Shaken, not Stirred - Automated Discovery of Subtle Attacks on Protocols using Mix-Nets

Jannik Dreier
Université de Lorraine, CNRS, Inria, LORIA

Pascal Lafourcade
Université de Clermont Auvergne, LIMOS

Dhekra Mahmoud
Université de Clermont Auvergne, LIMOS

Abstract

Mix-Nets are used to provide anonymity by passing a list of inputs through a collection of mix servers. Each server mixes the entries to create a new anonymized list, so that the correspondence between the output and the input is hidden. These Mix-Nets are used in numerous protocols in which the anonymity of participants is required, for example voting or electronic exam protocols. Some of these protocols have been proven secure using automated tools such as the cryptographic protocol verifier ProVerif, although they use the Mix-Net incorrectly. We propose a more detailed formal model of exponentiation and re-encryption Mix-Nets in the applied Π-Calculus, the language used by ProVerif, and show that using this model we can automatically discover attacks based on the incorrect use of the Mix-Net. In particular, we (re-)discover attacks on four cryptographic protocols using ProVerif: we show that an electronic exam protocol, two electronic voting protocols, and the "Crypto Santa" protocol do not satisfy the desired privacy properties. We then fix the vulnerable protocols by adding missing zero-knowledge proofs and analyze the resulting protocols using ProVerif. Again, in addition to the common abstract modeling of Zero Knowledge Proofs (ZKP), we also use a special model corresponding to weak (malleable) ZKPs. We show that in this case all attacks persist, and that we (re)discover these attacks automatically.

1 Introduction

The concept of Mix-Networks was introduced by David Chaum in 1981 [10] as a tool for achieving anonymity. The purpose of a Mix Network is to hide the correspondence between its input and output vectors. On a high level, a Mix-Net is run by a set of senders and a collection of mix servers, and works as follows. Each sender delivers its input to the first mix server. Each mix server privately shuffles its inputs list, and forwards the shuffled list to the next server. Unless all mix servers are corrupted, a Mix-Net should guarantee that the link between the senders and their associated messages in the output list remains secret to a third party.

Due to their importance in providing anonymity to the communicating parties, Mix-Nets play a significant role in building systems where privacy is a key security requirement, such as e-voting systems [14, 29] or e-exams [27]. In the context of voting, they are used to hide the link between voters and their votes to guarantee the voters' privacy, as typically the protocol publishes the list of the decrypted votes at the end to achieve verifiability. In such a protocol, any possibility to link a voter to his decrypted plaintext vote immediately breaks privacy. Voting protocols using Mix-Nets have been even employed in real political elections, for example in the Australian state of Victoria [18] or in Norway [28].

Since their introduction, many Mix-Net constructions have been proposed, analyzed and integrated into various cryptographic protocols. In [37], Park *et al.* presented the *Re-encryption Mix-Net*. These Mix-Nets rely on malleability properties of the encryption scheme, giving the possibility to re-encrypt a list of ciphertexts resulting in a new list of ciphertexts, without knowing or changing the associated plaintexts. In [30], Haenni *et al.* presented a different form of a Mix-Net, the so-called *Exponentiation Mix-Net*. These Mix-Nets create, from a list of ElGamal public keys [26], a new shuffled list of anonymized public keys, which can no longer be associated to individual parties, but can still be used to verify signed messages or encrypt data. Exponentiation Mix-Nets are used in various protocols, notably in electronic voting [30] and electronic exams [27].

The Remark! [27] exam protocol was analyzed and proven secure [23] using ProVerif [9], a tool for the formal analysis of cryptographic protocols in the symbolic model. In the symbolic model the algebraic properties of cryptographic primitives are usually modeled using equations, e.g., the equation $dec(enc(m,k),k) = m$ can be used to model a simple deterministic symmetric encryption scheme, where the functions dec and enc represent decryption and encryption, respectively. These equations can fail to take into account subtle behaviors of primitives which can lead to miss certain classes of attacks. For example, the model of Remark! in [23] abstracts away completely the implementation of the Mix-Nets. Hence, they missed an attack on the Mix-Net described in [40] that exploits the details of the exponentiation process. This attack was found manually and works as follows: an attacker submits a modified version of the public key of a targeted user as their own key to the Mix-Net. This enables the attacker to link both keys after the mixing, breaking the anonymity of the targeted user.

Similar attacks on Mix-Nets had already been described in 1994 [39], long before exponentiation mixnets [30] where proposed in 2011. Still, it took a decade for [40] to (apparently independently) discover that these Mix-Nets were insecure.

The goal of this paper is to propose new symbolic models that capture sufficiently well the mathematical properties of cryptographic primitives used in Mix-Nets and to automatically find such attacks using automated tools such as ProVerif, thus helping protocol designers avoiding these and similar attacks in the future.

Contributions. We show that with the advances of protocol verification tools it is now possible to model Mix-Nets more precisely. This allows us to automatically (re-)discover previously missed (but known) attacks, to discover new attacks, or to prove verify properties of protocols against a more powerful attacker that can exploit weaknesses of the Mix-Net. Our contributions are the following:

- We propose a detailed model of Haenni's [30] exponentiation Mix-Net, taking into account details of the exponentiations.
- Our refined model includes a new, more precise model of ElGamal encryption and signatures, which includes the exponentiation operations and is of independent interest as it can be used to model more precisely protocols that use ElGamal encryption or signatures, but no exponentiation Mix-Nets. To the best of our knowledge, until now symbolic models of ElGamal encryption completely abstracted away the exponentiations, hence potentially missing attacks.
- We then show that this new model of the exponentiation Mix-Net and ElGamal can be used to automatically analyze protocols using this Mix-Net. For this, we model and analyze three protocols in ProVerif: an e-voting protocol [30], an e-exam protocol [27], and the Crypto Santa protocol [42].
  - We give a formal model and analysis of the e-voting protocol from [30]. Our analysis shows that it is vulnerable to a privacy attack (similar to the one from [40]) due to a weakness in the Mix-Net, and the lack of Zero-Knowledge Proofs by the voters.
  - We are able to automatically find the attack from [40] on the exam protocol. Interestingly, Amin et al. [40], despite identifying this attack manually, used a simple formal model in ProVerif which was incapable of detecting the vulnerability.
  - We provide the first formal model and analysis of the Crypto Santa protocol, which turns out to be not vulnerable due to use of appropriate ZKPs by the participants.
- To fix the identified attacks we add the lacking ZKPs to the voting and exam protocols. We check the protocols again, using the standard symbolic modeling of ZKPs, and show that the corrected protocols are secure.
- The standard symbolic modeling corresponds to ideal ZKPs. We propose a model for weaker Zero-Knowledge Proofs (which are common [19], although they are known to be vulnerable [7]), and can show that they are insufficient in this case, as all attacks persist. To the best of our knowledge, these concrete attacks have not been described in the literature. Even the Crypto Santa protocol becomes vulnerable when using these weak proofs, yet the protocol does not specify which ZKPs should be used. Again, the model of weak ZKPs is of independent interest, as it can be used for any protocol using ZKPs of ElGamal keys, even if there is no Mix-Net.
- We also propose a refined model for re-encryption Mix-Nets, and analyze the Estonian e-Vote protocol [1]. As the protocol does not use any ZKP by the voters, it is vulnerable to attacks on the Mix-Net, which we can automatically detect (similar attacks were found manually before, e.g., [36]). We verify the protocol with added weak or strong ZKPs, and are able to show that only the strong ZKPs are sufficient to avoid attacks.

