Formal Analysis of Chaumian Mix Nets with Randomized Partial Checking

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

Tomasz Truderung
University of Trier, Germany
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.

***Keywords*-Mix Nets; Random Partial Checking; Cryptographic Analysis; Privacy; Verifiability; Accountability**

I. 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 $\mathsf{M}_0,\ldots,\mathsf{M}_{m-1}$ 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 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* [18]. 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., [21], [17], [7], [23], [9], [10], some of which have been broken [22]). Most of the constructions are quite complex.

A particularly simple and efficient construction is the one proposed by Jakobsson, Juels, and Rivest [9]. 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 [9], 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 [20]. Some systems, such as Scantegrity [4], have used a similar technique.

In [11], Kahazei and Wikström have pointed out several severe attacks on RPC mix nets as described in the original work [9] 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 [13], [14], 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 [14]. 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 the 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 would only cheat if the risk of being caught is small.

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 [19]. (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. Interestingly, in the more realistic case of risk-avoiding adversaries (which do not want to be caught cheating with a probability bigger than, say, 25% or even 75%, and hence, which are still willing to take big risks), the level of privacy for post-phase auditing is still very close to the ideal case.

We note that there has been no rigorous formal analysis of RPC mix nets before. In particular, none that provides formal security guarantees for privacy or verifiability/accountability. As mentioned, Kahazei and Wikström [11] 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 [11], this result does not capture privacy or verifiability of RPC mix nets.

Structure of this paper. In Section II, we describe Chaumian RPC mix nets and present our formal model. Accountability and verifiability for such mix nets are defined in Section III, with the formal analysis presented in Section IV. Our notion of privacy is introduced in Section V. The formal analysis of the privacy of Chaumian RPC mix nets is then provided in Section VI. We conclude in Section VII. Further details are provided in the appendix. We point the reader to [15] for the full version of this paper.

II. Chaumian RPC Mix Net

In this section, we recall the definition of a Chaumian mix net with randomized partial checking [9], 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 [11], 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 [9]: 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.

A. 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* $\mathsf{S}_1, \dots \mathsf{S}_n$, *$m$ mix servers* $\mathsf{M}_0, \dots, \mathsf{M}_{m-1}$, 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 [9]). However, as pointed out in [11], 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, $j \in \{0, \dots, m-1\}$, 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, $pk_0, \dots, pk_{2m-1}$, 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 $C_0 = \alpha^1_0, \dots, \alpha^n_0$ 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 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 $m_1, \dots, m_n$ (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 $l \le n$ the number of messages left in C'_{2j}.
2. *First mixing.* M_j uniformly at random chooses a permutation π_{2j} of $\{1, \dots, l\}$ 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 $\{1, \dots, l\}$ 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 $\pi_{2j}(1), \dots, \pi_{2j}(l)$ and commitments to the values $\pi^{-1}_{2j+1}(1), \dots, \pi^{-1}_{2j+1}(l)$ (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

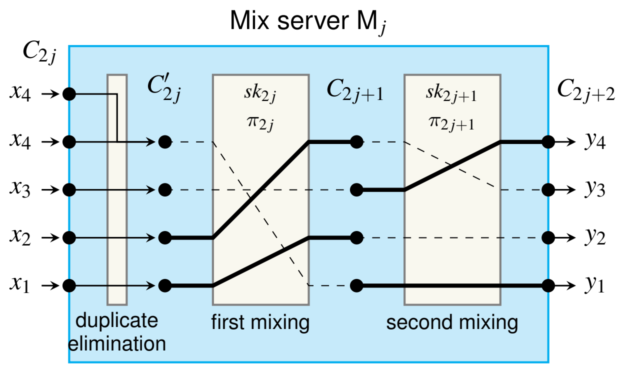

Figure 1. Mixing by M_j. Solid bold lines represent audited links and dashed lines represent not audited links.

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 $i \in \{1,\dots,l\}$ it is randomly decided, independently of other elements, whether i is added to $I_j \subseteq \{1,\dots,l\}$ 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 $i \in \{1,\dots,l\}$ 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 [11], the third step, which often has been omitted in implementations and is not mentioned in [9], 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.

