mix net then outputs the submitted messages after having applied a random permutation. Honest senders choose their inputs according to the distribution μ.

The level of privacy provided by the ideal mix net, as well as the justification of the result, coincides with the level of privacy provided by the ideal voting protocol, as studied by Küsters et al. in [16]. It depends on the number l of honest senders and the probability distribution μ on valid input plaintexts.

To define $\delta^{id}_{l,\mu}$, we need the following terminology. Let $\{p_1,\ldots,p_k\}$ be the set of valid plaintexts. Since the adversary knows the input plaintexts of the dishonest senders, he can simply filter out these plaintexts from the final output and obtain what we call the *pure output* $\vec{r} = (r_1,\ldots,r_k)$ of the protocol, where r_i, $i \in \{1,\ldots,k\}$, is the number of times the plaintext p_i occurs in the output after having filtered out the dishonest inputs. Note that, if l is the number of honest senders, then $r_1 + \cdots + r_k = l+1$ (l honest senders plus the sender under observation).

We denote by *Out* the set of all pure outputs. Let $A^i_{\vec{r}}$ denote the probability that the choices made by the honest senders yield the pure output $\vec{r}$, given that the sender under observation submits p_i. Further, let $M_{j,j'} = \{\vec{r} \in Out : A^j_{\vec{r}} \leq A^{j'}_{\vec{r}}\}$. Now, the intuition behind the definition of $\delta^{id}_{l,\mu}$ is as follows: If the observer, given a pure output $\vec{r}$, wants to decide whether the observed sender submitted p_j or $p_{j'}$, the best strategy of the observer is to opt for $p_{j'}$ if $\vec{r} \in M_{j,j'}$, i.e., the pure output is more likely if the sender submitted $p_{j'}$.

This leads to the following level of privacy provided by the ideal mix net protocol with l honest senders and the probability distribution μ:

$$\delta^{id}_{l,\mu} = \max_{j,j' \in \{1,\ldots,k\}} \sum_{\vec{r} \in M_{j,j'}} (A^{j'}_{\vec{r}} - A^j_{\vec{r}}),$$

with example values depicted in Figure 3. (Note that $A^{j'}_{\vec{r}}$ - $A^j_{\vec{r}}$ depend on l and μ.)

The proof of this statement is a straightforward adaption of the proof for the ideal voting protocol [16].

[4] Recall that valid input plaintexts all have the same length.

5.3 Definition of Privacy w.r.t. Risk-Avoiding Adversaries

To define the notion of privacy w.r.t. risk-avoiding adversaries, let $\mathsf{P}^{j}_{\mathsf{mix}}(n,m,\mu)$, $\hat{\pi}_{\mathsf{s}}(p)$, and π^* be defined as before.

We say that π^* is *risk-avoiding* if the probability that the system $(\hat{\pi}_{\mathsf{s}}(p) \parallel \pi^*)^{(\ell)}$ produces a run where the judge states a verdict $\mathsf{dis}(a)$ for some dishonest agent a is negligible as a function in the security parameter ℓ, for all valid input plaintexts p.

Now, in the definition of privacy w.r.t. risk-avoiding adversaries, we simply restrict the set of programs π^* to those that are risk-avoiding.

Definition 4. For $\mathsf{P}^{j}_{\mathsf{mix}}(n,m,\mu)$ let s be the sender under observation, $l < n-1$, and $\delta \in [0,1]$. We say that $P^{j}_{mix}(n,m,\mu)$ *with l honest senders achieves δ-privacy w.r.t. risk-avoiding adversaries*, if

$$\mathsf{Pr}[(\hat{\pi}_{\mathsf{s}}(p) \parallel \pi^*)^{(\ell)} \mapsto 1] - \mathsf{Pr}[(\hat{\pi}_{\mathsf{s}}(p') \parallel \pi^*)^{(\ell)} \mapsto 1] \tag{2}$$

is δ-bounded as a function of the security parameter ℓ, for all valid input plaintexts p, p' and all risk-avoiding programs π^* of the remaining parties such that (at least) l senders are honest in π^*.[5]

6 Analysis of Privacy of Chaumian RPC Mix Nets

We now state and prove the precise level of privacy Chaumian RPC mix nets have. We consider different cases, depending on whether in-/post-phase auditing is done and depending on whether adversaries are risk-avoiding or venturesome. As we will see, altogether the level of privacy is quite satisfying, only in the case of post-phase auditing and venturesome adversaries privacy is completely broken. The case of venturesome adversaries appears to be quite unlikely in practice, e.g., in the context of e-voting, where malicious behavior might be punished severely. (Recall from Section 4 that the probability of being caught cheating is high.) In all other cases, the level of privacy is quite close to the ideal case ($\delta^{id}_{l,\mu}$) given sufficiently many senders. Surprisingly, this is even so for venturesome adversaries in the case of in-phase voting.

We note that in our analysis of privacy, we always make the worst case assumption, namely that only one of the mix servers is honest; clearly, if all mix servers are dishonest there cannot be any privacy.

In what follows, we first shortly discuss the reasons why Chaumian RPC mix nets are not perfect, i.e., why they do not offer exactly the same level of privacy as the ideal mix net. We then state the cryptographic assumptions we use in the privacy proof, followed by the analysis of privacy for all the cases mentioned above.

[5] We note that in [17] we considered α-risk-avoiding adversaries whose risk of being blamed was required to be α-bounded, with $\alpha \in [0,1]$. However, such adversaries include those that flip a (biased) coin and depending on the outcome of the coin flip behave honestly or do whatever it takes to break privacy (even if they will be caught for sure). So, in this case the coin flip decides whether the adversary is going to follow a strategy that results in him being caught. But, unless the adversary does not care at all whether he is caught or not, he would not follow such a strategy. Also, as already mentioned in the introduction, our results show that if an adversary cheats in order to increase his advantage in breaking privacy (even only very slightly), he will be caught with very high probability. Therefore, here we decided to not consider the general notion of α-risk-avoiding adversaries anymore, but restrict our attention to (completely) risk-avoiding adversaries ($\alpha = 0$) and to venturesome adversaries ($\alpha = 1$).

6.1 Problems with Privacy

As already illustrated in Section 2.1, it is in the very nature of the RPC mix nets that some information about the input to a mix server is mapped to its output. Consequently, the adversary obtains some partial information about how the input of the honest mix server is mapped to its output. Hence, privacy cannot be as in the ideal case. Note that for the other (dishonest) mix servers, the adversary has full knowledge about the mapping from the input to the output.

The second reason why the level of privacy for Chaumian RPC mix nets is worse than in the ideal case is that the level of verifiability/accountability is imperfect as well. To attack privacy, an adversary can use his ability to change unnoticeably (with some probability) the input to the honest server. In the extreme case, the adversary could drop all honest entries before the honest mix server is invoked and then, knowing the plaintexts in the dishonest entries, the adversary could easily determine the input of the sender under observation. Note, however, that in the case of in-phase auditing this works only if the misbehavior of the adversary is not detected before the honest server gets to decrypt its input. But from our analysis of verifiability/accountability we know that such a misbehavior would be detected with high probability.

A particularly interesting case from an analysis point of view is hence the case of in-phase auditing and venturesome adversaries. In this case, an adversary is confronted with a trade-off between the risk he takes (of being caught and of the protocol being aborted) and the additional information he obtains by manipulation. Our formal analysis provides the optimal strategy for the adversary to resolve this trade-off.

6.2 Cryptographic Assumptions

We make the same assumptions about the cryptographic primitives used in the protocol as in the case of accountability, plus the assumption that the encryption scheme is IND-CCA2 secure and that the proof of correct decryption is zero-knowledge.

6.3 Privacy for Risk-Avoiding Adversaries

We begin our analysis of privacy with the class of risk-avoiding adversaries. The results presented here hold true both for the case of in-phase and post-phase auditing.

By the results of Section 4, we know that whenever dishonest mix servers change some entry, this can be detected with non-negligible probability. Therefore, a risk-avoiding adversary will not change or drop any entry of an honest sender. Consequently, risk-avoiding adversaries can only attack privacy passively, that is, without changing the input to the honest server in any significant way. More specifically, we obtain the following result.

Theorem 2. *The protocol $\mathsf{P}^{j}_{mix}(n,m,\mu)$ with l honest senders achieves $\delta_{l,\mu}$-privacy w.r.t. risk-avoiding adversaries, where*

$$\delta_{l,\mu} = \frac{1}{2^l} \cdot \sum_{i=0}^{l} \binom{l}{i} \delta^{id}_{i,\mu} \quad .$$

Moreover, $\delta_{l,\mu}$ is optimal, i.e., this protocol does not achieve δ-privacy w.r.t. risk-avoiding adversaries for any $\delta < \delta_{l,\mu}$.

Example values for $\delta_{l,\mu}$ are depicted in Figure 3. As can be seen, for risk-avoiding adversaries, the level of privacy provided by the Chaumian mix net is only slightly worse than the level of privacy in the ideal mix net protocol. Recall that our result holds under the pessimistic assumption that there is only one honest mix server.

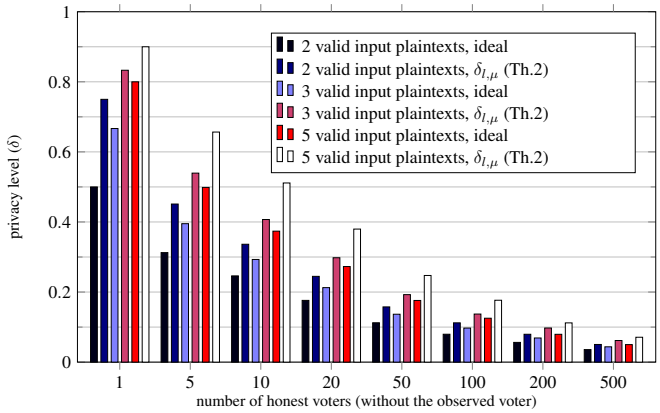

Figure 3: Level of privacy ($\delta_{l,\mu}$) w.r.t. risk-avoiding adversaries and in the ideal case $\delta^{id}_{l,\mu}$, uniform distribution of input plaintexts. These figures have been obtained by straightforward calculations using the δ-formulas as provided in the theorems.

Proof (sketch). We first introduce some notation and terminology. To simplify this notation, let us assume that the sender under observation has index 0 and the honest senders have indices from 1 to l. Let M_j denote the honest mix server.

Because we consider risk-avoiding adversaries, we know that, except with negligible probability, no entry is dropped or changed by the mix server (see Proposition 1 in Appendix C). In particular, the entry α^0_{2j} of the sender under observation occurs, as expected, in the input sequence C_{2j} of the honest mix server M_j (at some position). Similarly, the entries $\alpha^1_{2j},\dots\alpha^l_{2j}$ of the honest senders occur in C_{2j}. In a run of the mix net, these entries are divided by the audit procedure into two groups G_L and G_R. The group G_L contains those entries amongst $\alpha^0_{2j},\dots\alpha^l_{2j}$ for which the left link is opened during the audit procedure for M_j, i.e., the links from the entries in C'_{2j} to the corresponding entries in C_{2j+1} are revealed. The group G_R contains those entries for which the right links are opened. Considering, for example, Figure 1, entries x_1 and x_2 belong to G_L, whereas entries x_3 and x_4 belong to G_R. We call G_R and G_L *audit groups*.

Now, the rationale behind the definition of $\delta_{l,\mu}$ is the following: i represents the number of entries of honest senders that are, in a given run of the system, in the same audit group as the entry of the sender under observation. We consider all the possible cases, from $i = 0$ (the entry of the sender under observation is alone in its audit group, and hence, the adversary can easily see her choice) to $i = l$ (all the honest entries are in the same group as the entry of the sender under observation; in this case, privacy of the sender under observation is maximally protected). The probability that i honest senders belong to the same audit group as the sender under observation is $\binom{l}{i}\frac{1}{2^l}$, as it is decided independently for every honest entry if it belongs to the audit group of the sender under observation or not. Moreover, under the condition that the sender under observation is in an audit group with i honest senders, the situation corresponds to that of the ideal mix net with i honest senders. Hence, in this case, the level of privacy is $\delta^{id}_{i,\mu}$. Of course, the latter requires to use the hiding property of the commitment scheme, the assumption that the proofs of correct decryption are zero-knowledge, and a IND-CCA2-security of the public-key encryption scheme, in order to show that the adversary gains negligible advantage only by trying to break the cryptographic primitives (see Appendix C for details). □

6.4 Privacy for Venturesome Adversaries

We now analyze privacy for the class of venturesome adversaries. Here it makes a big difference whether in-phase or post-phase auditing is performed.

Post-phase auditing. We first study the case of post-phase auditing, for which, as argued next, no privacy can be guaranteed whatsoever, unless the honest server happens to be the first one (if this is the case, all the honest entries are in its input and privacy is as in the case of risk-avoiding adversaries). If the first mix server is not honest, a venturesome adversary can remove all the entries of the honest senders, except for the observed sender, before the honest mix server gets to mix its input. This will be detected with very high probability, but only after the protocol has been finished and after the adversary has obtained the information he wants. As a result of this way of cheating, the only honest entry left is the one of the sender under observation. So, for venturesome adversaries and post-phase auditing privacy is completely broken.

Formally, we obtain the following result.

Theorem 3. *The protocol $P^1_{mix}(n,m,\mu)$ (the first mix server is honest) with post-phase auditing and with l honest senders achieves $\delta_{l,\mu}$-privacy w.r.t. venturesome adversaries, where $\delta_{l,\mu}$ is defined as in Theorem 2.*

For $j > 1$, the protocol $P^j_{mix}(n,m,\mu)$ (the first mix server may not be honest) with post-phase auditing and with l honest senders does not achieve δ-privacy w.r.t. venturesome adversaries for any $\delta < 1$.[6]

The fact that for venturesome adversaries privacy is completely broken is quite obvious and has already been pointed out in the original proposal for RPC mix nets [11]. In [13], it was proposed to mitigate this problem by introducing an additional inner-most encryption with distributed decryption, where the private keys are distributed among the mix servers. By this, if one of the mix servers is honest, one can guarantee that the plaintexts are only output if no mix server was caught cheating. (This basically leads to the case of in-phase auditing discussed below.)

As already mentioned in the introduction, it is quite unlikely that adversaries are venturesome, at least in the context of e-voting, given the possibly severe negative consequences such adversaries would have to face. Also, as further discussed in the case of in-phase auditing below, our results show that an adversary who wants to obtain an advantage in breaking privacy would have to drop entries of honest senders before they reach an honest mix server. While the gain in breaking privacy when dropping a few honest entries (say k) is quite small, the probability of being caught is very high, namely $1-(\frac{3}{4})^k$. Hence, unless an adversary really does not care being caught, cheating does not make much sense in an attempt to break privacy.