Related work. There exist several automatic protocol verification tools handling equivalence properties, but we chose ProVerif as Deepsec [13] cannot handle our equational theory, and although Tamarin [6] supports Diffie-Hellman exponentiation, the built-in exponentiation operator cannot appear inside a user defined equation. In the case of exponentiation Mix-Nets we need to define encryption and signature schemes using the Diffie-Hellman keys, hence we need to re-use the exponentiation operator in the equations modeling encryption and signature. Previous work mostly focused on improving the modeling of exponentiation in isolation [17], but we need to connect exponentiation and encryption.

Computational tools such as EasyCrypt [5] or CryptoVerif [8] would be able to detect such attacks, they however provide a significantly lower level of automation. Instead, this paper follows a line of work which tries to refine the modeling of cryptographic primitives in symbolic models and thereby to reduce the gap to computational models, while keeping a high level of automation. Similar work has been done for other primitives, for example for digital signatures [32], Diffie-Hellman exponentiation in weak groups [17], hash functions [11] and authenticated encryption with additional data (AEAD) [16]. In all these papers, the authors proposed more detailed and more realistic models of cryptographic primitives in the symbolic framework, to be able to identify protocol flaws exploiting weaknesses of the cryptographic primitives.

In [32], the authors refine the existing Tamarin models of signature schemes so they discover attacks exploiting for example the fact that in some schemes a signature potentially verifies against multiple different keys. In [17], the authors propose a novel extension of the symbolic model of Diffie-Hellman groups for Tamarin enabling them to capture a large family of attacks exploiting weak groups that were previously outside the symbolic model. In [11], they propose a methodology to systematically discover attacks using Tamarin and ProVerif that exploit weaknesses in widely deployed hash functions, such as length extension attacks. They automatically find known attacks, but also new variants of these attack on several protocols. In [16], the authors provide an automated analysis method in Tamarin for protocols that use AEAD. This method allows them to systematically find attacks that exploit the subtleties of the specific type of AEAD used, as different schemes have different weaknesses.

We propose a refined model for Mix-Nets in ProVerif. Our new models allow us to automatically re-discover known attacks on several protocols, but also to discover new attacks that could not be found using previous models.

Note that in symbolic models, used by the above works, all negligible probabilities are abstracted away. For example, if there is only a negligible probability to generate twice the same random value, or for the adversary to break an encryption scheme, in a symbolic model this will simply be impossible. It has been shown that under certain assumptions this can nevertheless give computational guarantees (an approach called "computational soundness", see e.g. [4, 34]). There are also works on symbolic models that allow non-negligible probabilities in the control flow, i.e., where agents can chose among different actions with given probabilities (e.g., [12]), however there is currently no tool support for this. As the goal of our work is to automatically identify attacks, we use standard symbolic models, where mature tools exist.

The first formal model of Mix-Nets was provided by Wolff *et al.* [46]. They formalized the Mix-Net and its components using the CSP process algebra and the FDR model-checker. The formal analysis was conducted considering a passive attacker, and the Mix-Net model was not specific to a particular type of Mix-Net, but rather general. At this time, this tool was only able to deal with simple dedicated equational theory.

In 2014, Stathakidis *et al.* [43] proposed a formal model for re-encryption Mix-Net suitable for automation based on CSP process algebra and the FDR model checker. They model and analyze the protocol in the presence of an intruder based on Roscoe and Goldsmith's perfect Spy [41]. The main security property analyzed was the robustness of the mix protocol, which is a functional property. Moreover, they do not model any mathematical properties of the Mix-Net.

Also in 2014, Küsters *et al.* [33] provided the first computational formal security analysis of Chaumian Mix-Net with random partial checking. In their paper, they focus on accountability, where misbehavior should be detectable and the responsible of the misbehavior should be blamed. Their analysis in the computational model is not automated. There are other works on the verification of Mix-Nets, e.g., in [31], where the authors provide a mostly manual proof of the Verificatum Mix-Net, but to the best of our knowledge there is none targeting exponentiation Mix-Nets.

Our work in the symbolic model using ProVerif allows us to automatically discover flaws or verify protocols using Mix-Nets. Compared to previous work in the symbolic model, we capture more precisely the exponentiation used in the Mix-Nets to achieve anonymity of the exchanged messages.

Outline. We start by recalling the symbolic models and the applied Π-Calculus in Section 2. In Section 3, we describe exponentiation Mix-Nets and detail known attacks against those types of Mix-Nets in Section 3.1. In Sections 3.2, 3.3, 3.4, and 3.5, we present our new formal models for exponentiation Mix-Nets, ElGamal encryption and ZKPs, respectively. In Section 3.6, we apply those models to three different protocols using exponentiation Mix-Nets to guarantee the privacy of the participants. In Section 4, we describe re-encryption Mix-Nets and detail known attacks against those types of Mix-Nets. In Section 4.2 we present our new formal model for re-encryption Mix-Nets. In Section 4.3, we apply those models to the Estonian e-voting protocol. We conclude and discuss future work in Section 5.

2 Preliminaries

We provide a formal analysis of re-encryption and exponentiation Mix-Nets in the symbolic model, also called Dolev-Yao model [22]. In this model, the cryptographic primitives are represented by function symbols and considered as black-boxes; the messages are terms built using these primitives, which may be read, modified, deleted or injected by the adversary who has total control over the network. Under the assumption of perfect cryptography, the attacker is only able to perform cryptographic operations when it is in possession of the right keys. We model Mix-Net protocols in the applied Π-Calculus [3] and check privacy properties in ProVerif [9].

The applied Π-Calculus is a language for modeling security protocols. We briefly recall its syntax, semantics and equivalence relation. For the full details, see [3].

Syntax. A finite *signature* Σ is a finite set of function symbols each with an associated arity. Given a signature Σ, an infinite set of names $\mathcal{N}$ and an infinite set of variables $\mathcal{V}$, the set of *terms* $\mathcal{T}$ is defined as names, variables and function symbols applied to other terms. Terms are equipped with an equational theory (a set of equations), which induces an equivalence relation between terms, denoted $=_E$.