B. 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 [13], [14], which we briefly recall before presenting the model of RPC mix nets and which in turn is based on the IITM model [16], [12].

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 $p_1,\dots,p_l$ 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 $a_1,\dots,a_n$. An *instance of* P is a process of the form $\pi = (\pi_{\mathsf{a}_1} \parallel \dots \parallel \pi_{\mathsf{a}_n})$ 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 $f(\ell) \le \frac{1}{\ell^c}$, 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 $f(\ell) \le \lambda + \frac{1}{\ell^c}$, for all $\ell > \ell_0$.

2) Chaumian RPC Mix Nets Modeled as Protocols: We model an RPC mix net as a protocol in the sense of Section II-B1. The set of agents of such a protocol is as introduced in Section II-A 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), which it provides to all parties. The CRS is used by agents for non-interactive zero-knowledge proofs of correct decryption (see also Section IV-B).

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 II-A.

Sender. The honest program $\hat{\pi}_{\mathsf{S}}$ of a sender S implements the procedure described in Section II-A: 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. 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 II-A. 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 II-A. 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).

III. 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 [13]. They should also apply to other forms of mix nets.

The (general) definition of accountability of a protocol from [13] 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" 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 [13] 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 II-B1).

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 II-B2, 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 IV.

Verifiability. Accountability and verifiability are tightly related as shown in [13]. 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 [13]. 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 $\Pr[\pi^{(\ell)} \mapsto J : \mathsf{accept}]$ is an overwhelming function in ℓ.

(ii) For every instance π of P, the probability $\Pr[\pi^{(\ell)} \mapsto \neg\gamma_k \wedge J : \mathsf{accept}]$ 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 [13] 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.

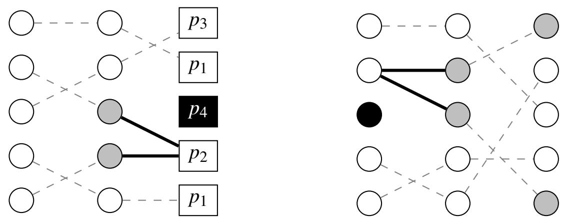

Figure 2. Examples of cheating by the last mix server (left-hand side) and by any mix server (right-hand side).

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.

IV. 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 IV-A, 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 IV-A.

A. 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, $\pi_{2m-1}^{-1}(p)$, 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 [11] 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.

B. 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 II-B2 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 [19]. (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 VI, 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 II-B, 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 NIZK property as well.

Now, the following theorem holds for Chaumian RPC mix nets as modeled in Section II-B 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 III, Theorem 1 also immediately implies that Chaumian RPC mix nets enjoy λ_k-verifiability with tolerance k.

In Appendix A, we provide a proof sketch for Theorem 1, with a detailed proof provided in the full version of this paper [15].

V. 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 [14]. 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 [10] 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 II-A, 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.[1] 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.[2]

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 V-C, we introduce and precisely define the concept of risk-avoiding adversaries.

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

As already mentioned in Section II-B, 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 II-B) 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 *$\mathsf{P}^j_{\mathsf{mix}}(n,m,\mu)$ with l honest senders achieves δ-privacy*, if

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

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

[1] 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.

[2] 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.

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'.

B. 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 [14]. 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 $\{p_1,\dots,p_k\}$ 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* $\vec{r} = (r_1,\dots,r_k)$ of the protocol, where r_i, $i \in \{1,\dots,k\}$, 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:

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

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 [14].

C. 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. Moreover, let $\alpha \in [0,1]$.

We say that π^* is *α-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 α-bounded as a function in the security parameter ℓ, for all valid input plaintexts p. We say that π^* is *completely risk-avoiding* if it is 0-risk-avoiding. Note that it is venturesome if it is 1-risk-avoiding.

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

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

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

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

VI. 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 the values α for α-risk-avoiding adversaries, ranging from completely risk-avoiding adversaries ($\alpha = 0$) to venturesome adversaries ($\alpha = 1$). 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 IV 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. For α-risk-avoiding adversaries, the level of privacy both in the case of in-phase and post-phase auditing is high even for quite big values of α (e.g., 75%), i.e., adversaries that are willing to take substantial risks.