In-phase auditing. Now we consider RPC mix nets with in-phase auditing. This is the most interesting case from the technical perspective, as the adversary, whenever he is to remove/change an entry, needs to balance out the risk of being caught, and hence, not learning anything useful, and the advantage cheating gives for breaking privacy.

In order to state the following theorem, let us note that, following the optimal strategy of cheating with a minimum risk of being caught (forming left collision groups of size two), as

[6] We note that in [17] we also considered the case of α-risk-avoiding adversaries in Theorem 3 for $\alpha \in (0,1)$. As mentioned in Section 5.3, we do not consider this notion anymore. It is clear that for α-risk-avoiding adversaries the level of privacy cannot be better than α: As already mentioned in Section 5.3, an α-risk-avoiding adversary could flip a (biased) coin, which comes up heads with probably α, and if it comes up heads this adversary could behave like a venturesome adversary and break privacy completely. However, this strategy (while formally a valid α-risk-avoiding strategy) does not seem to be what a realistic adversary would do, namely (despite being risk-avoiding) accept to be caught for sure only because a coin comes up heads. Unfortunately, in [17] the statement made about such adversaries in Theorem 3 was flawed.

described in Section 4, one dishonest mix server can drop $\lfloor \frac{l}{2} \rfloor$ out of l honest entries. From this it is easy to calculate how many mix server are needed to drop all honest tries from the initial input to the mix net.

Now, we obtain the following theorem for the level of privacy of Chaumian RPC mix nets in the case of in-phase auditing, venturesome adversaries, and assuming one honest mix server only.

Theorem 4. *The protocol $\mathsf{P}^{j}_{mix}(n,m,\mu)$ with in-phase auditing and with l honest senders achieves $\delta^*_{l,\mu}$-privacy w.r.t. venturesome adversaries, where*

$$\delta^*_{l,\mu} = \max_{l' \in \{0,\ldots,l\}} \left(\left(\frac{3}{4}\right)^{l-l'} \cdot \delta_{l',\mu} \right). \tag{3}$$

The protocol does not achieve δ-privacy w.r.t. venturesome adversaries for any $\delta < \delta^*_{l,\mu}$, assuming the number of mix servers preceding the honest mix server is sufficiently big in order to drop l honest entries following the optimal strategy, as discussed above.[7]

The intuition behind the constant $\delta^*_{l,\mu}$ is the following (see Appendix C.1 for the full proof): If an adversary lets l' entries of honest senders reach the honest mix server M_j (and hence, $l-l'$ are dropped) *and* the adversary is not caught dropping $l-l'$ entries, then his advantage of breaking privacy is (at most) $\delta_{l',\mu}$. This is because this case corresponds to the case with l' honest senders from Theorem 2. However, if the adversary is caught, he does not learn anything from the protocol run. The probability that the adversary is not caught if he drops $l-l'$ entries of honest sender is $\left(\frac{3}{4}\right)^{l-l'}$. We consider all possible l' as above and pick the one for which the advantage of the adversary is biggest.

We have computed the constants $\delta^*_{l,\mu}$ for those parameters we also considered for $\delta_{l,\mu}$ (Figure 3). For all those parameters, it has turned out that $\delta^*_{l,\mu} = \delta_{l,\mu}$. That is, the optimal strategy of the (venturesome) adversary is to *not* remove/change any of the honest entries (and, therefore, it coincides with the optimal strategy of a risk-avoiding adversary). This, in particular, means that the risk of being caught when manipulating honest entries, and in consequence not learning anything, always outweighs the advantage these manipulations yield for breaking privacy. Indeed, as we can see in Figure 3, the advantage in breaking privacy increases only very little, when we decrease the number of honest entries by, say 1, while the probability that the adversary gets caught (and learns nothing) when he drops even only one entry is quite substantial (25%). So, it appears that in the case of in-phase auditing it never makes sense for the adversary to cheat in its attempt to break privacy.

7 Conclusion

In this paper, we provided the first formal security analysis of Chaumian RPC mix nets. These mix nets are appealing not only due to their simplicity and efficiency, but also because they can be used with any IND-CCA2-secure public key encryption scheme (that allows for efficient proofs of correct decryption) and can handle arbitrarily long input messages.

We proved that these mix nets enjoy a high level of accountability/verifiability. The probability for a mix server of being caught cheating if it tries to manipulate k messages of honest senders is $1-(\frac{3}{4})^k$. Hence, already the manipulation of just one (two) message(s) is

[7] We note that in [17] we also considered α-risk-avoiding adversaries. However, for reasons explained before, we do not consider these adversaries anymore.

detected with probability 0.25 (0.43). In the context of e-voting, where cheating mix servers might face severe penalties, this might be a strong incentive to behave honestly.

The level of privacy is surprisingly good, namely close to the ideal case ($\delta^{id}_{l,\mu}$), already in a settings with a few hundred senders. The only exception is the case of post-phase auditing and venturesome adversaries, which take into account to be caught cheating for sure. In this case, there is no privacy. However, in the context of e-voting, it is quite unlikely that an adversary is venturesome, since an adversary caught cheating might have to face severe consequences. Also, our results show that if an adversary cheats in order to (even only very slightly) increase his advantage of breaking privacy, then the probably that this cheating is detected is quite high: to increase his advantage of breaking privacy an adversary has to drop entries of honest senders, but this is detected with high probability. So, unless an adversary really does not care if he is caught, cheating in an attempt to break privacy does not pay off, and hence, altogether the level of privacy provided to single senders/voters is quite satisfying.

In summary, our results show that Chaumian RPC mix nets offer a reasonable level of privacy and verifiability, and that they are still an interesting option for the use in e-voting systems.

Acknowledgment

This work was partially supported by *Deutsche Forschungsgemeinschaft* (DFG) under Grant KU 1434/6-2 within the priority programme 1496 "Reliably Secure Software Systems – RS^3".

References

[1] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. In H. Krawczyk, editor, *Advances in Cryptology, 18th Annual International Cryptology Conference (CRYPTO 1998)*, volume 1462 of *Lecture Notes in Computer Science*, pages 549–570. Springer, 1998.

[2] David Bernhard, Véronique Cortier, Olivier Pereira, and Bogdan Warinschi. Measuring vote privacy, revisited. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM Conference on Computer and Communications Security (CCS 2012)*, pages 941–952. ACM, 2012.

[3] David Bernhard, Olivier Pereira, and Bogdan Warinschi. How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios. In Xiaoyun Wang and Kazue 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 626–643. Springer, 2012.

[4] R. Carback, D. Chaum, J. Clark, adn J. Conway, E. 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 Elecion with Ballot Privacy. In *USENIX Security Symposium/ACCURATE Electronic Voting Technology (USENIX 2010)*. USENIX Association, 2010.

[5] David Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Commun. ACM*, 24(2):84–88, 1981.

[6] 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)*, pages 354–368. IEEE Computer Society, 2008.

[7] Philippe Golle, Sheng Zhong, Dan Boneh, Markus Jakobsson, and Ari Juels. Optimistic Mixing for Exit-Polls. In Yuliang Zheng, editor, *Advances in Cryptology - ASIACRYPT 2002, 8th International Conference on the Theory and Application of Cryptology and Information Security, Proceedings*, volume 2501 of *Lecture Notes in Computer Science*, pages 451–465. Springer, 2002.

[8] Marcin Gomulkiewicz, Marek Klonowski, and Miroslaw Kutylowski. Rapid Mixing and Security of Chaum's Visual Electronic Voting. In Einar Snekkenes and Dieter Gollmann, editors, *Computer Security - ESORICS 2003, 8th European Symposium on Research in Computer Security, Proceedings*, volume 2808 of *Lecture Notes in Computer Science*, pages 132–145. Springer, 2003.

[9] Jens Groth. Short Pairing-Based Non-interactive Zero-Knowledge Arguments. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*, pages 321–340. Springer, 2010.

[10] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect Non-interactive Zero Knowledge for NP. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*, pages 339–358. Springer, 2006.

[11] 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, 2002.

[12] Shahram Khazaei, Tal Moran, and Douglas Wikström. A Mix-Net from Any CCA2 Secure Cryptosystem. In Xiaoyun Wang and Kazue 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.

[13] Shahram Khazaei and Douglas Wikström. Randomized Partial Checking Revisited. In Ed Dawson, editor, *Topics in Cryptology - CT-RSA 2013 - The Cryptographers' Track at the RSA Conference 2013. Proceedings*, volume 7779 of *Lecture Notes in Computer Science*, pages 115–128. Springer, 2013.

[14] R. Küsters. Simulation-Based Security with Inexhaustible Interactive Turing Machines. In *Proceedings of the 19th IEEE Computer Security Foundations Workshop (CSFW-19 2006)*, pages 309–320. IEEE Computer Society, 2006. See [18] for a full and revised version.

[15] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Accountability: Definition and Relationship to Verifiability. In *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS 2010)*, pages 526–535. ACM, 2010.

[16] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Verifiability, Privacy, and Coercion-Resistance: New Insights from a Case Study. In *32nd IEEE Symposium on Security and Privacy (S&P 2011)*, pages 538–553. IEEE Computer Society, 2011.

[17] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Formal Analysis of Chaumian Mix Nets with Randomized Partial Checking. In *35th IEEE Symposium on Security and Privacy (S&P 2014)*. IEEE Computer Society, 2014. To appear.

[18] Ralf Küsters and Max Tuengerthal. The IITM Model: a Simple and Expressive Model for Universal Composability. Technical Report 2013/025, Cryptology ePrint Archive, 2013. Available at http://eprint.iacr.org/2013/025.

[19] C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In Michael K. Reiter and Pierangela Samarati, editors, *8th ACM Conference on Computer and Communications Security (CCS 2001)*, pages 116–125. ACM, 2001.

[20] Choonsik Park, Kazutomo Itoh, and Kaoru Kurosawa. Efficient Anonymous Channel and All/Nothing Election Scheme. In Tor Helleseth, editor, *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Proceedings*, volume 765 of *Lecture Notes in Computer Science*, pages 248–259. Springer, 1993.

[21] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Proceedings of the 11th Annual International Cryptology Conference (CRYPTO 1991)*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, 1991.

[22] Peter Y. A. Ryan, David Bismark, James Heather, Steve Schneider, and Zhe Xia. The Prêt à Voter Verifiable Election System. Technical report, University of Luxem- bourg, University of Surrey, 2010. http://www.pretavoter.com/publications/PretaVoter2010.pdf.

[23] K. Sako and J. Kilian. Receipt-Free Mix-Type Voting Scheme — A practical solution to the implementation of a voting booth. In *Advances in Cryptology — EUROCRYPT '95, International Conference on the Theory and Application of Cryptographic Techniques*, volume 921 of *Lecture Notes in Computer Science*, pages 393–403. Springer-Verlag, 1995.

[24] Douglas Wikström. Five Practical Attacks for "Optimistic Mixing for Exit-Polls". In Mitsuru Matsui and Robert J. Zuccherato, editors, *Selected Areas in Cryptography, 10th Annual International Workshop, SAC 2003, Revised Papers*, volume 3006 of *Lecture Notes in Computer Science*, pages 160–175. Springer, 2003.

[25] Douglas Wikström. A Universally Composable Mix-Net. In Moni Naor, editor, *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Proceedings*, volume 2951 of *Lecture Notes in Computer Science*, pages 317–335. Springer, 2004.

A Security Definitions for Cryptographic Primitives

A.1 IND-CCA2 Encryption

Let $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be an asymmetric encryption scheme.

Let C be a probabilistic polynomial-time algorithm, called a *challenger* that takes a bit b and a public/private key pair (pk, sk) and that serves two types of queries:

decryption query:
for a message y, the challenger returns the decryption of y, that is $\mathsf{Dec}_{sk}(y)$;

challenge query:
for a pair of messages (x_0, x_1), where x_0 and x_1 have the same length, the challenger encrypts x_b under pk and returns the ciphertext, that is $\mathsf{Enc}_{pk}(x_b)$.

Now, the encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is *IND-CCA2 secure* (see, for instance, [1]), if for every polynomially bounded adversary A who never submits decryption queries for a message y previously returned by a challenge query we have that

$$\mathsf{Pr}[(pk,sk) \leftarrow \mathsf{KeyGen}(1^\ell);\ b' \leftarrow A^{C(1,pk,sk)}(1^\ell, pk) : b' = 1]$$
$$-\mathsf{Pr}[(pk,sk) \leftarrow \mathsf{KeyGen}(1^\ell);\ b' \leftarrow A^{C(0,pk,sk)}(1^\ell, pk) : b' = 1]$$

is a negligible function in ℓ.

A.2 Commitments

A *commitment scheme* is a tuple (M, C, R, Comm), where, for each value ℓ of the security parameter, M_ℓ, C_ℓ and R_ℓ are sets of messages called, respectively, the *message space*, the *commitment space*, and the *opening space*, and Comm is a deterministic, polynomial-time algorithm that for each ℓ, each $m \in M_\ell$ and $r \in R_\ell$, outputs $c = \mathsf{Comm}(1^\ell, m, r) \in C_\ell$. By $\mathsf{Comm}(1^\ell, m)$ we denote the probabilistic algorithm that chooses a random r from R_ℓ with uniform probability and returns $\mathsf{Comm}(1^\ell, m, r)$.

Such a commitment scheme is *perfectly hiding*, if for each ℓ and each $m, m' \in M_\ell$, we have that $\mathsf{Comm}(1^\ell, m)$ and $\mathsf{Comm}(1^\ell, m')$ have the same distribution.

A commitment scheme (M, C, R, Comm) is *computationally binding*, if for all polynomially bounded adversaries A

$$\begin{aligned}\mathsf{Pr}[(m, m', r, r') \leftarrow A(1^\ell) :\ & \\ & r, r' \in R_\ell,\ m, m' \in M_\ell,\ m \neq m', \\ & \mathsf{Comm}(1^\ell, m, r) = \mathsf{Comm}(1^\ell, m', r')]\end{aligned}$$

is a negligible function in ℓ.

A.3 Non-interactive Zero-knowledge Proofs

Following [9, 10], we provide here a definition of non-interactive zero-knowledge proofs in the common reference string model.

Let R be an efficiently computable binary relation. For pairs $(x, w) \in R$, we call x the statement and w the witness. Let $L_R = \{x : \exists w \text{ such that } (x, w) \in R\}$. A *non-interactive proof system* for a language L_R is a tuple of probabilistic polynomial-time algorithms $(\mathsf{Setup}, \mathsf{Prover}, \mathsf{Verifier})$, where

– Setup (the common reference string generator) takes as input a security parameter 1^ℓ and the statement length n and produces a common reference string $\sigma \leftarrow \mathsf{Setup}(n)$,[8]

[8] We omit the security parameter in the notation, also for the prover and the verifier, for simplicity of notation.