The behavior of a probabilistic asymmetric encryption and

$P,Q,R ::=$	
$\mathbf{0}$	null process
$P \mid Q$	parallel composition
$!P$	replication
$\nu n \cdot P$	name restriction
if $M = N$ then P else Q	conditional
$in(c,x) \cdot P$	message input
$out(c,N)$	message output

Table 1: Grammar of Processes in the applied Π-Calculus.

$A,B,C ::=$	
P	plain process
$A \mid B$	parallel composition
$!A$	replication
$\nu n \cdot A$	name restriction
$\nu x \cdot A$	variable restriction
$\{^M/_x\}$	active substitution

Table 2: Grammar of extended processes.

decryption, for example, is usually modeled by:

$$dec(penc(m, pk(x), r), x) =_E m \qquad (1)$$

where pk is a unary function symbol representing the public key corresponding to the secret key x, m is the plaintext and r represents a random variable.

Systems are described as *processes*. The grammar for processes is depicted in Table 1, where M and N are terms, n is a name, x is a variable and c is a channel name. The null process $\mathbf{0}$ does nothing; $P \mid Q$ is the parallel composition of P and Q; the replication $!P$ behaves as an infinite number of copies of P running in parallel. The process $\nu n \cdot P$ makes a new private name n then it runs P. Finally, $c(x)$ and $\overline{c}\langle N \rangle$ stand for an input and an output on the channel c respectively.

Processes are also extended with *active substitutions*. We write $\{^M/_x\}$ for the substitution that replaces the variable x with the term M. The grammar of extended processes is given in Table 2. $\{^M/_x\}$ is considered as a process and $\nu x \cdot (\{^M/_x\}|P)$ corresponds to let $x = M$ in P.

A *frame* is defined as an extended process built from 0 and active substitutions by parallel composition and restrictions. The *domain* of a frame is the set of variables for which the frame defines a substitution, and which are not under restriction. An extended process A is *closed* when its free variables are all defined by an active substitution.

A *context* $\mathcal{C}[\cdot]$ is defined as a process with a hole, that can be filled with any process. An evaluation context is a context whose hole is not under a replication, a condition, an input or an output. A context $\mathcal{C}[\cdot]$ closes A when $\mathcal{C}[A]$ is closed.

Semantics and Equivalences.

Internal reduction is the smallest relation on extended processes following the rules in Table 3.

COMM	$\overline{c}\langle x \rangle \cdot P \mid c(x) \cdot Q \rightarrow P \mid Q$
THEN	$if\ N =_E N\ then\ P\ else\ Q \rightarrow P$
ELSE	$if\ M =_E N\ then\ P\ else\ Q \rightarrow P$
	for ground terms M, N where $M \neq_E N$

Table 3: Internal reduction in the applied Π-Calculus.

The applied Π-Calculus defines *observational equivalence* to model the indistinguishability of two processes by the adversary. We write $A \Downarrow a$ when A can send a message on channel a.

An *observational bisimulation* is a symmetric relation $\mathcal{R}$ between closed extended processes with the same domain such that $A\ \mathcal{R}\ B$ implies:

1. if $A \Downarrow a$, then $B \Downarrow a$;
2. if $A \rightarrow^* A'$ and A' is closed, then $B \rightarrow^* B'$ and $A'\ \mathcal{R}\ B'$ for some B';
3. $E[A]\ \mathcal{R}\ E[B]$ for all closing evaluation contexts $E[\ ]$.

Observational equivalence ($\approx$) is the largest such relation.

3 Exponentiation Mix-Nets

In [30], Haenni *et al.* have presented the so-called *Exponentiation Mix-Nets*, which create, from a list of ElGamal public keys, a new shuffled list of anonymized public keys, which can no longer be associated to individual parties.

Let us consider the usual ElGamal setup: a group G of large prime order q with generator g in which the discrete logarithm problem is hard. We have a list of n public keys $\langle pk_i \rangle = \langle pk_1, \ldots, pk_n \rangle$ such that $pk_i = g^{sk_i}$ and m mix servers M_j. The first mix server M_1 takes the original list of public keys $\langle pk_i \rangle$, generates a fresh random $r_1 \in \{0, \ldots, q-1\}$ and computes $\langle pk_i^{r_1} \rangle$. Then M_1 permutes the resulting terms with a secret shuffled order $\langle pk_{\pi_1(i)}^{r_1} \rangle$ where π_1 is the permutation of indexes applied by M_1. Then M_1 sends $\langle pk_{\pi_1(i)}^{r_1} \rangle$ to the next mix server along with g^{r_1}. The following mix servers repeat these same steps as required. The last mix server M_m outputs the list: $\langle g^{sk_{\pi(1)} \cdot r}, g^{sk_{\pi(2)} \cdot r}, \ldots, g^{sk_{\pi(n)} \cdot r} \rangle$ along with g^r where: $\pi = \prod_{i=1}^{m} \pi_i$ and $r = \prod_{i=1}^{m} r_i$. Each party in possession of their secret key sk_i should be the only one able to identify her pseudonym from the final list published by the last mix server by computing $(g^r)^{sk_i} = pk_i^r$.

3.1 Attacks against Exponentiation Mix-Nets

Although exponentiation Mix-Nets were designed by Haenni *et al.* [30] to preserve anonymity and unlinkability of voters in their e-Voting Protocol by attributing anonymous keys, there is an attack. An attacker $\mathcal{A}$ has access to the list of public keys

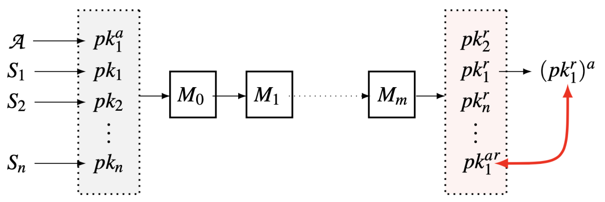

Figure 1: Attack against Exponentiation Mix-Nets.

$\langle pk_i \rangle$ of all participants. They want to learn the pseudonym generated by the mix servers for a specific public key pk_k. $\mathcal{A}$ can generate $a \in \{1,\ldots,q-1\}$, raise pk_k to the power a and append term $(pk_k)^a$ to the list of public keys as their own public key to be mixed. When the final list of pseudonyms is published, they raise each pseudonym to the power of a and look for a match. The attack, as illustrated in Figure 1 for a list of n public keys, exploits the commutative property of exponentiation: $((g^{sk_i})^a)^r = ((g^{sk_i})^r)^a$. This attack (a more specific variant of it) was described by Rakeei *et al.* [40], but it can be seen as a variant of an attack firstly described by Pfitzmann [39] in 1994.