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.

A. *Problems with Privacy*

As already illustrated in Section II-A, 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. As we know from our analysis of verifiability/accountability, an adversary who is willing to accept being caught with probability $1-(\frac{3}{4})^k$ could, for example, drop k entries originally sent by honest senders before they are processed by the honest mix server. In the extreme case, where k is the total number of honest senders (excluding the observed sender), all honest entries are dropped before the honest mix server is invoked and the adversary, knowing the plaintexts in the dishonest entries, can 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.

In fact, a particularly interesting case from an analysis point of view is when we consider in-phase auditing and not completely risk-avoiding adversaries, with venturesome adversaries being the most extreme case. 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.

B. *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.

C. *Privacy for Completely Risk-Avoiding Adversaries*

As a preparation for the analysis of privacy in the general case (for α-risk-avoiding adversaries with any value of α), we begin with the case of completely risk-avoiding adversaries ($\alpha = 0$). The results presented here hold true both for the case of in-phase and post-phase auditing.

By the results of Section IV, we know that whenever dishonest mix servers change some entry, this can be detected with non-negligible probability. Therefore, a completely 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 $\mathsf{P}^j_{\mathsf{mix}}(n,m,\mu)$ with l honest senders achieves $\delta_{l,\mu}$-privacy w.r.t. completely risk-avoiding adversaries, where*

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

Moreover, $\delta_{l,\mu}$ is optimal, i.e., this protocol does not achieve δ-privacy w.r.t. completely 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 completely 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.

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.

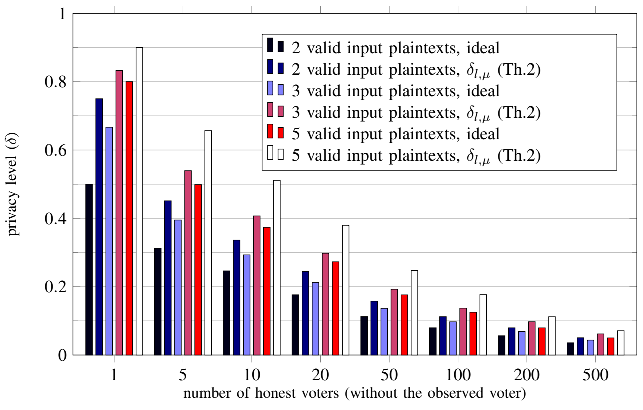

Figure 3. Level of privacy ($\delta_{l,\mu}$) w.r.t. completely risk-avoiding adversaries and in the ideal case $\delta^{id}_{l,\mu}$, uniform distribution of input plaintexts. These figures have been obtained by straightforward calculations using the δ-formulas as provided in the theorems.

Because we consider completely risk-avoiding adversaries, we know that no entry is dropped or changed by the mix server. 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 B for some more details, and [15] for the full proof). ∎

D. Privacy in the General Case

We now consider α-risk-avoiding adversaries, for all $\alpha \in [0,1]$, which includes, in particular, venturesome adversaries (for $\alpha = 1$), where 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 for venturesome adversaries, 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 completely risk-avoiding adversaries). If the first mix server is not honest, an α_k-risk-avoiding adversary, with $\alpha_k = 1 - \left(\frac{3}{4}\right)^k$, can, according to his most effective strategy as discussed in Section IV, drop k honest entries and, therefore, deliver only $l' = l - k$ honest entries to the honest mix server, assuming that there is a sufficient number of dishonest mix servers preceding the honest mix server. At this point, the situation is as in the case of completely risk-avoiding adversary with l' honest senders which yields the advantage $\delta_{l',\mu}$ (and this is the best an α_k-risk-avoiding adversary can achieve). Note that one dishonest mix server can drop $\lfloor \frac{l}{2} \rfloor$ out of l honest entries. Hence, it is easy to calculate how many dishonest mix servers are needed to drop k honest entries.

Formally, we obtain the following result.