– Prover (the prover) takes as input the security parameter 1^ℓ, a common reference string σ, a statement x, and a witness w and produces a proof $\pi \leftarrow \mathsf{Prover}(\sigma, x, w)$,
– $\mathsf{Verifier}$ (the verifier) takes as input the security parameter 1^ℓ, a common reference string σ, a statement x, and a proof π and outputs $1/0 \leftarrow \mathsf{Verifier}(\sigma, x, \pi)$ depending on whether it accepts π as a proof of x or not,

such that the following conditions (completeness and soundness) are satisfied.

Perfect completeness: For $n = \ell^{O(1)}$ and all adversaries A outputting $(x, w) \in R$ with $|x| = n$

$$\mathsf{Pr}[\sigma \leftarrow \mathsf{Setup}(n);\ (x, w) \leftarrow A(\sigma);\ \pi \leftarrow \mathsf{Prover}(\sigma, x, w);\ b \leftarrow \mathsf{Verifier}(\sigma, x, \pi) :\ b = 1] = 1.$$

This condition says that an honest prover should always be able to convince an honest verifier, if the statement is true.

Computational soundness: For $n = \ell^{O(1)}$ and all non-uniform polynomial time adversaries A, the probability

$$\mathsf{Pr}[\sigma \leftarrow \mathsf{Setup}(n);\ (x, \pi) \leftarrow A(\sigma);\ b \leftarrow \mathsf{Verifier}(\sigma, x, \pi) :\ x \notin L_R \text{ and } b = 1]$$

is a negligible function of the security parameter.

This condition captures that it should be infeasible for an adversary to come up with a proof of a false statement that is nevertheless accepted by the verifier.

We say that a non-interactive proof system $(\mathsf{Setup}, \mathsf{Prover}, \mathsf{Verifier})$ is zero-knowledge if the following condition is satisfied:

Computational general (multi-theorem) zero-knowledge: There exists a polynomial-time simulator $S = (S_1, S_2)$ such that, for $n = \ell^{O(1)}$ and all non-uniform polynomial time adversaries A submitting oracle queries $(x, w) \in R$ with $|x| = n$, we have

$$\mathsf{Pr}\left[\sigma \leftarrow \mathsf{Setup}(n)\ :\ A^{\mathsf{Prover}(\sigma,\cdot,\cdot)} = 1\right] = \mathsf{Pr}\left[(\sigma, \tau) \leftarrow S_1(n)\ :\ A^{S_2'(\sigma,\tau,\cdot,\cdot)} = 1\right]$$

where $S_2'(\sigma, \tau, x, w) = S_2(\sigma, \tau, x)$.

Above, we use the (general) multi-theorem variant of the zero-knowledge property, where the same common reference string can be used to produce many proofs. For our application, however, the weaker, single-theorem variant (where a common reference string can be only used for one ZK-proof), would actually suffice. This is because, in the mix net protocol we consider, the number of produced ZK-proofs is bounded and known a priori, which corresponds to the case, where A can only submit a bounded number of queries. In such a case, the single-theorem variant of the zero-knowledge property implies the multi-theorem variant (the length of σ can be expanded by factor of M, where M is the bound on the number of ZK-proofs).

B Proof of Theorem 1

Proving fairness (the first condition of the definition of accountability), is easy, so the rest of this section is devoted to the proof of completeness.

As we have already explained, we show that the strategy of the adversary that drops exactly $k+1$ honest entries as described in Section 4 (*cheating by any mix server*) is optimal for breaking the goal γ_k. The probability that using this strategy the adversary successfully removes $k+1$ honest entries is λ_k. Note that, using this strategy, one mix server can drop not more than half of its input entries. Therefore, if k is big, the adversary may need to use

more than one mix server. We may simply assume that there are enough mix servers for the adversary to carry out this strategy.

In the remainder of the proof, we show that no other strategy of the adversary is better than the aforementioned strategy.

Let P denote the composition of the (honest) programs of those parties of the system that are assumed to be always honest. The remaining parties are subsumed by an adversary A. Note that all mix servers are subsumed by A. Let us denote by X the event that, in the system $A \parallel P$, the goal γ_k is not achieved and no mix server is blamed. We want to show that the probability $\mathsf{Pr}[X]$ is λ_k-bounded, where $\lambda_k = (\frac{3}{4})^{k+1}$.

All the probabilities in this proof are computed, if not stated explicitly otherwise, over the sample space Ω such that every atomic event $\omega \in \Omega$ is a composition of random coins used by all the protocol participants of the system under consideration, that is the adversary and the parties in P. As usual, elements of Ω are sampled with uniform probability.

Tracing honest entries. We begin with defining how to trace honest entries throughout a run of the mix net. We define the number of honest entries in C_i, considering separately C_{2m} (that is the output of the last mix server) and the case of C_p with $0 \leq p < 2m$.

Let l be the number of honest senders and $\vec{x}$ be the (plaintext) input of these senders. *The number of honest entries in* C_{2m} is defined as the size of the maximal multiset $\vec{x}'$ such that $\vec{x}' \subseteq \vec{x}$ and $\vec{x}' \subseteq C_{2m}$, where by $\subseteq$ we denote the multiset subset relation. It is easy to see that the goal γ_k is achieved in a run if and only if the number of honest entries in C_{2m} is not smaller than $l-k$.

Now, let us consider the second case, i.e. C_p, for $0 \leq p < 2m$. Without loss of generality, we can assume that the indices of the honest senders are $1,\ldots,l$. Let us recall that, in such a case, α_p^i denotes the ciphertext produced by the i-th honest sender according to the protocol specification by encrypting her plaintext with the public keys $pk_{2m-1},\ldots,pk_p$. Let α_p denote the set $\{\alpha_p^1,\ldots,\alpha_p^l\}$ (recall that, with overwhelming probability, ciphertexts created by honest voters are distinct). *The number of honest entries in* C_p, for $0 \leq p < 2m-1$, is defined as the number of those elements of α_p that are in C_p.

It is easy to see that, if the j-th mix server follows the protocol specification, then the number of honest entries in its output (C_{2j+2}) is not smaller than the number of honest entries in its input (C_{2j}). Note also that, as we may assume that duplicate elimination steps are performed correctly,[9] the numbers of honest entries in C_{2j} and C'_{2j} are always the same. Finally, let us notice that in the optimal strategy described above, whenever a dishonest mix server does one manipulation, it decreases the number of honest entries by one.

Good runs. Let us consider an adversary A' which precisely simulates A, but carries out the following additional steps. When it produces output of a mix server M_j (that is messages C'_{2j}, C_{2j+1}, C_{2j+2}, comm_{2j}, and comm_{2j+1}, where comm_{2j}, comm_{2j+1} are supposed to be commitments to the permutations π_{2j} and π_{2j+1}, respectively), receives an audit challenge I_j, and produces its response to this challenge (in an honest run that would be appropriate openings to the challenged commitments), A' additionally simulates the responses of A to *all possible alternative challenges* I'_j. For this, A' always rewinds its state to the one right before receiving a challenge. In such a simulation, only the challenges change, otherwise all random coins are unchanged. If, during this process, A' discovers that (for different challenges) A opens the same commitment to two different values, it reports this conflict, by writing the commitment and the two different openings (on a distinct tape). Note that, although the number of possible challenges may be very big, it is constant and, therefore, this simulation can be

[9] If duplicate elimination is not done correctly by a mix server, the judge can easily detect this and would then blame the mix server. Since we prove completeness now, the assumption is that the judge does not blame any mix server. More precisely, we are interested in the probability that the goal is violated and no mix server is blamed.

done in a polynomial time. Note also that, for the same $\omega \in \Omega$, the runs of $A \parallel P$ and $A' \parallel P$ are identical, up to the additional simulation that A' performs and the possible reported conflicts.

Let us denote by G the subset of Ω such that for $\omega \in G$ we have:

(i) in the run of the system $(A' \parallel P)$ for ω, no conflict is reported,

(ii) in the run of the system $(A' \parallel P)$ for ω, no ZK proof of an invalid statement is produced that is accepted by the verifier (this includes also all ZK proofs produced and the verifications of these proofs performed in the simulation that A' performed),

(iii) ciphertexts produced by two different honest senders are different.

We will call the runs of $(A \parallel P)$ for $\omega \in G$, *good runs*.

It is easy to see that the probability of G is overwhelming. Indeed, one can easily see that the set of those $\omega \in \Omega$ for which $A' \parallel P$ reports a conflict is negligible: otherwise A' would break the computational binding property of the used commitment scheme. Also, the set of runs where mix nets output ZK proofs for invalid statements that are accepted by the verifier is, by the computational soundness of the used proof system, negligible. Finally, by the assumption about the used encryption scheme, two honest senders produce the same ciphertext only with negligible probability.

By the above observation, to complete the proof, it is enough to show that $\mathsf{Pr}[X \mid G] \leq (\frac{3}{4})^{k+1}$.

Overall structure. We know that the number of honest entries in C_0 is equal to the number l of honest senders. Let L be the set of vectors $\vec{l} = (l_0, \ldots, l_{m-1})$, where $l_j \in \{0, \ldots, l\}$, and $l_{m-1} < l - k$ (recall that m is the number of mix servers). For $j \in \{0, \ldots, m-1\}$, by l_j^* we will represent the event that the j-th mix server produces an output (denoted by C_{2j+2}) containing exactly l_j honest entries (it entails, in particular, that the protocol is not interrupted before this mix server is invoked which means that no previous server is blamed). By abuse of notation, let l_j represent, besides a number, the event that, additionally, this mix server is not blamed (the audit that follows does not discover any misbehaviour). Now, $\vec{l} = l_0, \ldots, l_{m-1}$ represents the event that no mix server is blamed and, for each $j \in \{0, \ldots, m-1\}$, the number of honest entries in C_{2j+2} is l_j. We define $L^* = \{\vec{l} \in L \mid \mathsf{Pr}[\vec{l} \mid G] \neq 0\}$.

As we have noted, the goal γ_k is *not* achieved if and only if the number of honest entries in C_{2m} is smaller than $l - k$. Therefore, by the definition of L^* (more precisely, by the assumption that $l_{m-1} < l - k$), the event $X \cap G$ holds if and only if $\vec{l}$ holds, for some $\vec{l} \in L^*$, Therefore, we have that

$$
\begin{aligned}
\mathsf{Pr}[X \mid G] &= \sum_{\vec{l} \in L^*} \mathsf{Pr}[\vec{l} \mid G] \\
&= \sum_{\vec{l} \in L^*} \begin{array}{l} \mathsf{Pr}[l_0^* \mid G] \cdot \mathsf{Pr}[l_0 \mid G, l_0^*] \cdot \\ \mathsf{Pr}[l_1^* \mid G, l_0] \cdot \mathsf{Pr}[l_1 \mid G, l_0, l_1^*] \cdots \\ \cdots \mathsf{Pr}[l_{m-1}^* \mid G, l_0, \ldots, l_{m-2}] \cdot \\ \qquad \cdot \mathsf{Pr}[l_{m-1} \mid G, l_0, \ldots, l_{m-2}, l_{m-1}^*] \end{array} \\
&= \sum_{\vec{l} \in L^*} \begin{array}{l} \mathsf{Pr}[l_0^* \mid G] \cdot \mathsf{Pr}[l_1^* \mid G, l_0] \cdots \\ \qquad \cdots \mathsf{Pr}[l_{m-1}^* \mid G, l_0, \ldots, l_{m-2}] \cdot \\ \mathsf{Pr}[l_0 \mid G, l_0^*] \cdot \mathsf{Pr}[l_1 \mid G, l_0, l_1^*] \cdots \\ \qquad \cdots \mathsf{Pr}[l_{m-1} \mid G, l_0, \ldots, l_{m-2}, l_{m-1}^*] \end{array}
\end{aligned}
$$

We will show the following fact which means that, if a dishonest mix server drops k_j honest entries, then the probability that this goes undetected is at most $(\frac{3}{4})^{k_j}$.

Lemma 1. *For all $j \in \{0, \ldots, m-1\}$ and $\vec{l} \in L^*$:*

$$Pr_j = \mathsf{Pr}[l_j \mid G, l_0, \ldots, l_{j-1}, l_j^*] \leq \left(\tfrac{3}{4}\right)^{k_j}, \tag{4}$$

where $k_j = \max(0, l_{j-1} - l_j)$ and where we put $l_{-1} = l$ (recall that l is the number of honest senders and hence the number of honest entries in the input to M_0).

We prove this lemma below. Now, using this lemma, we complete the proof of Theorem 1. First, we note that (4) implies the following fact.

$$\mathsf{Pr}[l_0 \mid G, l_0^*] \cdots \mathsf{Pr}[l_{m-1} \mid G, l_0, \ldots, l_{m-2}, l_{m-1}^*] \leq \left(\tfrac{3}{4}\right)^{k'}$$

where $k_j = \max(0, l_{j-1} - l_j)$ and $k' = k_0 + \cdots + k_{m-1}$. Now, because $l_{m-1} < l - k$, we have $k < k'$ and therefore we obtain:

$$\mathsf{Pr}[X \mid G] \leq \left(\tfrac{3}{4}\right)^{k+1} \cdot \sum_{\vec{l} \in L^*} \mathsf{Pr}[l_0^* \mid G] \cdot \mathsf{Pr}[l_1^* \mid G, l_0] \cdots \\ \cdots \mathsf{Pr}[l_{m-1}^* \mid G, l_0, \ldots, l_{m-2}]$$

One can easily see that the sum above is not bigger than 1, since the sum is equal to

$$\sum_{l_0} \mathsf{Pr}[l_0^* \mid G] \sum_{l_1} \mathsf{Pr}[l_1^* \mid G, l_0] \cdots \sum_{l_{m-1}} \mathsf{Pr}[l_{m-1}^* \mid G, l_0, \ldots, l_{m-2}],$$

where if conditionals have zero probability, we define the conditional probabilities to be zero.

Therefore, we obtain that

$$\mathsf{Pr}[X \mid G] \leq \left(\tfrac{3}{4}\right)^{k+1} \quad ,$$

which completes the proof.

Proof of Lemma 1. To prove (4), we need to first introduce some notation. For $\omega \in \Omega$, we define the event $\omega_j \subseteq \Omega$ where all random coins coincide with those of ω except possibly for the coins used by the auditors to generate the challenge for M_j. So, ω_j leaves it open how M_j is challenged, but otherwise the output of M_j is determined by ω_j. Let Ω_j be set of all ω_j as above. Similarly, for $\omega \in \Omega$ let $\omega_j^A \subseteq \Omega$ be the event where the random coins used by the auditors coincide with those of ω but other random coins are not fixed. Note that $\omega_j \cap \omega_j^A$ fixes a unique run of the system. In what follows, let $\Omega_j^* = \{\omega_j \mid \omega_j \in \Omega_j, \omega_j \cap G \cap l_0 \cap \cdots \cap l_{j-1} \cap l_j^* \neq \emptyset\}$. Now, we can represent the probability Pr_j as

$$Pr_j = \sum_{\omega_j \in \Omega_j^*} \mathsf{Pr}[\omega_j] \cdot \mathsf{Pr}[l_j \mid G, l_0, \ldots, l_{j-1}, l_j^*, \omega_j]. \tag{5}$$

We will show that for each $\omega_j \in \Omega_j^*$

$$\mathsf{Pr}[l_j \mid G, l_0, \ldots, l_{j-1}, l_j^*, \omega_j] \leq \left(\tfrac{3}{4}\right)^{k_j} \tag{6}$$

This implies that (5) is upper bounded by $(\frac{3}{4})^{k_j}$ and, hence, that (4) holds true. By this, proving (6) completes the proof.