3.2 Models of Exponentiation Mix-Nets

We give a formal specification of exponentiation Mix-Nets.

Definition 3.1 (Mix-Net) *A* Mix-Net *is a tuple* $(S, M, A, \tilde{n}_p)$ *where S is the process executed by the senders, M is the process executed by the Mix-Net servers, A is a process executed by an authority and $\tilde{n}_p$ is a set of private channel names, where A can be the null process whenever the protocol does not involve such an entity.*

Definition 3.2 (Exp-mixnet instance) *An* exp-mixnet instance *is a closed process* $EP = \nu\tilde{n} \cdot (S\sigma_{id_1}\sigma_{skS_1} | \ldots | S\sigma_{id_j}\sigma_{skS_j} \; |M\sigma_{m_1}| \ldots \; |M\sigma_{m_k})$ *where $\tilde{n}$ is the set of all restricted names, which includes the set of the protocol's private channels;* $S\sigma_{id_i}\sigma_{skS_i}$ *are the processes run by the senders with the substitutions specifying the identity and the secret key of the i^{th} sender respectively;* $M\sigma_{m_j}$ *is the process executed by the Mix-Net servers with the substitution specifying in particular their secret exponent.*

We write $MP_{\{id_1,id_2\}}[\cdot]$ for the Mix-Net instance MP without the processes for the senders id_1 and id_2.This allows us to define anonymity.

Definition 3.3 (Anonymous Shuffling) *A Mix-Net ensures* Anonymous Shuffling *if for any mixnet processes MP, any senders id_1 and id_2 and any private materials p_1 and p_2 (i.e., their keys or messages):*

$$MP_{\{id_1,id_2\}}[S\sigma_{id_1}\sigma_{p_1} \,|\, S\sigma_{id_2}\sigma_{p_2}] \approx MP_{\{id_1,id_2\}}[S\sigma_{id_1}\sigma_{p_2} \,|\, S\sigma_{id_2}\sigma_{p_1}]$$

In the case of exponentiation Mix-Nets, this definition requires that the process where id_1 has sk_1 and id_2 has sk_2 as their secret key, is equivalent to the process where id_2 has sk_1 and id_1 has sk_2. This prevents the attacker from obtaining information about the identity of the sender based on the outcome of the Mix-Net.

As stated above, we consider a Dolev-Yao attacker. They has complete control of the network, except the private channels: they can eavesdrop, remove, substitute, duplicate and delay messages that parties are sending to one another, and insert messages of their choice on the public channels. Depending on the properties, we also allow the attacker to corrupt parties. Such corrupted parties cooperate with the attacker by revealing their secret data and/or taking instructions from them. We model them using the definition given in [20]: if the process P is an honest party, then the process P^{c_1,c_2} is its corrupted version. The latter is a variant of the honest process sharing with the attacker channels c_1 and c_2. P^{c_1,c_2} sends all its inputs and freshly generated names to the attacker via channel c_1, and receives (via channel c_2) messages from the attacker that will determine its behavior.

Naturally, it is interesting to consider corrupted or dishonest senders, as they might be interested in obtaining a link between the outcome of a Mix-Net and the identity of some other users. We can do this by replacing honest senders with corrupted ones. For example, if we assume that candidate id_3 is dishonest, we obtain:

$$MP_{\{id_1,id_2,id_3\}}[S\sigma_{id_1}\sigma_{p_1} \,|\, S\sigma_{id_2}\sigma_{p_2} \,|\, (S\sigma_{id_3}\sigma_{p_3})^{c_1,c_2}]$$
$$\approx MP_{\{id_1,id_2,p_3\}}[S\sigma_{id_1}\sigma_{p_2} \,|\, S\sigma_{id_2}\sigma_{p_1} \,|\, (S\sigma_{id_3}\sigma_{p_3})^{c_1,c_2}]$$

3.3 Model and analysis in ProVerif

Exponentiation Mix-Nets have been modeled previously, but only as part of other, bigger protocols, for example in the analysis of the E-exam protocol Remark! [23]. However, as the Mix-Net was only part of a bigger protocol, this model used a high-level abstraction of the Mix-Net's functionality, which completely abstracts the details of the computations. This explains why some attacks were missed. The following functions were used to model the Mix-Net's computations:

- $pk(sk)$ computes the public key associated to the private key sk,
- $pseudo\_pub(pk(sk), rce)$ is the function used by the Mix-Net servers to compute the public pseudonyms of the participants based on their public keys $pk(sk)$ and the random exponent rce,
- $pseudo\_priv(sk, exp(rce))$ is the function used by the participants to compute their pseudonym from their private key sk and the new generator $exp(rce)$ (representing g^{rce}) generated by the Mix-Net.

The equational theory also involves a function *checkpseudo* which is a binary function taking as argument two pseudonyms and returns *true* when the first argument, representing the pseudonym computed by the Mix-Net, matches the second argument which represents the pseudonym recomputed by the candidate based on the new basis: $checkpseudo(pseudo\_pub(pk(sk), rce), pseudo\_priv(sk, exp(rce))) = true$.

This modeling is sufficient to represent the Mix-Net functionality, but it is clearly insufficient to capture the attack described above, as all exponentiations are hidden away by the functions *pseudo_pub* and *pseudo_priv*.

However, recent progress in ProVerif's development enables us to use a much more precise modeling, which we describe in the following.

Equational theory. Our new function consists of a single binary function *exp* which denotes the operation of exponentiation performed by the participants of the protocol. The properties of this primitive are captured by:

$$\begin{aligned} exp(exp(g,x),y) &= exp(exp(g,y),x) \\ exp(exp(exp(g,x),y),z) &= exp(exp(exp(g,x),z),y) \end{aligned} \tag{2}$$

This equational theory directly models exponentiation, enabling us to capture the attack when checking the protocol against Anonymous Shuffling. Note that in the equations g is a fixed generator and not a variable, as otherwise the pre-treatment of equations by ProVerif does not terminate. The first equation takes into account the equation needed for the protocol to work but is insufficient to capture attacks against exponentiation Mix-Nets. This is the reason why we need to explicitly give an equation for three exponents, as the attack uses an additional exponentiation. Adding additional exponentiation (more than three) to the equational theory caused non-termination of ProVerif in our examples.

Model. The process for each Mix-Net server M is given in Figure 2, the honest sender process is: let S(sk$_S$) = out(ch, (exp(g, sk$_S$))). The sender simply outputs their public key. Each Mix-Net server inputs the list of all keys, checks that they are distinct, applies their random exponent to all keys and outputs the list of all keys in random order (modeled using parallel outputs on the same channel). We note that all keys should be different. Otherwise a trivial attack consists in copying the public key to track: the corresponding pseudonym is the one figuring twice in the list output by the Mix-Net.