Theorem 3. *For all $\alpha \in [0,1]$, the protocol $\mathsf{P}^1_{mix}(n,m,\mu)$ with post-phase audit and with l honest senders achieves $\delta_{l,\mu}$-privacy w.r.t. α-risk-avoiding adversaries, where $\delta_{l,\mu}$ is as in Theorem 2.*

For $j > 1$, the protocol $\mathsf{P}^j_{mix}(n,m,\mu)$ with post-phase audit and with l honest senders achieves $\delta_{l',\mu}$-privacy w.r.t. α_k-risk-avoiding adversaries, where $l' = \max(0, l-k)$ and $\alpha_k = 1 - \left(\frac{3}{4}\right)^k$. The protocol does not achieve δ-privacy w.r.t. α_k-risk-avoiding adversaries for any $\delta < \delta_{l',\mu}$, assuming that the number of mix servers preceding the honest mix server is sufficiently big, as discussed above.

Notice that, as a corollary of the above theorem we obtain that RPC mix nets do not achieve δ-privacy w.r.t. venturesome adversaries for any $\delta < 1$ (again assuming a sufficient number of mix servers preceding the honest mix server), which means that for venturesome adversaries privacy is completely broken. Indeed, 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, privacy is completely broken in this case. This case is quite obvious and has already been pointed out in the original proposal for RPC mix nets [9]. In [11], 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.)

The more relevant case from a practical point of view, compared to the case of venturesome adversaries ($\alpha = 1$), seems to be the case of α-risk-avoiding adversaries, for relatively small α (due to, as mentioned, possible penalties for cheating). In this case, Theorem 3 means that adversaries who are willing to accept only limited risk of being caught gain only very little in terms of breaking privacy. For example, considering the case with three valid input plaintexts (e.g., in an election with three candidates), an adversary who is willing to accept a risk of about 75% of being caught (which is already very high), can drop only 5 honest entries. For 50 honest senders, by this he gains only 0.01 in his ability to break privacy (that is, the probability of guessing correctly the choice the observed sender made increases only very slightly). If the number of honest senders is 200, then the adversary gains only 0.001, which is insignificant for all practical purposes. Therefore, while by cheating an adversary can obtain some advantage, for an adversary who is willing to take only limited (but still high) risk, this advantage is insignificant. Hence, in this realistic case the mitigation mentioned above, which would make RPC mix nets more complex, might not even be necessary.

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 that cheating provides for breaking privacy.

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

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

with $k' = \min(k,l)$ and $\alpha_k = 1 - \left(\frac{3}{4}\right)^k$.

The protocol does not achieve δ-privacy w.r.t. α_k-risk-avoiding adversaries for any $\delta < \delta^*_{l,k,\mu}$, assuming the number of mix servers preceding the honest mix server is sufficiently big, as discussed above Theorem 3.

The intuition behind the constant $\delta^*_{l,\mu}$ is the following. We consider those strategies of the adversary (indexed with l'), where he always drops the same number of honest entries, letting l' entries reach the honest server. If cheating is not detected, which happens with probability $\left(\frac{3}{4}\right)^{l-l'}$ ($l-l'$ is the number of dropped entries), then the advantage of the adversary is $\delta_{l',\mu}$. This is because this case strictly corresponds to the case with l' honest senders from Theorem 2. Otherwise, if the adversary is caught, he does not learn anything from the protocol run. We consider all possible l' as above, for which the risk of being caught does not exceed α_k (hence, the range $l-k',\dots,l$), and pick the one for which the advantage of the adversary is biggest. Clearly, in the proof, we demonstrate that the family of strategies considered above contains an optimal strategy which maximizes the advantage of the adversary (see Appendix C for some more details and [15] for the full proof of Theorem 4).

Note that the above theorem covers the case of venturesome adversaries.

We have computed the constants $\delta^*_{l,k,\mu}$ for those parameters we also considered for $\delta_{l,\mu}$ (Figure 3). For all those parameters, it has turned out that $\delta^*_{l,k,\mu} = \delta_{l,\mu}$ (for all k). That is, the optimal strategy of the adversary is to *not* remove/change any of the honest entries (and, therefore, it coincides with the optimal strategy of a completely risk-avoiding adversary). In particular, even if we consider a venturesome adversary, the risk of being caught, when manipulating honest entries, and in consequence not learning anything always outweighs the advantage these manipulations bring 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.