In order to prove (6), we first observe several useful facts. Let us notice that all runs in ω_j determine the messages output by the j-th mix net before it is challenged by the auditors in the same way.

Now, because $\omega_j \in \Omega_j^*$, and hence, in particular $\omega_j \cap I_j^* \neq \emptyset$, we know that the number of honest entries in the output C_{2j+2} equals l_j. So we know the following:

(F1) The number of honest entries in the output C_{2j+2} is equal to l_j.

Similarly, we know the following:

(F2) The number of honest entries in C'_{2j} is equal to l_{j-1}.

Since $\omega_j \cap G \neq \emptyset$, we obtain that $\omega_j \subseteq G$: By the definition of ω_j and G, and the construction of A', it is easy to see that if one run in ω_j satisfies the Conditions (i), (ii), and (iii) of the definition of G, then all runs in ω_j satisfy these conditions.

In particular, by Condition (i) of the definition of G, we obtain the following fact:

(F3) There exists a function f such that for every run in ω_j and for every commitment c produced by some mix server in this run, if some mix server produces an opening to c, then the opened value is $f(c)$.

Similarly, by Condition (ii) of the definition of G, we obtain the following fact.

(F4) For every run in ω_j if a ZK proof produced by some mix server would convince the verifier, then the statement is actually true.

Next, in order to prove (6), we first rule out some trivial cases. The remainder of the proof then requires some combinatorial arguments.

Trivial cases. In the trivial cases we consider the probability

$$\mathsf{Pr}[l_j \mid G, l_0, \ldots, l_{j-1}, l^*_j, \omega_j] \tag{7}$$

in (6).

If the output of M_j does not conform to the expected format, then this mix server is immediately blamed. In this case, the probability (7) is zero (as the event l_j implies that, in particular, M_j is not blamed) and (6) trivially follows. So, we will assume that:

(A1) The output of M_j conforms to the expected format, i.e. M_j outputs C'_{2j}, C_{2j+1}, C_{2j+2}, comm_{2j}, and comm_{2j+1}, where C'_{2j} is correctly obtained from C_{2j} by the procedure of duplicate elimination, C'_{2j}, C_{2j+1}, and C_{2j+2} contain the same number r of elements, and comm_{2j} and comm_{2j+1} also contain r elements each, where the latter are supposed to be the commitments to the permutations π_{2j} and π^{-1}_{2j+1}, respectively (but M_j might be cheating).

If $l_{j-1} - l_j \leq 0$, then $k_j = 0$ and (6) trivially holds. So we will assume that

(A2) $l_{j-1} - l_j > 0$ (and thus $k_j > 0$), i.e. some number of honest entries is dropped by M_j.

The non-trivial case. Now, we prove (7) under the assumptions (A1) and (A2).

For simplicity of the argument, we assume that for every run in ω_j if M_j is challenged to open a commitment, it will always open the commitment correctly and such that the opened value is in the range $\{1, \ldots, r\}$. The case where M_j does not do this for every run in ω_j (and hence, every challenge) is proven in a similar way. Note that if M_j does not open a commitment as required, it will be blamed by the judge. Hence, such a run does not belong to the event l_j.

Let us consider the output of M_j as determined by ω_j. For each $i \in \{1, \ldots, r\}$, we will denote by $\mathit{link}_L(i)$ the index $f(c)$, where $c = \mathsf{comm}_{2j}[i]$ is the commitment of M_j to the i-th element of the permutation π_{2j} and f is given as in (F3). Note that by our assumption $\mathit{link}_L(i) \in \{1, \ldots, r\}$. Similarly, we will denote by $\mathit{link}_R(i)$ the index $f(c) \in \{1, \ldots, r\}$, where $c = \mathsf{comm}_{2j+1}[i]$. Note that by (F3) the functions link_L and link_R are the same for all runs in ω_j.

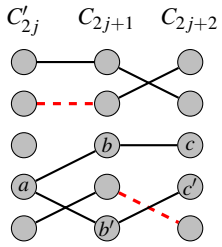

Figure 4: An example configuration induces by ω_j. Dashed red lines represent unsafe links, while solid black lines represent safe links. Note that a safe (black) link indicates the correct decryption relation. Thus, for instance, b and b' are the decryption of a and thus $b = b'$. Similarly, it follows that $c = c'$. Moreover, if a is an honest entry, then $c = c'$ is the corresponding honest entry produced by the same sender. Assuming that all the entries in the left column are honest, the right column might only contain two distinct honest entries. In this case, the mix server is nevertheless not blamed if no red link is audited and, moreover, if the two links pointing to a are not audited at the same time.

Figure 5: Examples of collision groups (nodes in rectangles) in the first and the second mixing.

Further, for each $i \in \{1,\ldots,r\}$, we will say that the index i is *left-unsafe* if $C_{2j+1}[i]$ is not the decryption of $C'_{2j}[i']$, where $i' = link_L(i)$. Note that if a left-unsafe link is audited (as requested by ω_j^A), then by (F4) the mix server is blamed. Similarly, we will say that the index i is *right-unsafe* if $C_{2j+2}[i']$ is not the decryption of $C_{2j+1}[i]$, where $i' = link_R(i)$. Again, if a right-unsafe link is audited, the mix server is blamed.

With the above definition, ω_j induces a configuration as the one presented in Figure 4. To complete the proof, we need to show that, for all possible such configurations which drop exactly k_j honest entries, the probability that the mix server is not blamed is bounded by $(\frac{3}{4})^{k_j}$. Computing this probability is, essentially, a purely combinatorial argument, as presented in what follows.

If two or more indices from the middle column point to the same index in the left column, we call it a left collision group. Formally, a *left collision* is a maximal set of indices L such that there exists a with $link_L(i) = a$ for all $i \in L$. Analogously, we define a *right collision group* as a maximal set of two or more indices from the middle column pointing to the same index in the right column. Examples of a left and a right collision groups of size 3 are depicted in Figure 5.

Let A be a subset of all runs in $G \cap l_0 \cap \cdots \cap l_{j-1} \cap l_j^* \cap \omega_j$ such that at most one index of every left collision group is right-unsafe. Let $\bar{A}$ denote its complement. As $\bar{A} \cap G \cap l_0 \cap$

$\cdots \cap l_{j-1} \cap l_j^* \cap \omega_j$ implies that either two indices of a left collision group are challenged to the left, or a right-unsafe link is challenged to the right, and hence, that M_j is blamed, we get that $l_j \cap G \cap l_0 \cap \cdots \cap l_{j-1} \cap l_j^* \cap \omega_j \subseteq A$ since l_j requires that M_j is not blamed. If $\mathsf{Pr}[A \mid G, l_0, \ldots, l_{j-1}, l_j^*, \omega_j] = 0$, we are done. Therefore, in the following we assume $\mathsf{Pr}[A \mid G, l_0, \ldots, l_{j-1}, l_j^*, \omega_j] \neq 0$. It suffices to show that

$$\mathsf{Pr}[l_j \mid G, l_0, \ldots, l_{j-1}, l_j^*, \omega_j, A] \leq \left(\frac{3}{4}\right)^{k_j}.$$

That is, in the following we assume that at most one index of every left collision group is right unsafe.

We say that an index in the middle column is *honest* if one of the following holds true:

1. it is neither left- nor right-unsafe nor belonging to a left collision group,
2. if it belongs to a left collision group that contains at least two indices that are not left-unsafe, then it is the lowest index of this collision group that is neither left- nor right-unsafe. (Note that, because of the event A, we know that one of the above mentioned indices is not right-unsafe.)

Let h_1 denote the number of indices that are not honest in the middle column.

Intuitively, the indices that are not honest in the middle column correspond to potentially dropped (honest) entries of M_j. (They do not have to actually drop entries because an adversary might for instance use right-unsafe indices to undo the effect of a left-unsafe link, which, however, would not be a good strategy of cheating.) Dishonest indices might not be the only reason that entries are dropped. Entries might also be further dropped due to right collision groups.

Now, the structure of the proof is roughly as follows: We show that the probably that the adversary is not blamed due to having produced dishonest indices is bounded by $\left(\frac{3}{4}\right)^{h_1}$. We then show that to every honest index i, except for some number h_2 of honest indices, we can assign a unique index k in the output of M_j such the decryption of the entry pointed to by i is the entry pointed to by k. The last step of the proof is to show that the probability of not getting blamed due to right collision groups, given that M_j is not blamed due to producing dishonest indices, is bounded by $\left(\frac{3}{4}\right)^{h_2}$. The theorem then follows by the observation that the number of dropped entries is at most $h_1 + h_2$ (that is the number of dishonest indices plus the number of honest indices that did not get an assignment).

Now, given the facts and the assumptions listed above, it is easy to see that M_j is not blamed if and only if no unsafe link is challenged (otherwise, by (F4), the ZK proof would not verify) and if at most one link from each left (or right) collision group is challenged to the left (or the right), as otherwise it is visible that M_j did not commit to a permutation.)

Let B be a subset of all runs in A such that no left-unsafe index is challenged to the left, no right-unsafe index is challenged to the right, and no two indices of a left collision group are challenged to the left. In other words, in runs in B the mix server M_j is not blamed due to having produced dishonest indices. (Still, in a run in B the mix server M_j can be blamed if the configuration contains a right collision group and this group is discovered.) We now show that

$$\mathsf{Pr}[B \mid G \cap l_0 \cap \cdots \cap l_{j-1} \cap l_j^* \cap \omega_j \cap A] \leq \left(\frac{3}{4}\right)^{h_1}. \tag{8}$$

The set of dishonest indices can be partitioned as follows: A dishonest index might i) belong to a left collision group which contains an honest index (and hence, at least two indices that

are not left unsafe), ii) belong to a left collision group that contains exactly one index that is not left unsafe (and hence, this left collision group does not contain an honest index), or iii) it might not belong to any left collision group but is left- and/or right-unsafe or it might belong to a left collision group which, however, contains only indices that are left unsafe. We now look at these (disjoint) sets separately.

The probably that at most one index in a collision group of size k as in i) is challenged to the left is $\frac{k+1}{2^k}$ (we consider $k+1$ cases, each occurring with probability $\frac{1}{2^k}$: one case if no index from the group is chosen for audit to the left, and k cases if exactly one index is chosen). An elementary calculation shows that $\frac{k+1}{2^k} \leq \left(\frac{3}{4}\right)^{k-1}$.

Now, let us consider left collision groups of the form ii). The probability that no index from such a left collision group of size k (≥ 2) that is not left unsafe is challenged to the left is $\left(\frac{1}{2}\right)^{k-1} \leq \left(\frac{3}{4}\right)^{k}$.

The dishonest indices in the set iii) are left- and/or right unsafe. So, for each such index, the probability that it is not challenged to its unsafe side is at most $\frac{1}{2} < \frac{3}{4}$.

As the indices are audited independently and the cases above do not overlap, we immediately get (8).

In the following, we assign to all but h_2 honest indices in the middle column (with h_2 being defined below) a unique index in the output column. More precisely, to every honest index i in the middle column (except for h_2 honest indices), we assign a unique (and different) index a in the output column of M_j such that the decryption of the entry pointed to by i is the entry pointed to by a. We call these honest indices *completely honest*. So, except for h_2 honest indices, all honest indices are completely honest.

Then, we have that $k_j \leq h_1 + h_2$, because for every completely honest index i, the input entry b pointed to by i (i.e., the entry in the left column at the index $link_L(i)$) has a valid decryption in the output column, namely at the index a assigned to i by the provided construction. Hence, b makes it through M_j. Moreover, the above assignment (from the input entries to the output entries) is one-to-one, because there is at most one honest index in a left collision group and two different completely honest indices are not assigned the same index. Therefore, the number of dropped entries of M_j is at most as high as the number of indices that are not completely honest. And, as the set of indices that are not completely honest is the disjoint union of the h_1 indices that are not honest in the middle column and the h_2 indices that are honest in the middle column but not completely honest, $k_j \leq h_1 + h_2$ follows.

Let C be a subset of all runs in A such that at most one index of a right collision group in M_j is challenged to the right. (Note that this is trivially true if there is no right collision group.) In other words, in runs in C the mix server M_j is not blamed due to the cheating done by using right collision groups.

Then, $C \cap B$ is the event that M_j is neither blamed for using left collision groups nor for right collision groups nor for left-unsafe indices nor for right-unsafe indices. And, as already mentioned, given the facts and assumptions listed before, M_j is not blamed exactly in this event, i.e. $C \cap B = l_j \cap A$.

Let $\{B_i : i = 1, \ldots, p\}$ denote a partition of B such that one B_i contains all runs in B where for every index that is left or right unsafe or belongs to a left collision groups it is fixed in the same way (in the different runs in B_i) whether this index is challenged to the left or to the right. In other words, one B_i determines one pattern of how indices in left collision groups and indices with left or right unsafe links are audited. (If $p = 0$, then this means that B is empty. In this case, we are done. Otherwise (if $p > 0$), all B_i are non-empty, by the definition of a partition.)

In what follows, we define for every B_i an h_2^i following the above sketched intuition such

that $k_j \leq h_1 + h_2^i$. Let $h_2^{min} = \min\{h_2^1, \ldots, h_2^p\}$. We then show that

$$\mathsf{Pr}[C \mid B_i] \leq \left(\frac{3}{4}\right)^{h_2^i}. \tag{9}$$

This then completes the proof:

$$\begin{aligned}
\mathsf{Pr}[C \cap B] &= \mathsf{Pr}[B] \cdot \mathsf{Pr}[C \mid B] \\
&= \mathsf{Pr}[B] \cdot \sum_{i=1}^{p} \mathsf{Pr}[C \cap B_i \mid B] \\
&= \mathsf{Pr}[B] \cdot \sum_{i=1}^{p} \mathsf{Pr}[C \mid B_i] \cdot \mathsf{Pr}[B_i \mid B] \\
&\leq \mathsf{Pr}[B] \cdot \sum_{i=1}^{p} \left(\frac{3}{4}\right)^{h_2^i} \cdot \mathsf{Pr}[B_i \mid B] \\
&\leq \mathsf{Pr}[B] \cdot \left(\frac{3}{4}\right)^{h_2^{min}} \\
&\leq \left(\frac{3}{4}\right)^{h_1} \cdot \left(\frac{3}{4}\right)^{h_2^{min}} \\
&\leq \left(\frac{3}{4}\right)^{k_j}.
\end{aligned}$$

So, it remains to define h_2^i and proof (9). In what follows, let $B^* = B_i$. The following assumption (P1) is, as argued below, made w.l.o.g. Also, the fact (P2) will be useful.