In ProVerif, we add dedicated private channels between the two participants swapping their keys and the Mix-Net to ensure that both keys take part in the mixing. The keys are still published, and the attacker can still add a malicious key to the mixing.

Analysis. The result of the analysis of exponentiation Mix-Nets is given in Table 4. ProVerif concludes that *Anonymous Shuffling* property is not satisfied and finds the exact same attack depicted in Figure 1.

```
let $M(e_{N_j}) =$
1: $in(ch, pk_{S_1})$.
2: $\vdots$
3: $in(ch, pk_{S_k})$.
4: if $(pk_{S_1} <> pk_{S_2})\&\&\ldots\&\&(pk_{S_1} <> pk_{S_k})$ then
5: $\vdots$
6: if $(pk_{S_k} <> pk_{S_1})\&\&\ldots\&\&(pk_{S_k} <> pk_{S_{k-1}})$ then
7: ( $out(ch, exp(pk_{S_1}, e_{N_j}))$ ||
8:   $\vdots$
9:   $out(ch, exp(pk_{S_k}, e_{N_j}))$ ).
```

Figure 2: Mixnet process.

3.4 Refined Model of ElGamal

ElGamal asymmetric encryption is typically modeled the same way as any other public key encryption schemes: using a model for an abstract standard probabilistic asymmetric encryption scheme, as in Equation (1) (e.g., in [15, 24]).

However, when using our equational theory depicted in Equations (2) to model the Mix-Net operations, the public keys are of the form $exp(\cdot,\cdot)$ and not $pk(\cdot)$, so we need to modify the equational theory used to model ElGamal encryption, and also the equational theory used to model signatures.

An intuitive approach to modeling ElGamal encryption, where public keys are the result of the exponentiation operator, would be as follows: $dec(enc(m, exp(g, sk), r), sk) = m$ where g is a fixed generator. However, this equation only allows encryption under public keys with the form $exp(g, sk)$. The attack described in Section 3.6.3 would not be detectable using this equation as the attacker's key involves two exponentiations.

Another attempt at modeling ElGamal would be to replace the fixed generator g in the aforementioned equation with a variable to allow encryption with different generators: $dec(enc(m, exp(X, sk), r), sk) = m$. This equation is incorrect. Consider a dishonest party constructing their public key $D = exp(H, d)$ from an honest key $H = exp(g, h)$. When the dishonest party receive a message m encrypted with their public key $enc(m, exp(H, d), r)$, they should not be able to obtain m as when deciphering one should obtain $m \cdot ((g^h)^d)^r \cdot ((g^r)^d)^{-1} \neq m$. However, with the latter equation we have $dec(enc(m, exp(H, d), r), d) = m$. To avoid this issue, we include the generator used in the encryption in the equation, which is consistent with the real cryptographic primitive (the public key includes the generator as a parameter).

To that end, we use the following equations:

$$\begin{aligned} \text{dec(enc(m, X, exp(X, s), r), X, s)} &= \text{m} \\ \text{getmess(sign(m, X, s))} &= \text{m} \\ \text{checksign(sign(m, X, s), X, exp(X, s))} &= \text{m} \end{aligned} \tag{3}$$

The first equation models ElGamal probabilistic public key encryption and decryption. To encrypt a message m one needs to generate a random r and to use the public parameters of the receiver: a basis X and the key $exp(X,s)$ such that s is the secret key of the receiver. For example, with respect to the notation given in the previous section, let $pk_S = exp(g, sk_S)$ be the public key of a sender and $exp(pk_S, e_N)$ be the pseudonym created by the Mix-Net. To encrypt a message m into the ciphertext c, we use the constructor enc with the following parameters m, $exp(g, e_N)$, $exp(pk_S, e_N)$ and r, respectively. This works because of the commutativity of the function exp defined in the first equation of the equational theory given in Equations (2). Thus, the only way to decrypt c is to apply the destructor dec using the secret key sk_S as parameter and specifying the basis $X = exp(g, e_N)$ used for the encryption.

The second and the third equations model a digital signature algorithm. To sign a message m, one needs to specify the basis X used in the public parameter $exp(X,s)$ and the secret exponent s. Then, $sign(m,X,s)$ is a digital signature for the message m. The second equation states that anyone can extract a message from a signature (we do not assume the signature scheme to be hiding). The third equation models the verification of a signature using the corresponding public key $exp(X,s)$ and the basis X with respect to the parameters used by the signer. This model of ElGamal probabilistic public key encryption is not bound to any protocol, and more precise than the usual models, as it inherits the algebraic properties of the exponentiation operation defined in the equational theory given in Equations (2). Obviously, it can also be used in protocols that do not include exponentiation Mix-Nets. Our model of the ElGamal encryption has allowed us to find a subtle attack described later in Section 3.6.3.

3.5 Refined Model of Zero Knowledge Proofs

A *Zero Knowledge Proof* (ZKP) allows a party to show to another party that a mathematical statement is true without revealing anything other than the truth of the statement itself. A specific class of ZKPs are proofs of knowledge, in which the prover demonstrates knowledge of the preimage $x \in X$ of a public value $y = \phi(x) \in Y$, where ϕ is supposed to be a one way function. These proofs are in particular used to prove knowledge of the discrete logarithm $y = g^x$ in a multiplicative finite group G_q with a generator g of order q. Moreover, interactive proofs between provers and verifiers can be turned into non-interactive ones using the Fiat-Shamir heuristic [25].

We enriched our model with a non-interactive knowledge proof (NIZKP) of the discrete logarithm. The sender is required to transmit its message along with a ZKP proving its possession of either the secret key sk_S for the exponentiation Mix-Nets or the randomness used for encryption for the re-encryption Mix-Net. Upon a valid verification of the proof given by the sender, the Mix-Net servers accept the entry. Otherwise, they reject the message.

Intuitively, adding a ZKP of the possession of the secret information along with the messages fixes the attack mentioned above. However, two variants of the Fiat-Shamir transformation appear in the literature. In [7], Bernhard *et al.* distinguish a weaker and a stronger variant. Both variants begin with the prover making a commitment. The stronger variant hashes both the commitment and the statement to be proved, while the weak variant hashes only the commitment.

The latter version is subject to an attack which allows a malicious party to “fake” a proof as follows [7]. To create a proof, an honest prover, having as public key $pk = g^{sk}$, picks a random $a \in \{1, \dots, q-1\}$, computes $A = g^a$ and then, they hash A to create a challenge $c = \mathcal{H}(A)$. Finally, they compute $f = a + c \cdot sk$. The proof corresponds to the pair (c, f) and the verification procedure consists in checking whether c is equal to $\mathcal{H}(g^f \cdot (pk)^{-c})$ or not. For an honestly generated proof the verification procedure succeeds since we have $\mathcal{H}(g^f \cdot pk^{-c}) = \mathcal{H}(g^{a+c\cdot sk} \cdot g^{-sk\cdot c}) = \mathcal{H}(g^a) = c$.