VII. 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 $1 - \left(\frac{3}{4}\right)^k$. Hence, already the manipulation of just one (two) message(s) is 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 (i.e., α-risk-avoiding adversaries with $\alpha = 1$), in which case there is no privacy. Otherwise, for less bold adversaries who nevertheless are willing to take big risks of being caught (e.g., $\alpha = 0.75$), the level of privacy is close to ideal, both for in-phase and post-phase auditing. Interestingly, in-phase auditing is only slightly better than post-phase auditing in this case. Altogether, since in the context of e-voting it is rather unlikely that an adversary is very venturesome, the level of privacy provided to single senders/voters is quite satisfying.

In summary, our results show that Chaumian RPC 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.

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] 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.

[10] 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.

[11] 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.

[12] Ralf 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 [16] for a full and revised version.

[13] 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.

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

[15] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Formal Analysis of Chaumian Mix Nets with Randomized Partial Checking. Technical report, University of Trier. Available at http://infsec.uni-trier.de/publications/paper/KuestersTruderungVogt-TR-RPCChaumian-2014.pdf, 2014.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

Appendix

A. Proof Sketch of Theorem 1

Here we provide a high-level intuition about the 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 completeness property.

The basic idea is to show that the strategy of the adversary that drops exactly $k+1$ honest entries as described in Section IV (*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.

Let us now consider how a dishonest mix server M_j may behave in the general case. First, we can notice that, if a mix server does not post messages in the expected format, it is trivially detected. This includes the duplicate elimination step, where every departure from the prescribed protocol is easily detectable.

First mixing. We now focus on the first mixing step performed by a mix server M_j. Let $comm_1, \dots comm_l$ be values posted by M_j as commitments to π_{2j}.

We say that an index i is *unsafe*, if either the mix server is not able to open $comm_i$ to a proper index (that is an integer in the range $\{1, \dots, l\}$) or, if it is able to open it to a proper index i', then it is not able to demonstrate that $C_{2j+1}[i]$ is a decryption of $C'_{2j}[i']$. Otherwise an index is said to be *safe*.

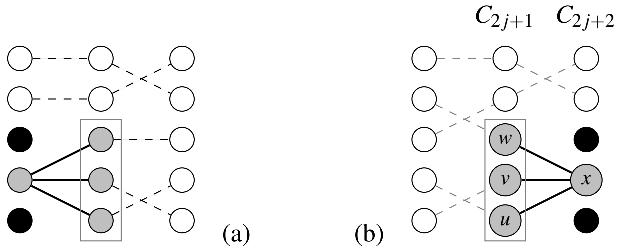

Figure 4. An example of a collision groups (nodes in rectangles) in the first and the second mixing.

Now, one can show that the strategy that the adversary uses is not optimal if there is some unsafe index. Indeed, one unsafe index allows the adversary to change exactly one entry, with the probability of detection $\frac{1}{2}$ (instead of $\frac{1}{4}$, when the optimal strategy is used). Therefore, we can only consider strategies that never produce unsafe indices.

We say that a (safe) index i points to i', if i' is the (only) value that the mix server is able to open $comm_i$ to (we know that the adversary is able to open $comm_i$ only to one value, because otherwise we could break the binding property of the commitments). A set Y of (safe) indices is called a *collision group* if $|Y| > 1$ and all indices in Y point to the same index i'. A collision group of size 3 is illustrated in Figure 4, (a), where all the gray elements in the middle column point to the same element in the left column.

Because of the assumption that the used ZK proofs are sound, one can show that if the adversary does not produce unsafe indices or collision groups, he cannot change the outcome of the mix net even by one entry. Therefore, to show that the defined above strategy is indeed optimal (not worse than others), as far as the first mixing step of any dishonest mix server is considered, we need to show that using collisions groups of size bigger than two is suboptimal (collision groups of size two are used in the optimal strategy). Indeed, this can be shown by elementary calculations.

Second mixing of M_j, for $j < 2m$. Now let us analyze the second mixing step excluding the last mix server. We define the notion of safe and unsafe indices and the one of collision groups as in the case of the first mixing. Analogously to the above case, it is easy to show that using unsafe indices is not optimal.