(P1) No right collision group contains two links that are opened to the right according to B^*.

Otherwise, if some right collision group contains two indices of which the right links are opened according to B^*, then the server is blamed. Hence the probability of not getting blamed (given B^*) is 0. In particular, $\mathsf{Pr}[C \mid B^*] = 0$, and hence, we would be done with proving (9).

(P2) For every left collision group that contains an honest index there is, according to B^*, an opened index to the right which is neither left nor right unsafe.

Indeed, as every left collision group with an honest index contains at least two indices that are not left-unsafe, given B, (at least) one of them must be open to the right. This index, again given B, cannot be right-unsafe.

For assigning unique outputs to honest indices in the middle column, we proceed as follows:

I) If an honest index i does not belong to a right collision group, we assign i to its right index $link_R(i)$.

II) Let i be an honest index that belongs to a left collision group L (so i is *the* honest index in L) and to a right collision group G. Note that in B^* it is fixed whether i is opened to the right or to the left (since i belongs to a left collision group).

- If i is opened to the right according to B^*, we assign to i the index to which G is linked to (i.e., the index $link_R(i)$).

- If i is opened to the left according to B^*, by (P2), there is another index i' in L which is neither left nor right unsafe and that is opened to the right according to B^*. We assign i to $link_R(i')$.

III) The remaining honest indices are those that do not belong to a left collision group but to a right collision group. While, by I) and II), all honest indices considered above have obtained an assignment, the remaining ones have not obtained an assignment yet. For these indices the assignments (if any) are defined as follows:

So, let i be an honest index that does not belong to a left collision group but to a right collision group. Let G denote the right collision group i belongs to. If $link_R(i)$ (which is the index to which all indices in G are linked to) has already been used for an assignment (i.e., some of the honest indices in I) or II) have been assigned to $link_R(i)$), then i is not assigned any index. In particular, this means that i is not completely honest, and hence, it contributes to h_2^i.

Otherwise (if $link_R(i)$) has not been assigned to), if i is the minimal index in the group G which does not belong to a left collision group, then i is assigned to $link_R(i)$.[10] If i is not minimal, it is not assigned any index, which, in particular, means that i is not completely honest, and hence, it contributes to h_2^i.

As, by (P1), in every right collision group, there is at most one index that by B^* is opened to the right, we indeed have for every completely honest index a different assignment. Also, by the construction it is clear that the entry corresponding to the index, say a, to which a completely honest index i is assigned is the result of correctly decrypting the entry corresponding to $link_L(i)$ (in the two mixing steps in M_j). Moreover, $link_L(i)$ is different for different completely honest indices (because two honest indices do not belong to the same left collision group). Hence, in this sense the entry at $link_L(i)$ makes it to the output of M_j, namely at index a. So, indeed we have that not more than $h_1 + h_2^i$ honest entries can have been dropped, and hence, $k_j \leq h_1 + h_2^i$, where h_2^i is defined to be the number of honest indices which have not received an assignment.

In what follows, let $h_2^* = h_2^i$. It remains to show that $\mathsf{Pr}[C \mid B^*] \leq \left(\frac{3}{4}\right)^{h_2^*}$. To this end, let $G_1, \ldots, G_z$ denote the right collision groups. For $i = 1, \ldots, z$, let t_i denote the number of honest indices in G_i that did not get an assignment, i.e., those honest indices that are not completely honest. Then, we have that $\sum_{i=1}^{z} t_i = h_2^*$.

Note that in B^* the random coins of all indices that belong to a left collision group or that have a left or right unsafe link are fixed in a specific way. For all honest indices that are not completely honest, B^* does not fix how they are audited because these indices do not belong to left collision groups and neither have left nor right unsafe links. How these indices are audited is chosen independently and uniformly at random even given B^*.

W.l.o.g., we assume that the $G_1, \ldots, G_z$ are ordered in such a way that $G_1, \ldots G_y$ are those right collision groups that contain an index opened to the right (according to B^*) and $G_{y+1}, \ldots, G_z$ are those right collision groups that do not contain such an index.

Then, G_i (for $i = y+1, \ldots, z$) contains at least $t_i + 1$ indices for which it is not determined by B^* how they are challenged (and for which the challenge is chosen independently and uniformly at random, even given B^*), as argued next: Clearly, as already mentioned, since G_i contains t_i honest but not completely honest indices, we know that for these t_i indices it is not determined by B^* how they are audited. Moreover, clearly none of these indices is assigned to the index q that G_i is linked to. This implies, by (III), that q is assigned to some other honest

[10] Minimality is not actually important. If $link_R(i)$) has not been assigned yet, we could take any index in G that does not belong to a left collision group and assign it to $link_R(i)$. However, exactly one of these indices should be assigned to $link_R(i)$.

index k. If k does not belong to a left collision group, then it must have obtained its assignment according to (III). But then the way k is audited is not determined by B^* and we are done, because, in this case, we altogether have t_i+1 indices in G_i for which the way they are audited is not determined by B^*. Now, consider the case that k belongs to a left collision group. This case is not possible: According to (II), k can only be assigned to q if q is connected to an opened linked. But, by assumption, G_i does not contain indices that are opened to the right.

For $i = 1,\ldots,y$, let C_i contain all runs in B^* such that no index of those indices in G_i that are not fixed by B^* is challenged to the right. Note that G_i contains at least t_i such indices, namely, the honest but not completely honest indices in G_i. Hence, we have that $\mathsf{Pr}[C_i \mid B^*] \leq \left(\frac{1}{2}\right)^{t_i} \leq \left(\frac{3}{4}\right)^{t_i}$.

For $i = y+1,\ldots,z$, let C_i contain all runs in B^* such that at most one index of those indices in G_i that are not fixed by B^* is challenged to the right. We know by the above that there are at least t_i+1 such indices in G_i. Hence, we have that $\mathsf{Pr}[C_i \mid B^*] \leq \frac{t_i+1+1}{2^{t_i+1}} \leq \left(\frac{3}{4}\right)^{t_i}$.

By definition of C and the construction of the C_i, we have that $C \cap B^* = \bigcap_{i=1}^{z} C_i$. Also, since different C_i talk about different sets of (independent) indices, we know that $\{C_i : i = 1,\ldots,z\}$ are independent, given B^*. Hence, we obtain the following:

$$\mathsf{Pr}[C \mid B^*] = \mathsf{Pr}[\bigcap_{i=1}^{z} C_i \mid B^*] = \prod_{i=1}^{z} \mathsf{Pr}[C_i \mid B^*] \leq \prod_{i=1}^{z} \left(\frac{3}{4}\right)^{t_i} \leq \left(\frac{3}{4}\right)^{h_2^*}.$$

This proves (9), and concludes the proof. □

Remark 1. As one can see from the proof (and as shortly remarked in Section 2.2), the probability distribution μ does not play any role in the above result. Indeed, we could allow the adversary to provide unencrypted input for the honest senders and the result would still work.

As an immediate corollary of the above proof, we obtain the following:

Corollary 1. *Consider the system $A \parallel P$ as above. Let X_j denote the event that in a run of $A \parallel P$ in the input C_{2j} for M_j not all entries of honest senders are contained, i.e., at least one is missing. Let B denote the event that the judge outputs $\mathsf{dis}(\mathsf{M}_i)$ for some dishonest agent, i.e., the judge blames some dishonest agent.*

Then, provided that $\mathsf{Pr}[G, X_j] > 0$, we have that $\mathsf{Pr}[B \mid G, X_j] \geq \frac{1}{4}$.[11]

Proof. The corollary follows easily with very similar reasoning as the one for $\mathsf{Pr}[X \mid G]$ using Lemma 1 in the proof of Theorem 1. □

C Proof of Theorem 2

Let P denote the considered mix net system $\mathsf{P}^j_{\mathsf{mix}}(n,m,\mu)$. More precisely, in what follows we denote by P a process consisting of only the programs of the honest parties in $\mathsf{P}^j_{\mathsf{mix}}(n,m,\mu)$. All dishonest parties are subsumed by an adversary process A. Hence, an instance of $\mathsf{P}^j_{\mathsf{mix}}(n,m,\mu)$ is of the form $A \parallel P$.

We will use the notion of audit groups, as introduced in the proof sketch of Theorem 2 (page 18). Now, however, in the context of a general (not necessarily risk-avoiding) adversary. In this case, $G_L \cup G_R$ does not need to contain all $\alpha^0_{2j},\ldots\alpha^l_{2j}$ (but, instead, we have $G_L \cup$

[11] The above events clearly depend on the specific security parameter. The statement is true for all choices of the security parameter.

$G_R \subseteq \{\alpha^0_{2j}, \ldots \alpha^l_{2j}\}$), because some of these entries may have been dropped/modified by the adversary.

Let $I_L, I_R \subseteq \{0, \ldots, l\}$ denote the sets of indices of those senders whose entries belong to G_L or, respectively, G_R. Let I_0 contain the remaining indices amongst $\{0, \ldots, l\}$ (i.e. those that do not belong to neither I_L nor I_R).

Idealized variant of the protocol. Let I_L and I_R be as above. We will denote by $P_{Id(I_L,I_R)}$ a variant of the protocol, where honest senders and the sender under observation use idealized encryption (they encrypt sequences of zeros instead of the original messages) in the step determined by I_L/I_R. Formally, this protocol is defined as the original protocol P with the following differences:

(a) A sender with an index $i \in I_L$, to obtain α^i_{2j+1}, instead of encrypting α^i_{2j+2} with the key pk_{2j+1}, encrypts a sequence of zeros of the same length. The pair $(\alpha^i_{2j+2}, \alpha^i_{2j+1})$ is then logged (to be used later by M_j).

(b) Similarly, a sender with an index $i \in \{0, \ldots, l\}$ that does *not* belong to I_L, to obtain α^i_{2j}, instead of encrypting α^i_{2j+1} with the key pk_{2j}, encrypts a sequence of zeros of the same length. As above, the pair $(\alpha^i_{2j+1}, \alpha^i_{2j})$ is logged.

(c) The honest mix server M_j, when it decrypts a ciphertext c, first checks whether there is a pair (d, c) logged as above. If this is the case, the mix server takes d as the result of decryption; otherwise M_j applies the decryption algorithm to c and returns its result.

When the honest mix server is requested to produce a zero-knowledge proof of an idealized encryption, it fails.

The following lemma states that, as long as the pair I_L/I_R adequately describes what happens in the audit phase, the original protocol and the idealized variant are indistinguishable. This fact is true even if the adversary knows the choices of all the senders. To express this, we will denote by $\bar{P}$ and $\bar{P}_{Id(I_L,I_R)}$ the variants of, respectively, P and $P_{Id(I_L,I_R)}$, where the adversary determines the choices (i.e. the input plaintexts) of all the honest senders (the choices of dishonest senders are, clearly, determined by the adversary as well).

Before we state this lemma, we fix some notation. By $\mathsf{Pr}[E]$ we will denote the probability of an event E over the sample space Ω such that every atomic event $\omega \in \Omega$ is a composition of random coins used by all the protocol participants of the system under consideration. As usual, elements of Ω are picked with uniform probability. By $(A \parallel \bar{P}) \mapsto 1$ we denote the event that a run of the system $A \parallel \bar{P}$ is accepted (A outputs 1). By $(A \parallel \bar{P}) \mapsto I_L/I_R$ we denote the event that a run of the system $A \parallel \bar{P}$ *conforms to* I_L/I_R, i.e. either the protocol is aborted before M_j executes its step or (M_j executes its steps and) the entries of senders in I_L are in the left audit group G_L, the entries of the sender I_R are in the right audit group G_R and the entries of the remaining senders in $0, \ldots, l$ are not delivered to M_j (are not in C_{2j}). Similarly, $(A \parallel \bar{P}_{Id(I_L,I_R)}) \mapsto 1$ and $(A \parallel \bar{P}_{Id(I_L,I_R)}) \mapsto I_L/I_R$ denote the corresponding events for the system $A \parallel \bar{P}_{Id(I_L,I_R)}$.

As usual, for systems Q_0 and Q_1, we write $Q_0 \equiv Q_1$, if Q_0 and Q_1 are computationally equivalent, i.e.

$$\left| \mathsf{Pr}\left[Q_0 \mapsto 1\right] - \mathsf{Pr}\left[Q_1 \mapsto 1\right] \right|$$

is negligible (as a function of the security parameter). Moreover, for I_L/I_R as above, we write

$$Q_0 \equiv_{I_L/I_R} Q_1$$

if

$$\left|\mathsf{Pr}\left[Q_0 \mapsto 1, I_L/I_R\right] - \mathsf{Pr}\left[Q_1 \mapsto 1, I_L/I_R\right]\right|$$

is negligible, where $Q_0 \mapsto 1, I_L/I_R$ denotes the event that the run of the system Q_0 is both accepted and in I_L/I_R.

Lemma 2. *Let I_L and I_R be as above. For all polynomially bounded adversaries A we have*

$$A \parallel \bar{P} \equiv_{I_L/I_R} A \parallel \bar{P}_{Id(I_L,I_R)},$$

Proof of Lemma 2. The proof proceeds in two steps. First we show that, for the system $\bar{P}' = \bar{P}_{Id(I_L)}$, where only senders in I_L perform idealized encryption as described in item (a), we have $A \parallel \bar{P} \equiv_{I_L/I_R} A \parallel \bar{P}'$. Then we show that, $A \parallel \bar{P}' \equiv_{I_L/I_R} \bar{A} \parallel P''$ where $\bar{P}'' = \bar{P}_{Id(I_L,I_R)}$. Below, we only give the proof for the first step; the proof for the second one is very similar. So, we are given an adversary A as in the statement of the lemma. We have to prove that

$$A \parallel \bar{P} \equiv_{I_L/I_R} A \parallel \bar{P}'. \tag{10}$$

Recall that $\bar{P}$ and $\bar{P}'$ contain as their part the common reference string generator Setup, which is invoked in the setup phase of the system, before the actual protocol is executed. Also, the honest mix server M_j (which is part of $\bar{P}$ and $\bar{P}'$) calls, as its sub-procedure, the prover Prover for the zero-knowledge proofs. Recall that, as the definition of zero-knowledge requires (see Section A.3), these calls are only made for true statements, that is, statements $(x,w) \in R$ (in our case x are decryption statements and the witness w is the secret key of the honest mix server M_j, see Section 4.2). Note that this is true also in the idealized system $\bar{P}'$, as the zero-knowledge proofs of decryption are, by the construction of $\bar{P}'$, never required for the ciphertexts obtained by idealized encryption. (By the definition of $\bar{P}'$, in case a link involving idealized encryption is audited, the honest mix server M_j just fails, i.e., stops its execution, without invoking the prover.)