However, a dishonest prover can fake a NIZKP as follows. They pick A' a random element from G_q, f' a random exponent, computes $c' = \mathcal{H}(A')$ and the verification of the proof (c', f') succeeds for $pk' = (g^{f'} \cdot A'^{-1})^{c'^{-1}}$ as $\mathcal{H}(g^{f'} \cdot pk'^{-c'}) = \mathcal{H}(g^{f'} \cdot ((g^{f'} \cdot A'^{-1})^{c'^{-1}})^{-c'}) = \mathcal{H}(A') = c'$. The public key depends on the faked proof, *i.e.*, the dishonest prover must compute the proof before choosing pk'.

Because of this attack, the use of the weak variant may invalidate the proof of knowledge. The authors of [7] stated that the weak Fiat-Shamir transformation can safely be used when the statement (in this example, the public key) is fixed first (as in the attack the public key depends on the proof). However, since exponentiation Mix-Net or re-encryption Mix-Net are carried on as part of other bigger protocols, the list of inputs is not a priori known. It is therefore interesting to investigate the impact of the weak variant in our context. Note also that the weak Fiat-Shamir transformation is widely used in practice when it comes to non-interactive proof systems: in [19] the authors examined over 75 different open-source implementations of proof systems that use the Fiat-Shamir heuristic and found 36 systems using the weak variant.

In the case of exponentiation Mix-Nets, the use of weak ZKPs allows for the following attack. Let $(c, f) = (\mathcal{H}(g^a), a + c \cdot sk)$ be the proof generated by an honest participant with public key $pk = g^{sk}$. An attacker computes $c' = \mathcal{H}(A^{c^{-1}})$, where $A = g^a$ is the commitment of the honest participant, chooses $pk' = pk^{c'^{-1}}$ as public key and computes $f' = c^{-1} \cdot f$. The attacker sends (c', f') as a proof along with their public key pk' to the Mix-Net. The verification procedure of the ZKP by the Mix-Net succeeds since we have $\mathcal{H}(g^{f'} \cdot (pk'^{c'})^{-1}) = \mathcal{H}(g^{c^{-1} \cdot a + sk} \cdot g^{-sk}) = c'$.

As the verification of the proof succeeds, the attacker can then perform the same attack as described in Figure 1 by raising pseudonyms to the power of c'^{-1} (instead of a).

Models and analysis. We use two different models, one

Protocol	ZKP	Result	Time
Exponentiation Mix-Nets	without	✗	2 s
	weak	✗	1 m 6 s
	strong	✓	3 s
Re-encryption Mix-Nets	without	✗	1 s
	weak	✗	2 s
	strong	✓	1 s

Table 4: Results of our analysis of Anonymous Shuffling, with and without the added ZKP.

modeling weak ZKPs and the other modeling strong ZKPs.

$$ck(szkp(A,g,x),g,exp(g,x),h(g,exp(g,x),A)) = true \quad (4)$$
$$ck(wzkp(A,X,x),X,exp(X,x),h(A)) = true \quad (5)$$

These equations state that the verification of a proof of knowledge of a secret key succeeds only if the proof was created genuinely: it needs to be checked using the same generator, public key and hash that was used to generate it. The first argument of both functions (*szkp* and *wzkp*) refers to the commitment used in the proof. The hash of the commitment is used by *ck* to check the adequacy of the challenge, and in the case of the strong variant the hash includes the public key. Moreover, in case of the strong ZKP, the proof is only valid when using the public generator g, as this is part of the statement. In the case of a weak ZKP, the hash only includes the commitment, and the generator is now a variable as it is no longer part of the statement. The latter allows the attacker to perform the attack described above, where the public key is $exp(exp(g,x),a)$ for some x and a. Note that in our ProVerif code, when the Mix-Net verifies a weak proof, we input the value X from the network – fixing g would prevent the attacker from using a public of the form $exp(exp(g,x),a)$, as in the first equation.

With the strong variant ProVerif concludes that Anonymous Shuffling is satisfied for exponentiation Mix-Nets (Table 4). When using the weak ZKP, ProVerif returns attack traces on the exponentiation Mix-Net. The ProVerif attack works in the same way as the one without a ZKP (Figure 3.1): an attacker builds a new key $pk' = pk^{\alpha}$ from their victim's key pk and bypasses the proof check due to Equation (5), since $ck(wzkp(A,pk,\alpha),pk,exp(pk,\alpha),h(A)) = true$. In reality, this attack does not work for any α, but for example choosing $\alpha = c'^{-1}$ results in a real attack as described above. Using this approach, we were able to instantiate all attack traces found by the tool with real attacks. In fact, ProVerif cannot find the real attack directly as our equational theory does not encompass inverses (and adding them is currently not possible in ProVerif). However, our equations do not induce false vulnerabilities in the modeled cryptographic primitives since they represent a real (albeit abstracted) behavior of the primitive, and the missing values are simple to instantiate.

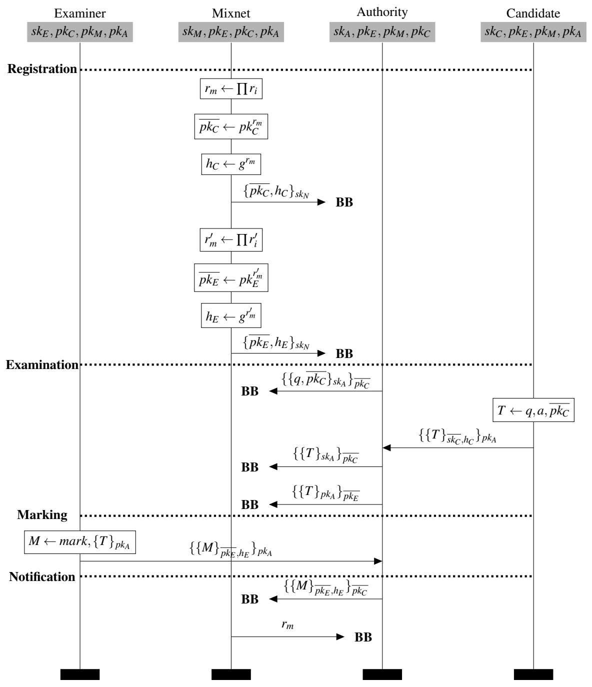
Examiner
sk_E, pk_C, pk_M, pk_A
Mixnet
sk_M, pk_E, pk_C, pk_A
Authority
sk_A, pk_E, pk_M, pk_C
Candidate
sk_C, pk_E, pk_M, pk_A
Registration
$r_m \leftarrow \prod r_i$
$\overline{pk_C} \leftarrow pk_C^{r_m}$
$h_C \leftarrow g^{r_m}$
$\{\overline{pk_C}, h_C\}_{sk_N}$
BB
$r'_m \leftarrow \prod r'_i$
$\overline{pk_E} \leftarrow pk_E^{r'_m}$
$h_E \leftarrow g^{r'_m}$
$\{\overline{pk_E}, h_E\}_{sk_N}$
BB
Examination
$\{\{q, \overline{pk_C}\}_{sk_A}\}_{\overline{pk_C}}$
BB
$T \leftarrow q, a, \overline{pk_C}$
$\{\{T\}_{\overline{sk_C}, h_C}\}_{pk_A}$
$\{\{T\}_{sk_A}\}_{\overline{pk_C}}$
BB
$\{\{T\}_{pk_A}\}_{\overline{pk_E}}$
BB
Marking
$M \leftarrow mark, \{T\}_{pk_A}$
$\{\{M\}_{\overline{pk_E}, h_E}\}_{pk_A}$
Notification
$\{\{M\}_{\overline{pk_E}, h_E}\}_{\overline{pk_C}}$
BB
r_m
BB