To conclude the proof in this case, one can show that it is completely useless for a corrupted mix server to produce output with collision groups, because it does not cause any honest entries to be changed/dropped. To see this, let us consider a collision group of size k (see Figure 4, (b) for an example). By correctness of the ZK proofs, x is the decryption of all the ciphertexts in the collision group. If one of the ciphertexts in this group (say w) is an honest entry (that is $w = \alpha^i_{2j+1}$), then, because we consider safe indices and because of the soundness of the decryption proofs, x

must have been produced by the same (honest) sender, that is $x = \alpha^i_{2j+2}$. The same reasoning would apply to another entry (say v), if it were honest. Therefore, due to our assumption that the probability for collisions of ciphertexts is negligible, it cannot be the case that v is an honest entry. Therefore, a collision group can contain at most one honest entry and this entry is not dropped.

Second mixing of the last mix server. As noted before, it is possible to change one honest entry by cheating at this step with the same probability of being detected ($\frac{1}{4}$) as in the case of the optimal strategy (*cheating by any mix server*). The last mix server can cheat *both* at its first and second mixing step at the same time. One can easily show, however, that this combination does not make the risk of being caught smaller.

Remark 1. As already noted in Section II-B, the probability distribution μ does not play any role in the proof of Theorem 1. Indeed, we could allow the adversary to provide unencrypted input for the honest senders and the result would still work.

B. Proof Sketch of Theorem 2

First we prove a result that is used both in the proof of this theorem 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 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). 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 (output 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.

Lemma 1. *The advantage of A (in distinguishing if the sender under observation has chosen p or p') is not bigger than the advantage of A^*.*

In the proof of this lemma, we use an idealized variant of the protocol, where honest senders use ideal encryption in one of the two steps corresponding to the honest mix server: an honest sender either encrypts a string of zeros (instead of α^i_{2j+2}) to obtain α^i_{2j+1} or she encrypts a string of zeros (instead of α^i_{2j+1}) to obtain α^i_{2j}. We show, by reducing this problem to IND-CCA2 security of the used encryption scheme, that such a system (under the assumption that the audit does not asks for revealing the ideal encryption steps), is indistinguishable from the original system. Here we also use the hiding property of the commitment scheme and the assumption that the produced proofs of correct decryption are zero-knowledge. Therefore, in the proof of Lemma 1, we can replace the original system with the idealized one, where the view of the adversary is completely independent of the choices made by honest senders, as long as they produce the same final multiset of plaintexts.

Now, the proof of Theorem 2 proceeds as follows. Since we consider a completely risk-avoiding adversary A, we know that all the entries of the honest senders and of the sender under observation make it to M_j. By Lemma 1, we can consider A^* instead of A (which is safe to do, as it can only make the advantage of the adversary bigger). It is easy to see that the computations carried out by A^* yield the constant from the theorem, as explained in Section VI-C. By this we can conclude that no completely risk-avoiding adversary can achieve higher advantage.

C. Proof Sketch of Theorem 4

By Lemma 1, we can only consider adversaries A such that $A = A^*$. We show that, without loss of generality (without worsening the advantage of the adversary in breaking privacy), we can consider only programs A that always try to drop the same number of entries (and deliver the same number l' of honest entries). We show this using, again, idealized protocol, as described in the proof of Lemma 1, and proving that, given some state of the protocol execution, the optimal choice of the adversary depends only on how many honest entries have been dropped so far.

Let us consider a strategy that always tries to drop the same number $l-l'$ of honest entries. As explained in the paragraph after Theorem 4, the advantage of the adversary using such a strategy is $\left(\frac{3}{4}\right)^{l-l'} \cdot \delta_{l',\mu}$.

Recall now that the adversary is assumed to be α_k-risk avoiding. By the above, the best strategy of the adversary is to uniformly try to deliver l' honest entries, where l' is picked in such a way that the risk of being caught does not exceed α_k (which is when $l' \in \{l-k', \dots, l\}$) and otherwise l' maximizes the advantage of the adversary. This yields (3).