We now make the algorithms Setup and Prover explicit by representing $\bar{P}$ as the composition of Setup, Prover, and $\bar{P}_0$. Similarly, we can represent $\bar{P}'$ as the composition of Setup, Prover, and $\bar{P}'_0$:

$$A \parallel \bar{P} \;=\; A \parallel \mathsf{Setup} \parallel \mathsf{Prover} \parallel \bar{P}_0$$

and

$$A \parallel \bar{P}' \;=\; A \parallel \mathsf{Setup} \parallel \mathsf{Prover} \parallel \bar{P}'_0.$$

In the systems above, the common reference string σ produced by Setup is given to all the remaining components of the system (by the scheduler).

By the zero-knowledge property of the used proofs, we know that there exists a simulator (S_1, S_2) such that

$$A \parallel \mathsf{Setup} \parallel \mathsf{Prover} \parallel \bar{P}_0 \;\equiv\; A \parallel S_1 \parallel S'_2 \parallel \bar{P}_0 \tag{11}$$

and

$$A \parallel \mathsf{Setup} \parallel \mathsf{Prover} \parallel \bar{P}'_0 \;\equiv\; A \parallel S_1 \parallel S'_2 \parallel \bar{P}'_0. \tag{12}$$

where S_1 generates a simulated common reference string σ and a trap-door τ (again, σ is implicitly distributed to all parties; τ is given only to S'_2/S_2), and S'_2, when queried on (x,w), ignores the witness w and returns $S_2(\sigma,\tau,x)$. For the equivalence (12), we use, as mentioned before, that the honest mix server M_j simply fails without invoking Prover in case a link involving ideal encryption is audited.

It follows that

$$A \parallel \mathsf{Setup} \parallel \mathsf{Prover} \parallel \bar{P}_0 \;\equiv_{I_L/I_R}\; A \parallel S_1 \parallel S'_2 \parallel \bar{P}_0 \tag{13}$$

and, similarly,

$$A \parallel \mathsf{Setup} \parallel \mathsf{Prover} \parallel \bar{P}_0' \equiv_{I_L/I_R} A \parallel S_1 \parallel S_2' \parallel \bar{P}_0' \ , \tag{14}$$

as otherwise, since it is (computationally) easy to check if a given run conforms to I_L/I_R, one could break the zero-knowledge property of our ZK-proof system.

Now, let $C(b)$ denote the challenger from the IND-CCA2 game run with secret bit b. For simplicity of notation, we assume that C also generates the public/private key pair and provides the adversary with the public key. We will show how to construct a simulator S_{CCA2} which simulates $\bar{P}_0/\bar{P}_0'$ and uses $C(b)$ such that, when restricted to the event I_L/I_R, the system $S_{CCA2} \parallel C(1)$ behaves as $\bar{P}_0$ and $S_{CCA2} \parallel C(0)$ behaves as $\bar{P}_0'$. More precisely, we will have that

$$A \parallel S_1 \parallel S_2' \parallel \bar{P}_0 \equiv_{I_L/I_R} A \parallel S_1 \parallel S_2' \parallel S_{CCA2} \parallel C(1) \tag{15}$$

and

$$A \parallel S_1 \parallel S_2' \parallel \bar{P}_0' \equiv_{I_L/I_R} A \parallel S_1 \parallel S_2' \parallel S_{CCA2} \parallel C(0). \tag{16}$$

Given the above equivalence, which we will prove below, the proof concludes as follows. By the IND-CCA2 property of the used encryption scheme and the fact that S_{CCA2} never asks $C(b)$ for the decryption of ciphertexts it has received before from $C(b)$ (as will be clear from the construction of S_{CCA2}), we have that

$$A \parallel S_1 \parallel S_2' \parallel S_{CCA2} \parallel C(0) \equiv A \parallel S_1 \parallel S_2' \parallel S_{CCA2} \parallel C(1).$$

Similarly as above, it then follows that

$$A \parallel S_1 \parallel S_2' \parallel S_{CCA2} \parallel C(0) \equiv_{I_L/I_R} A \parallel S_1 \parallel S_2' \parallel S_{CCA2} \parallel C(1). \tag{17}$$

Now, by transitivity of $\equiv_{|I_L/I_R}$, we obtain by (15), (16), and (17) that

$$A \parallel S_1 \parallel S_2' \parallel \bar{P}_0 \equiv_{I_L/I_R} A \parallel S_1 \parallel S_2' \parallel \bar{P}_0'.$$

Together with (13) and (14), we now immediately obtain (10), as desired.

Construction of the simulator S_{CCA2}: The simulator generates the key pair sk_{2j}, pk_{2j}, that is, the keys of the honest mix server M_j used in the first mixing step. The keys sk_{2j+1}, pk_{2j+1} of M_j, used in the second mixing, are generated by the challenger C who will publish only the public key pk_{2k+1}. Recall that the keys of all remaining mix servers, which are assumed to be dishonest, are generated by A.

The simulator can easily simulate the actions taken by the honest senders not in I_L (recall that, in the considered systems, the adversary picks the input plaintext of the honest senders) and by the auditors. He can also easily simulate the first mixing step of the honest mix server (as he generates the keys for this step). The only two steps not covered yet are (1) generation of honest encrypted input for honest senders in I_L and (2) the second mixing step of the honest mix server. Simulation of these steps is described below.

Honest input generation To obtain the encrypted input of the (honest) senders in I_L, the simulator proceeds as follows. It first encrypts the input plaintexts of senders in I_L with the public keys $pk_{2m-1}, \ldots, pk_{2j+2}$. Let $\vec{y}_1$ denote the ciphertexts obtained in this way. Let $\vec{y}_0$ denote a vector (of the same length as $\vec{y}_1$) of strings of zeros of the same length as the corresponding messages in $\vec{y}_1$. The simulator sends these two vectors (one pair of a component of $\vec{y}_0$ and $\vec{y}_1$ at a time) to the challenger, who sends back an encryption $\vec{z}$ of $\vec{y}_b$ under the public key pk_{2j+1} (where b is the secret bit chosen by the challenger). Now, the simulator obtains the encrypted input $\vec{c}$ of senders in I_L by further encrypting the elements of $\vec{z}$ under $pk_{2j}, \ldots, pk_0$, in this order.

The second mixing of the honest mix server The input to the second mixing of the honest mix server contains elements of $\vec{z}$ (these are the entries of senders in I_L), plus possibly some other messages, e.g., entries of other honest and dishonest senders. Note that some of the elements in $\vec{z}$ might not make it to the honest mix server because the adversary might have dropped them or modified them before they reach the honest mix server. The simulator can simply ask the challenger to decrypt messages not in $\vec{z}$. The simulator is not allowed, however, to ask the challenger to decrypt messages in $\vec{z}$, as this would violate the rules of the IND-CCA2 game. Let us notice, however, that the simulator knows a priori the "decryptions" of messages in $\vec{z}$, as produced by M_j: these are, both for the ideal variant $\bar{P}'$ and for the real variant $\bar{P}$, the corresponding messages in $\vec{y}_1$. Altogether, the simulator obtains all decrypted messages for the second mixing step. The simulator then chooses a random permutation to rearrange these messages in order to obtain the output of this mixing step. In the audit phase, the simulator also produces the zero-knowledge proofs by calling S_2. Note that the simulator does not need the secret key sk_{2j+1} (which it does not know); S_2 can be invoked without this key. In the general case, the auditing might require the simulator to prove a link for the second mix server that is not actually true (because ideal encryption was used). The simulator can nevertheless invoke S_2, which will not output a reasonable proof. However, for (15) and (16) we only consider runs where auditing conforms to I_L/I_R. In such runs only true statements have to be proven.

By the construction of S_{CCA2}, it is easy to see that the systems $B = (A \parallel S_1 \parallel S_2' \parallel \bar{P}_0')$ and $B' = (A \parallel S_1 \parallel S_2' \parallel S_{CCA2} \parallel C(0))$ behave in exactly the same way except if they have to produce zero knowledge proofs for statements that are not true. In $\bar{P}_0'$, the mix server would fail, i.e., stop running, if it is asked to prove a statement that is not true. Conversely, S_{CCA2} in any case calls S_2'. However, for runs in I_L/I_R no wrong statements need to be proved. (In such runs, ideal encryption is only done for links that are not audited.) And hence, for runs in I_L/I_R the systems behave in exactly the same way. Note that whether or not a run is in I_L/I_R is determined before the honest mix server has to produce its proofs. Hence, a run of B belongs to I_L/I_R iff the corresponding run of B' belongs to I_L/I_R as well. (In particular, we have that $\mathsf{Pr}[B \mapsto I_L/I_R] = \mathsf{Pr}[B' \mapsto I_L/I_R]$.) Altogether, we obtain (16). Analogously, we obtain (15). This is even simpler. The equivalence (15) in fact holds true in general not only for those runs in I_L/I_R, because no ideal encryption is used. □

We also state a similar result for the case where the adversary does not get to see (or simply ignores) the proofs of correct decryption produced by the honest mix server M_j. In this case, invalid zero-knowledge proofs may be produced and it is hence irrelevant at which steps the honest senders perform the idealized encryption. Therefore, we can assume that the idealized encryption is always used by an honest sender with index i to produce, say, α^i_{2j+1} (which corresponds to the second mixing step of the honest mixer M_j). We will denote such a system by $\bar{P}_{Id}$. In this system, the senders perform idealized encryption as just described, the adversary determines the plaintext input of the honest senders, and auditors do not check zero-knowledge proofs produced for auditing M_j; in particular, they do not stop the run of the system even if one of these proofs is flawed. (But all other proofs are checked.) Now, similar to Lemma 2, we obtain the following result.

Lemma 3. *For an adversary A that ignores zero-knowledge proofs produced by* M_j *or does not receive them, we have* $A \parallel \bar{P} \equiv A \parallel \bar{P}_{Id}$.

The proof of this lemma is very similar to the proof of Lemma 2. It is actually simpler as we do not need to use the zero-knowledge property of the used proof system (as the proofs are ignored and can be simply omitted). Lemma 3 allows us to reason, for instance, about the decisions the adversary makes before M_j outputs its data.

As an immediate corollary of Lemma 3, we obtain:

Corollary 2. *For an adversary A that ignores zero-knowledge proofs produced by* M_j *or does not receive them, we have* $A \parallel P \equiv A \parallel P_{Id}$.

We will consider one more variant of the above result. This time, we consider the system P^o_{Id} which is defined as the systems P, except that the idealized encryption is performed by the sender under observation (and only by this sender) for the second mixing step of the honest mix server, that is, when the key pk_{2j+1} is used for encryption. (For the following result, it actually does not matter whether the idealized encryption is performed at the first or the second mixing step). As before, we will also consider the variant $\bar{P}^o_{Id}$ of this system.

The following result states that as long as the entry of the sender under observation is not delivered to the honest mix server, the original system P and the system P^o_{Id} are indistinguishable.

Lemma 4. *For all polynomially bounded adversaries A we have*

$$A \parallel \bar{P} \equiv_{ND} A \parallel \bar{P}^o_{Id},$$

where ND denotes the event that the entry of the sender under observation is not delivered to the honest mix server.

Again, the proof of this result is very similar to the proof of Lemma 2. Note that, if the event *ND* holds, the entry of the sender under observation is not decrypted by M_j and therefore not audited (no zero-knowledge proof is requested for it).

Now, we state and prove a result that is used both in the proof of Theorem 2 and Theorem 4. Let p and p' be valid plaintexts and A be an arbitrary (not necessarily risk-avoiding) program of the adversary that, intuitively, tries to distinguish whether the sender under observation has chosen p or p'. From the program A, we derive a program A^* in the following way: A^* simulates A up to and including the point where the honest mix server M_j produces its output. At this point, we consider two cases: (1) If the entry α^0_{2j} of the sender under observation is not in the input of M_j, A^* flips a coin to determine its decision. (2) Otherwise, A^* decrypts the output of M_j (recall that the adversary subsumes all the mix servers but M_j and so he knows all the necessary keys). In particular A^* obtains the multiset Q of all plaintexts that have been chosen by the honest senders from the audit group to which the sender under observation belongs, including the sender under observation. Now, A^* accepts the run (outputs 1) if and only if the following is true: the probability that the choices of $|Q|-1$ honest senders (making their choices according to the probability distribution μ) yield Q, given that the sender under observation chooses p, is bigger than the probability that the choices of $|Q|-1$ honest senders yield Q, given that the sender under observation chooses p'.

Now, we can prove the following result, where $\mathsf{Adv}^{priv}_{A,P,p,p'}(\ell)$ denotes the advantage of the adversary A interacting with P in distinguishing whether the sender under observation uses the plaintext p or p', i.e.,

$$|\mathsf{Pr}[(A \parallel P(p))^{(\ell)} \mapsto 1] - \mathsf{Pr}[(A \parallel P(p'))^{(\ell)} \mapsto 1]| \ , \tag{18}$$

where $P(p)$ means that the sender under observation uses p as its plaintext. For two functions f and f' in ℓ by $f \leq_{neg} f'$ we mean that there exists a negligible function $\nu(\ell)$ such that $f(\ell) \leq f'(\ell) + \nu(\ell)$ for all ℓ. Intuitively, the result says that the advantage of A is not bigger than the advantage of A^*.

Lemma 5. *For all valid p and p' we have that*

$$\mathsf{Adv}^{priv}_{A,P,p,p'} \leq_{neg} \mathsf{Adv}^{priv}_{A^*,P,p,p'} \ .$$

The proof of this lemma is given in Section C.2, it uses Lemmas 2, 3, and 4 as well as Corollary 2.

To prove Theorem 2, we also need the following property about risk-avoiding adversaries.

Proposition 1. *Let $A \parallel P$ be the system as defined above and assume that A is risk-avoiding. Let, as in Corollary 1, X_j denote the event that in a run of $A \parallel P$ in the input C_{2j} for M_j not all entries of honest senders are contained, i.e., at least one is missing. Then, $\mathsf{Pr}[X_j]$ is negligible (as a function of the security parameter).*

Proof. Let B denote the event that the judge outputs $\mathsf{dis}(\mathsf{M}_i)$ for some dishonest agent, i.e., the judge blames some dishonest agent.