Figure 3: Remark! protocol diagram.

3.6 Applications

We perform formal analysis of three cryptographic protocols using Exponentiation Mix-Nets for different privacy purposes. A formal analysis of the electronic exam protocol has been presented in [23] where the protocol has been proved secure and no attacks have been found against privacy properties. Regarding the two other protocols, to the best of our knowledge, we present their first formal analysis. The timings for the results of the formal analysis given below were done on macOS with a M1 processor and 16 GB of RAM. All ProVerif files are available online [2]. We use a public Bulletin Board, denoted **BB**. A Bulletin Board is a public append-only (*i.e.*, nobody can delete) broadcast message channel. We also assume that the exponentiation Mix-Net behaves correctly. For simplicity, we have modeled the exponentiation Mix-Nets performed by all servers as a single honest Mix-Net server. All the aforementioned protocols are analyzed with and without the proposed fix described in Section 3.5.

3.6.1 Remark! Protocol

We first analyze the Remark! protocol, an electronic exam protocol, designed by Giustolisi *et. al* [27] and meant to achieve multiple privacy properties without relying on trusted parties. The following description of the protocol is based on [23, 27].

Protocol Description. The Remark! protocol involves four types of parties: Mix-Net servers, the candidate(s) sitting the exam, the examiner(s) correcting the answers and marking them, and an exam authority collecting the answers, dispatching them for the marking phase and delivering the final marks. It is assumed that each party is given a pair of public/private ElGamal keys with a common generator g, *i.e.,* the private key x and the public key $y = g^x$. The protocol runs in four phases: Registration, Examination, Marking and Notification. The protocol's sequence diagram is depicted in Figure 3.

Registration. The registration phase uses an exponentiation Mix-Net to generate pseudonyms $\overline{pk_C}$ and $\overline{pk_E}$ for both candidates C and examiners E based on their public keys pk_C and pk_E as described in Section 3. Let r_m and r'_m denote the Mix-Net's exponents related to candidates and examiners respectively, and let $(h_C = g^{r_m}, h_E = g^{r'_m})$ be the new generators.

Examination. The exam authority begins by signing (using its secret key sk_A) and encrypting (with the candidates' pseudonyms $\overline{pk_C}$) the questions q and publishes the result on the bulletin board **BB**. Then, the exam authority collects the candidates' answers a (which are signed with the candidates' pseudonym keys, and encrypted with the authority's key pk_A), verifies the signatures, resigns them, encrypts them using the corresponding candidates' pseudonyms and publishes them.

Marking. The exam authority encrypts the answers with examiners pseudonyms $\overline{pk_E}$ and publishes them. Each examiner marks the received tests with a mark *mark*, signs them using its pseudonym, encrypts them with the pkA and sends the marked tests back to the authority.

Notification. The authority receives the marks, verifies the signatures of the examiners and publishes the signed marks encrypted with the candidates' pseudonyms. Finally, the Mix-Net servers de-anonymize the candidates' pseudonyms by revealing the secret exponent r_m.

Formal Analysis. Dreier *et. al* [23] formalized privacy properties for electronic exams. Here we focus on *Anonymous Marking* and *Anonymous Examiner*'s informal definitions. An e-exam protocol ensures *Anonymous Marking* if a process where two candidates C_1 and C_2 answer a_1 and a_2 respectively, is indistinguishable from a process where the candidates switch their answers. Similarly, it ensures *Anonymous Examiner* if a process where two examiners E_1 and E_2 grade exam forms f_1 and f_2 respectively, is indistinguishable from a process where the examiners switch the forms.

The mentioned properties were proven satisfied with ProVerif on the model given in [23]. This model uses the abstract model for the Mix-Net described in the beginning of Section 3.3. We took the existing ProVerif files and replaced the abstract model of the Mix-Net with our refined model, and checked the same properties. Now, ProVerif finds attacks on Anonymous Examiner and Anonymous Marking. The results of the analysis are depicted in Table 5.

Anonymous Marking and Anonymous Examiner are not satisfied because candidates C_i sign their answer a_i with their pseudonyms $\overline{pk_{C_i}}$, and examiners sign their marks similarly; while pseudonyms generated by Mix-Net are linkable using the attack described in Section 3.1. Such vulnerabilities break the expected fairness of the marking procedure and also might make examiners vulnerable to coercion. The results of the analysis are depicted in Table 5.

Protocol	ZKP	Property	Result	Time
Remark! [27]	without	Anonymous Marking	✗	3 m 16 s
		Anonymous Examiner	✗	4 m 19 s
	weak	Anonymous Marking	✗	9 m 35 s
		Anonymous Examiner	✗	9 m 23 s
	strong	Anonymous Marking	✓	11 s
		Anonymous Examiner	✓	7 s
Haenni Voting [30]	without	Vote Privacy	✗	4 m 35 s
	weak		✗	9 m 35 s
	strong		✓	14 s
Crypto Santa [42]	weak	Anonymous Shuffling	✗	4 m 6 s
	strong		✓	9 s
IVXV [1]	without	Vote Privacy	✗	1 s
	weak		✗	25 s
	strong		✓	8 s

Table 5: Results of our analysis, with and without the added ZKP. Crypto Santa required a ZKP from the start.