Since A is risk-avoiding, we know that $\mathsf{Pr}[B]$ is negligible. We have that

$$\begin{aligned}
\mathsf{Pr}\left[B\right] &= \mathsf{Pr}\left[B, X_j\right] + \mathsf{Pr}\left[B, \bar{X}_j\right] \\
&\geq \mathsf{Pr}\left[B, X_j\right] \\
&\geq \mathsf{Pr}\left[G, B, X_j\right] \\
&\geq \mathsf{Pr}\left[B \mid G, X_j\right] \cdot \mathsf{Pr}\left[G, X_j\right], \\
&\geq \frac{1}{4} \cdot \mathsf{Pr}\left[G, X_j\right]
\end{aligned}$$

provided that $\mathsf{Pr}[G, X_j] > 0$, where G is defined as in Appendix B. The last inequality is obtained by Corollary 1. Note that if $\mathsf{Pr}[G, X_j] = 0$, then $\mathsf{Pr}\left[B\right] \geq \frac{1}{4} \cdot \mathsf{Pr}\left[G, X_j\right]$ is true as well. Since $\mathsf{Pr}\left[B\right]$ is negligible, it follows that $\mathsf{Pr}\left[G, X_j\right]$ is negligible too. Since $\mathsf{Pr}\left[G\right]$ is overwhelming (see Appendix B), it, in particular, follows that $\mathsf{Pr}\left[X_j\right]$ is negligible. □

Now, the proof of Theorem 2 proceeds as follows. Since we consider a risk-avoiding adversary A, by Proposition 1 we know that, except in negligible cases, all the entries of the honest senders and of the sender under observation make it to M_j. By Lemma 5, we can consider A^* instead of A (the advantage of A^* is not less than the one of A). It is easy to see that the computations carried out by A^* yield the constant from the theorem, as explained in Section 6.3 and 5.2. By this, we can conclude that no risk-avoiding adversary can achieve higher advantage.

C.1 Proof of Theorem 4

It is easy to see that the constant $\delta^*_{l,\mu}$ defined in Theorem 4 is optimal, assuming that the number of mix servers preceding the honest mix server is sufficiently big in order to drop l honest entries following the optimal strategy: In fact, let $l' \in \{0, \ldots, l\}$ such that $\delta^*_{l,\mu} = \left(\frac{3}{4}\right)^{l-l'} \cdot \delta_{l',\mu}$. Now, let A be some adversary such that l' honest entries reach M_j following the optimal strategy (in particular, $l - l'$ honest entries are dropped). Let A^* be the adversary obtained from A as in Lemma 5. Then, just as in the proof of Theorem 2, it is easy to see that the advantage of A^* is $\left(\frac{3}{4}\right)^{l-l'} \cdot \delta_{l',\mu}$.

Now, we show that the advantage for every venturesome adversary A is bounded by $\delta^*_{l,\mu}$.

By Lemma 5, it suffices to consider adversaries A such that $A = A^*$. By Corollary 2 (note that we can apply this lemma to A, as $A = A^*$ ignores the decryption proofs), one can use, instead of P, the idealized protocol $P' = P_{Id}$.

Now, besides the advantage $\mathsf{Adv}^{priv}_{A,P',p,p'}$, where A interacts with P', we also use the notion of *restricted advantage* $\mathsf{Adv}^{priv}_{A,P',p,p'}[\alpha]$, for an event α, which is defined as

$$|\mathsf{Pr}[(A \parallel P'(p)) \mapsto 1, \alpha] - \mathsf{Pr}[(A \parallel P'(p')) \mapsto 1, \alpha]|.$$

Moreover, we will use the maximal (restricted) advantage of the adversary, where the maximum is taken over all possible input plaintexts p, p':

$$\mathsf{Adv}^{priv}_{A,P'} = \max_{\text{valid } p,p'} \mathsf{Adv}^{priv}_{A,P',p,p'}$$

In the following, we will write simply Adv instead of $\mathsf{Adv}^{priv}_{A,P'}$.

Our goal is to show that Adv is $\delta^*_{l,\mu}$-bounded. For this, it is enough to show that

$$\mathsf{Adv}[G] \leq \delta^*_{l,\mu},$$

where G is as defined in Appendix B (recall that the probability for G is overwhelming).

Now, one can see that

$$\mathsf{Adv}[G, \text{blamed before } \mathsf{M}_j] = 0,$$

where "blamed before M_j" denotes the event that one of the mix servers is blamed before the honest mix server gets to process its input. This is because, in the ideal system, the runs considered here (i.e. the runs for which, in particular, the event "blamed before M_j" holds) are completely independent on the choices of honest senders and the sender under observation. Therefore

$$\mathsf{Adv}[G] \leq \mathsf{Adv}[G, \text{not blamed before } \mathsf{M}_j]$$

For similar reasons,

$$\mathsf{Adv}[G, \text{not blamed before } \mathsf{M}_j, \overline{D}] = 0,$$

where $\overline{D}$ denotes the event that the entry of the sender under observation is not delivered to the honest mix server (by D we denote the complementary event). Therefore,

$$\mathsf{Adv}[G] \leq \mathsf{Adv}[G, \text{not blamed before } \mathsf{M}_j, D].$$

In what follows, we assume that $\mathsf{Pr}[G, D] > 0$ (for the considered security parameter). Otherwise, we trivially have that $\mathsf{Adv}[G] = 0$, and hence, that $\mathsf{Adv}[G] \leq \delta^*_{l,\mu}$.

Let L be the set of vectors $\vec{l} = (l_0, \ldots, l_{j-1})$, where $l_i \in \{0, \ldots, l\}$. As in the proof of Theorem 1 (Appendix B), we will represent by l^*_i the event that the i-th mix server produces an output containing exactly l_i honest entries and by l_i we will represent the event that, additionally, this mix server is not blamed. Now, $\vec{l} = l_0, \ldots, l_{j-1}$ represents the event that no mix server before M_j is blamed and, for each $j' \in \{0, \ldots, j-1\}$, the number of honest entries in $C_{2j'+2}$ is $l_{j'}$, which means that, in particular, the number of honest entries delivered to the honest mix server M_j is l_{j-1}. Let us notice that the event "not blamed before M_j" holds, if and only an event $\vec{l}$ holds, for some $\vec{l} \in L$.

Therefore, we can rewrite the above inequality as

$$\mathsf{Adv}[G] \leq \sum_{\vec{l} \in L^*} \mathsf{Adv}[G, D, \vec{l}], \tag{19}$$

where $L^* = \{\vec{l} \in L \mid \mathsf{Pr}[\vec{l} \mid G, D] \neq 0\}$. From this, we obtain that

$$\mathsf{Adv}[G] \leq \sum_{\vec{l} \in L^*} \Big(\mathsf{Pr}[\vec{l}, G, D] \cdot |\mathsf{Pr}[(A \parallel P'(p)) \mapsto 1 \mid G, D, \vec{l}] - \mathsf{Pr}[(A \parallel P'(p')) \mapsto 1 \mid G, D, \vec{l}]| \Big).$$

Now, one can show that

$$|\mathsf{Pr}[(A \parallel P'(p)) \mapsto 1 \mid G, D, \vec{l}] - \mathsf{Pr}[(A \parallel P'(p')) \mapsto 1 \mid G, D, \vec{l}]| \leq \delta_{l_{j-1},\mu}.$$

To see this, first note that in the system P' the choices of honest senders (i.e. the plaintext input of these senders) are stochastically independent of the actions taken by the adversary before M_j produces its output. Moreover, we know that $A = A^*$. That is, the result that A^* outputs depends only on the content of the multi set Q, i.e. the multi set of input plaintexts sent by the senders belonging to the same audit group to which the sender under observation belongs. Now, the condition $G, D, \vec{l}$ says that l_{j-1} entries of honest senders plus the entry for the sender under observation reaches M_j. We could imagine that the plaintexts for these entries are chosen by the honest senders only when they reach M_j (because of the their independence to the actions of the adversary). Hence, the situation resembles the one for the risk-avoiding adversary with l_{j-1} honest senders. (This adversary did not have a choice but deliver all honest entries to the last mix server.) Similarly to this case, we therefore obtain the above inequality.

Therefore, we have that

$$\begin{aligned}
\mathsf{Adv}[G] &\leq \sum_{\vec{l} \in L^*} \mathsf{Pr}[\vec{l}, G, D] \cdot \delta_{l_{j-1},\mu} \\
&= \sum_{\vec{l} \in L^*} \mathsf{Pr}[G, D] \cdot \mathsf{Pr}[\vec{l} \mid G, D] \cdot \delta_{l_{j-1},\mu} \\
&\leq \sum_{\vec{l} \in L^*} \mathsf{Pr}[\vec{l} \mid G, D] \cdot \delta_{l_{j-1},\mu} \\
&= \sum_{\vec{l} \in L^*} \begin{array}{l} \mathsf{Pr}[l_0^* \mid G, D] \cdot \mathsf{Pr}[l_0 \mid G, D, l_0^*] \cdot \\ \mathsf{Pr}[l_1^* \mid G, D, l_0] \cdot \mathsf{Pr}[l_1 \mid G, D, l_0, l_1^*] \cdots \\ \quad \cdots \mathsf{Pr}[l_{j-1}^* \mid G, D, l_0, \ldots, l_{j-2}] \cdot \\ \qquad \cdot \mathsf{Pr}[l_{j-1} \mid G, D, l_0, \ldots, l_{j-2}, l_{j-1}^*] \\ \quad \cdot \delta_{l_{j-1},\mu} \end{array} \\
&= \sum_{\vec{l} \in L^*} \begin{array}{l} \mathsf{Pr}[l_0^* \mid G, D] \cdot \mathsf{Pr}[l_1^* \mid G, D, l_0] \cdots \\ \qquad \cdots \mathsf{Pr}[l_{j-1}^* \mid G, D, l_0, \ldots, l_{j-2}] \cdot \\ \mathsf{Pr}[l_0 \mid G, D, l_0^*] \cdot \mathsf{Pr}[l_1 \mid G, D, l_0, l_1^*] \cdots \\ \qquad \cdots \mathsf{Pr}[l_{j-1} \mid G, D, l_0, \ldots, l_{j-2}, l_{j-1}^*] \\ \quad \cdot \delta_{l_{j-1},\mu} \end{array}
\end{aligned}$$

Using a similar argument as in the proof of Theorem 1, we obtain that

$$\mathsf{Pr}[l_0 \mid G, D, l_0^*] \cdot \mathsf{Pr}[l_1 \mid G, D, l_0, l_1^*] \cdots \mathsf{Pr}[l_{j-1} \mid G, D, l_0, \ldots, l_{j-2}, l_{j-1}^*] \leq (\tfrac{3}{4})^{k'} \leq (\tfrac{3}{4})^{l - l_{j-1}}$$

where $k_i = \max(0, l_{i-1} - l_i)$ and $k' = k_0 + \cdots + k_{j-1}$. Therefore, by the definition of $\delta_{l,\mu}^*$, we obtain that

$$\begin{aligned}
\mathsf{Adv}[G] &\leq \sum_{\vec{l} \in L^*} \begin{array}{l} \mathsf{Pr}[l_0^* \mid G, D] \cdot \mathsf{Pr}[l_1^* \mid G, D, l_0] \cdots \\ \qquad \cdots \mathsf{Pr}[l_{j-1}^* \mid G, D, l_0, \ldots, l_{j-2}] \cdot \\ \quad \cdot \left(\tfrac{3}{4}\right)^{l - l_{j-1}} \cdot \delta_{l_{j-1},\mu} \end{array} \\
&\leq \sum_{\vec{l} \in L^*} \begin{array}{l} \mathsf{Pr}[l_0^* \mid G, D] \cdot \mathsf{Pr}[l_1^* \mid G, D, l_0] \cdots \\ \qquad \cdots \mathsf{Pr}[l_{j-1}^* \mid G, D, l_0, \ldots, l_{j-2}] \cdot \\ \quad \cdot \delta_{l,\mu}^* \end{array} \\
&\leq \delta_{l,\mu}^* \ , \qquad (20)
\end{aligned}$$

where, as in Appendix B, it is easy to see that the sum is bounded by 1. This concludes the proof.

C.2 Proof of Lemma 5

In this proof, we will write $f \equiv f'$, if f and f' are functions in the security parameter ℓ such that $|f(\ell) - f'(\ell)|$ is a negligible function in ℓ.

We will use the convention that the choice of the sender under observation depends on a bit b: if $b = 0$, this sender uses p, if $b = 1$, he uses p'. This bit is chosen with uniform probability. Using this convention, we have to show that

$$\mathsf{Pr}[A \parallel P \mapsto b] \leq_{negl} \mathsf{Pr}[A^* \parallel P \mapsto b], \tag{21}$$

where $(A \parallel P \mapsto b)$ denotes the event that A, interacting with P, correctly guesses the bit b, and similarly for $A^* \parallel P \mapsto b$.

Aborted runs. First, let us consider the case that the protocol is aborted (because some misbehavior of a dishonest mix net has been detected) before the honest mix server M_j takes its actions.

We show that, in this case, the advantage of A is identical (up to some negligible function) to the advantage of A^*, i.e.,

$$\mathsf{Pr}[(A \parallel P \mapsto b) \wedge \mathsf{aborted}] \equiv \mathsf{Pr}[(A^* \parallel P \mapsto b) \wedge \mathsf{aborted}], \tag{22}$$

where $\mathsf{aborted}$ denotes the event that a run is aborted before M_j takes its actions. Note that these events are essentially the same independently of whether we consider the system $A \parallel P$ or $A^* \parallel P$. This is because initially A^* simply simulates A, and it is only after the protocol has been aborted or after M_j has taken its actions that the behavior of A and A^* may diverge. If the event $\mathsf{aborted}$ occurs, the adversary does not get to see the output of the honest mix server and, therefore, we can use Corollary 2 and replace P by its idealized counterpart $P' = P_{Id}$. More precisely, by Corollary 2 we obtain that

$$\mathsf{Pr}[(A \parallel P \mapsto b) \wedge \mathsf{aborted}] \equiv \mathsf{Pr}[(A \parallel P' \mapsto b) \wedge \mathsf{aborted}].$$

To see this, we consider the adversary A' that picks a random bit b to determine the choice of the sender under observation and then simulates A. Finally, A' outputs the following decision: if the run is not aborted, then A' outputs 0; otherwise, A' outputs 1 if and only if A correctly guesses the bit b (that is A outputs b). One can see that if the above equivalence was not true, then A' would break Corollary 2. In the similar way we conclude that

$$\mathsf{Pr}[(A^* \parallel P \mapsto b) \wedge \mathsf{aborted}] \equiv \mathsf{Pr}[(A^* \parallel P' \mapsto b) \wedge \mathsf{aborted}].$$

Now, (22) follows from

$$\mathsf{Pr}[(A \parallel P' \mapsto b) \wedge \mathsf{aborted}] = \mathsf{Pr}[(A^* \parallel P' \mapsto b) \wedge \mathsf{aborted}].$$

This equality is easy to prove, because, in this case, by the definition of the idealized system P', the view of the adversary is completely independent of the choice of the sender under observation and, thus, of b.

Runs, where the entry of the sender under observation is not delivered. Now we prove that

$$\mathsf{Pr}[(A \parallel P \mapsto b) \wedge ND] \equiv \mathsf{Pr}[(A^* \parallel P \mapsto b) \wedge ND], \tag{23}$$

where ND, as already introduced, denotes the event that the entry of the sender under observation is not delivered to the honest mix server. Note that, again, this event is essentially the same independent of whether we consider the system with A or with A^*, as these system diverge only after this event is determined.

This fact follows immediately from Lemma 4 and the equality

$$\mathsf{Pr}[(A \parallel P' \mapsto b) \wedge ND] = \mathsf{Pr}[(A^* \parallel P' \mapsto b) \wedge ND],$$

where $P' = P^o_{Id}$. This equality holds true because, in the system P', where the sender under observation performs idealized encryption (for M_j) and the entry of this sender is not decrypted by M_j, the view of the adversary is completely independent of the choice of this sender.

Not aborted runs, where the entry of the sender under observation is delivered. Let $(0 \in I_L)$ denotes the event that the run is not aborted and the sender under observation (who has index 0) belongs to the left audit group. Similarly, let $(0 \in I_R)$ denotes the event that the run is not aborted and the sender under observation belongs to the right audit group. Note that the events $(0 \in I_L)$ and $(0 \in I_R)$ are essentailly the same independently whether we consider the system $A \parallel P$ or $A^* \parallel P$. Note also that both $(0 \in I_L)$ and $(0 \in I_R)$ imply that the entry of the sender under observation is delivered.

By (22) and (23), to complete the proof, it is enough to show that both

$$\mathsf{Pr}[(A \parallel P \mapsto b), (0 \in I_L)] \leq_{negl} \mathsf{Pr}[(A^* \parallel P \mapsto b), (0 \in I_L)]$$

and

$$\mathsf{Pr}[(A \parallel P \mapsto b), (0 \in I_R)] \leq_{negl} \mathsf{Pr}[(A^* \parallel P \mapsto b), (0 \in I_R)]$$

hold true. We will consider the first case. The proof for the second case is very similar.

Let us now consider events of the form

$$X = (0 \in I_L), I_L/I_R, Q,$$

where $(0 \in I_L)$ is as defined above and

- (I_L/I_R) denotes the event that the set of honest senders, including the sender under observation, that are in the left audit group is I_L and the set of the honest senders in the right audit group is I_R.
- Q denotes the event that the multiset of plaintexts chosen by the senders in I_L is Q. (By abuse of notation, Q denotes both a multiset of plaintexts and an event.)

Note also that the events X, $(0 \in I_L)$, I_L/I_R, and Q, are essentially the same independently of whether we consider the system $A \parallel P$ or the system $A^* \parallel P$ (the systems A and A^* diverge only when all those events are determined).

Note that for different choices of the index sets I_L and I_R and the multiset Q of plaintexts, one obtains a different event X. In this sense, X denotes an element of a family of events of the described form. Note that the number of elements in this family is fix and finite and independent of the security parameter. (Clearly, the events themselves depend on the security parameter.) Therefore, to complete the proof, it is enough to show that, for all X of non-negligible probability,

$$\mathsf{Pr}[(A \parallel P \mapsto b),\ X] \leq_{negl} \mathsf{Pr}[(A^* \parallel P \mapsto b),\ X] \tag{24}$$

The rest of this section is devoted to proving (24) for a fixed event X of non-negligible probability.

The event Q determines possible vectors $z_1, \ldots z_r$ of (plaintext) input messages of senders in I_L (including the sender under observation), that yield Q. Note that the length of each z_i is $|I_L| = |Q|$. We will denote the collection of these vectors by Z_Q. By abuse of notation, each $z \in Z_Q$ is interpreted to be an event containing all runs where the senders in I_L chose their plaintexts according to (the vector) z. Again, the event z is defined independently of whether we consider the system $A \parallel P$ or the system $A^* \parallel P$.

The main technical result used in this proof is the following lemma.

Lemma 6. *For each $z \in Z_Q$, we have*

$$\mathsf{Pr}[(A \parallel P \mapsto 1), X \mid z] \equiv \mathsf{Pr}[(A \parallel P \mapsto b), X \mid Q]. \tag{25}$$

Proof. The sample space Ω can be represented as $\Omega_1 \times \Omega_2$, where $\omega_1 \in \Omega_1$ are random coins used by the senders in I_L to determine their choices and $\omega_2 \in \Omega_2$ are the remaining random coins. We will show that, for all $\omega_1, \omega_1' \in \Omega_1$ we have

$$\mathsf{Pr}[(A \parallel P' \mapsto 1), X \mid \omega_1] = \mathsf{Pr}[(A \parallel P' \mapsto 1), X \mid \omega_1'], \tag{26}$$

where $P' = P_{Id(I_L/I_R)}$. From this it follows by Lemma 2 that

$$\mathsf{Pr}[(A \parallel P \mapsto 1), X \mid \omega_1] \equiv \mathsf{Pr}[(A \parallel P \mapsto 1), X \mid \omega_1']. \tag{27}$$

Indeed,

$$\mathsf{Pr}[(A \parallel P \mapsto 1), X \mid \omega_1] = \mathsf{Pr}[(A' \parallel \bar{P} \mapsto 1), X]$$

and

$$\mathsf{Pr}[(A \parallel P' \mapsto 1), X \mid \omega_1] = \mathsf{Pr}[(A' \parallel \bar{P}' \mapsto 1), X]$$

where A' is the adversary for $\bar{P}$ (and for $\bar{P}'$) who uses the fixed coins ω_1 to determine the choices of the senders in I_L, determines the choices of the remaining honest senders according to the probability distribution μ and then simulates A. Now, it must be that

$$\mathsf{Pr}[(A' \parallel \bar{P} \mapsto 1), X] \equiv \mathsf{Pr}[(A' \parallel \bar{P}' \mapsto 1), X]$$

because otherwise we would have

$$\mathsf{Pr}[(A'' \parallel \bar{P} \mapsto 1), I_L/I_R] \not\equiv \mathsf{Pr}[(A'' \parallel \bar{P}' \mapsto 1), I_L/I_R]$$

where A'' is the adversary that works as A', but outputs 0 whenever the run is not in X, which would constradict Lemma 2. Note that A'' can decide whether a run belongs to X or not because A'' controls all mix servers up to M_j (and all following M_j). In particular, A'' knows all private keys of these mix servers, and hence, can determine which ciphertext in the input to M_j belongs to which honest sender. This proves (27).

From (27), the equivalence (25) easily follows, because

$$\begin{aligned}
\mathsf{Pr}[(A \parallel P \mapsto 1), X \mid z] &= \sum_{\omega_1 \in z} \mathsf{Pr}[\omega_1 \mid z] \cdot \mathsf{Pr}[(A \parallel P \mapsto 1), X \mid \omega_1] \\
&\equiv \mathsf{Pr}[(A \parallel P \mapsto 1), X \mid \omega_1^0] \cdot \sum_{\omega_1 \in z} \mathsf{Pr}[\omega_1 \mid z] \\
&= \mathsf{Pr}[(A \parallel P \mapsto 1), X \mid \omega_1^0],
\end{aligned}$$

where $\omega_1 \in z$ means that ω_1 yields the vector of choices z and ω_1^0 is some element such that $\omega_1^0 \in z$, and similarly one can show that

$$\mathsf{Pr}[(A \parallel P \mapsto 1), X \mid Q] \equiv \mathsf{Pr}[(A \parallel P \mapsto 1), X \mid \omega_1^0]$$

where $\omega_1^0 \in z$ for some $z \in Z_Q$. Hence, to complete the proof, we need to prove (26).

To prove (26), we use the assumption that the commitment scheme is perfectly hiding. We will show that, for the system $A \parallel P'$, the view of A is exactly the same for ω_1 and ω_1', if we change accordingly (a) random coins used to generate the second permutation of M_j, (b) random coins used to generate commitments used in this step, and (c) random coins used to

generate encryption of honest senders. From this, (26) immediately follows. This reasoning is formalized as follows.

Let z and z' be the vectors of choices determined, respectively, by ω_1 and ω'_1. In general, z' is a permutation of z. It is, however, enough to consider the case where z' is obtained from z by swapping two elements, say, the choices of the first and second sender. We will consider this case.

Let

$$B = \{\omega_2 : (A \parallel P')^{(\omega_1,\omega_2)} \mapsto 1, X\}$$

and

$$B' = \{\omega'_2 : (A \parallel P')^{(\omega'_1,\omega'_2)} \mapsto 1, X\},$$

where $(A \parallel P')^{\omega} \mapsto 1, X$ means that the run of the system $(A \parallel P')$ using random coins ω is accepted by A and falls into X. We will show that B and B' are of the same size, by constructing a bijection f from B to B'. We define $f(\omega_2) = \omega'_2$ in the following way, where ω'_2 coincides with ω_2 except for the following changes, and using the notation $\omega = (\omega_1, \omega_2)$ and $\omega' = (\omega'_1, \omega'_2)$:

– We swap in ω'_2 the random coins used by the first and the second sender to encrypt their messages with the keys $pk_{2m-1}, \ldots, pk_{2j+2}$, that is *before* the idealized encryption takes place (by this not only their plaintexts are swapped but also their encrypted entries α^1_{2j+2} and α^2_{2j+2} that will be output by M_j; note however that the result of the idealized encryption, which is, up to the length, independent of its input, is *not* changed, i.e. is the same for ω and ω', and thus the encrypted input also stays the same).

– The random coins in ω'_2 used by the honest mix server M_j in the second mixing step to generate permutations (π_{2j+1} for ω and π'_{2j+1} for ω') are changed in such a way that the output of this mix server is the same for ω and for ω'.

  This is possible, as the view of the adversary, before M_j is called is the same for ω and ω' and, therefore, the input to M_j (determined by the adversary) is exactly the same in both cases. Moreover, by the previous item, the output of M_j will be the same up to the order of the elements (more specifically, the elements α^1_{2j+2} and α^2_{2j+2} would be swapped, if we used the same permutation; the change in the permutation simple swaps these elements back).

– Now, let c denotes the commitment on the permutation π_{2j+1} as given for ω. By the assumption that the used commitment scheme is perfectly hiding, we know that, for c, π_{2j+1} and π'_{2j+1} as above, there exists a bijection h from the set of random coins r such that $\mathsf{comm}(\pi_{2j+1}, r) = c$ to the set of random coins r' such that $\mathsf{comm}(\pi'_{2j+1}, r') = c$.

  To obtain ω' from ω, we take the random coins r of ω used to produce the commitment c on the permutation π_{2j+1}, and replace r by $h(r)$ (note that $h(r)$ will produce the same commitment c on π'_{2j+1}).

One can see that f is indeed a bijection from B to B'. In particular, one can see that the view of the adversary for ω and ω' is exactly the same and, hence, the decision the adversary outputs is the same. Hence B and B' are of the same size, which implies (26). □

Let us remark that the Lemma 6 could also be proven if we used computationally hiding commitments. Proving this would, however, require a more complex reduction argument.

Because, the above lemma works for any adversary A, it also works for A^*. Therefore we obtain:

Corollary 3. *For each $z \in Z_Q$, we have*

$$\mathsf{Pr}[(A^* \parallel P \mapsto 1), X \mid z] \equiv \mathsf{Pr}[(A^* \parallel P \mapsto b), X \mid Q].$$

Note that the decision of A^*, by definition, is based solely and deterministically on Q and therefore this decision is the same for all runs in X. Let us assume that A^* outputs 1 for all runs in X; the proof in the case where it outputs 0 is analogous. In this case, by the definition of A^*, we know that

$$\mathsf{Pr}[b = 0 \mid Q] \leq \mathsf{Pr}[b = 1 \mid Q]$$

and hence

$$\mathsf{Pr}[b = 0, Q] \leq \mathsf{Pr}[b = 1, Q] \tag{28}$$

Note that the events $(b = 0)$, $(b = 1)$, and Q, used in the above probabilities, are defined independently of whether we consider A or A^*: besides by the bit b, they are determined solely by the random coins used by the senders in the set I_L (which is a fixed set of indices) to determine their choices.

In the following, we denote by Z_0 the set of those elements z in Z_Q (recall that z is a vector determining the choices of senders in I_L, including the choice of the sender under observation, compatible with Q; recall also that the choice of the sender under observation may be either p or p') for which the choice of the sender under observation is p. Similarly, we denote by Z_1 the set of those elements z in Z for which the choice of the sender under observation is p'. Note that by the definitions of Z_0 and Z_1 we have that

$$\sum_{z \in Z_0} \mathsf{Pr}[z] = \mathsf{Pr}[b = 0, Q] \tag{29}$$

and

$$\sum_{z \in Z_1} \mathsf{Pr}[z] = \mathsf{Pr}[b = 1, Q]. \tag{30}$$

Note that, again, all the events used in the above probabilities are defined independently of whether we consider A or A^*. Now, using (28), (29), (30), and Lemma 6, we obtain

$$\begin{aligned}
\mathsf{Pr}[(A \parallel P \mapsto b), X] = & \sum_{z \in Z_Q} \mathsf{Pr}[z] \cdot \mathsf{Pr}[(A \parallel P \mapsto b), X \mid z] \\
= & \sum_{z \in Z_0} \mathsf{Pr}[z] \cdot \mathsf{Pr}[(A \parallel P \mapsto 0), X \mid z] \\
& + \sum_{z \in Z_1} \mathsf{Pr}[z] \cdot \mathsf{Pr}[(A \parallel P \mapsto 1), X \mid z] \\
\equiv & \ \mathsf{Pr}[b = 0, Q] \cdot \mathsf{Pr}[(A \parallel P \mapsto 0), X \mid Q] \\
& + \mathsf{Pr}[b = 1, Q] \cdot \mathsf{Pr}[(A \parallel P \mapsto 1), X \mid Q] \\
\leq_{negl} & \ \mathsf{Pr}[b = 1 \mid Q] \cdot \mathsf{Pr}[X \mid Q]
\end{aligned}$$

(Note that the probability of z is bigger than 0, for all security parameters, and therefore the conditional probabilities above are always well defined). Similarly, using Corollary 3, we obtain:

$$\begin{aligned}
\mathsf{Pr}[(A^* \parallel P \mapsto b), X] \equiv & \ \mathsf{Pr}[b = 0, Q] \cdot \mathsf{Pr}[(A^* \parallel P \mapsto 0), X \mid Q] \\
& + \mathsf{Pr}[b = 1, Q] \cdot \mathsf{Pr}[(A^* \parallel P \mapsto 1), X \mid Q] \\
= & \ \mathsf{Pr}[b = 1 \mid Q] \cdot \mathsf{Pr}[X \mid Q]
\end{aligned}$$

For the last equation we use the assumption made above that, for the runs in X, the adversary A^* always outputs 1. Combining the above results, we obtain (24).