3.6.2 Crypto Santa Protocol

The second protocol we have analyzed is the Crypto Santa protocol. It is based on a Christmas tradition called Secret Santa. During this ceremony, the participants are randomly assigned a person to whom they give a gift. In particular, the identity of the gift giver is to remain a secret. The Crypto Santa protocol is a cryptographic protocol designed by P. Ryan [42] to cryptographically implement the Secret Santa tradition. The protocol is designed to protect the anonymity of the gift giver, and is essentially based on an Exponentiation Mix-Net.

Protocol Description. The participants are n players. Each player is assumed to have a pair of a public and secret keys (pk_i, sk_i) where $pk_i = g^{sk_i}$. The public keys are arranged into a list $\tilde{L} = (pk_1, \ldots, pk_n)$ such that the key pk_i belongs to the player P_i. Given a list L let $L[j]$ denote the j^{th} term of the list L. The players take the list of public keys in turns, and each player i performs an exponentiation with s_i and shuffling step.

The final output is the list $L_n = \Pi_n(L_{n-1}[1]^{s_n}, \ldots, L_{n-1}[n]^{s_n})$ along with the final generator $g^s = g^{s_1 s_2 \cdots s_n}$. Note that the position of each pseudonym in the final outputted list L_n is relevant, as it determines a player to whom the gift is offered. If $L_n[j] = pk_i^s$, then P_i presents their gift $gift_i$ to the player P_j.

To prevent a player from cheating, the protocol suggests that each player provides two different Zero-Knowledge Proofs: a proof that he has performed the shuffle correctly (based on the specifications from [45]), and a proof of knowledge of their secret key sk_i using a standard ZK proof of the discrete log [42]. We do not give further details about the proof of shuffle since in our model we assume that the shuffle procedure is performed correctly.

Formal Analysis. In [42] there is no formal definition of the security properties ensured by Crypto Santa proto-

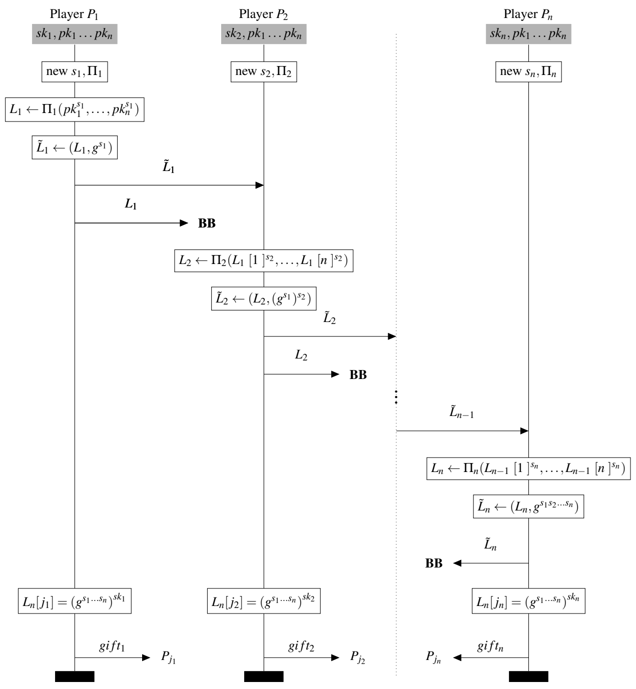

Figure 4: Crypto Santa Protocol

col. Nevertheless, they emphasized the fact that the gift giver should be anonymous. Hence, in the final list, the players' pseudonyms should be anonymized. Thus, the Crypto Santa protocol should probably guarantee Anonymous Shuffling (Def. 3.3). The results of our analysis are depicted in Table 5: Anonymous Shuffling is satisfied in case of strong ZKPs, but broken in case of weak ZKPs. Note that the original paper [42] does not specify which ZKP is to be used, and weak ZKPs are widely used [19].

3.6.3 Haenni's Internet Vote Protocol

The next protocol we analyze is an internet vote protocol designed by Haenni *et. al* [30]. This paper also introduced the concept of exponentiation Mix-Nets. The security of this protocol relies on the anonymity obtained by shuffling the voters' public keys. Casting a vote consists in signing the encrypted candidate choice with the anonymized public key generated by the Mix-Net. To obtain the election result, votes carrying a valid signature are decrypted and counted.

Protocol Description. This voting protocol involves four types of parties: the anonymizers, which are the Mix-Net servers, the voters, the talliers and an election authority. The protocol runs in four phases: Registration, Preparation, Vote Casting and Tallying. We assume an anonymous channel $\mathcal{C}$ between the voters and the Bulletin Board. The protocol's sequence diagram is depicted in Figure 5.

Registration. The election authority begins by setting up a Public Key Infrastructure for the voters. Each voter id_{V_i}

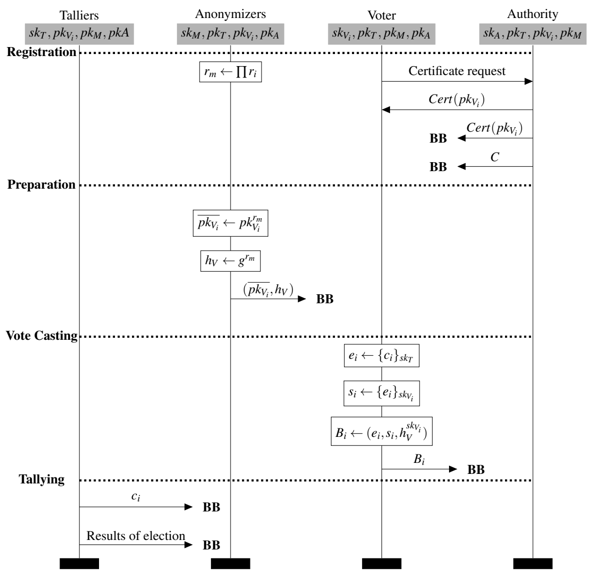

Figure 5: Haenni voting protocol sequence diagram.

is therefore equipped with a key pair $(sk_i, pk_i = g^{sk_i})$ and a public certificate binding their public key pk_i to their identity id_{V_i}.

Election Preparation. During this phase the election authority publishes the set C of possible candidates. The authority has to publish all certificates corresponding to eligible voters. Let $Y = \{y_1, \ldots, y_n\}$ be the list of voters' public keys. The anonymizers take as input Y and output the shuffled list $\tilde{Y} = \{\tilde{y_1}, \ldots, \tilde{y_n}\}$ together with the new generator $\tilde{g}$. The talliers jointly generate a public key using a threshold encryption scheme. The talliers jointly generate a public key using a threshold encryption scheme.

Vote Casting. Let $c_i \in C$ be an eligible candidate. To vote for c_i, a voter v_i needs to encrypt c_i with the talliers' public key y, signs the encrypted vote e_i with their private key sk_i, computes their pseudonym based on the new generator $\tilde{g}$ created by the anonymizers, and submits the ballot $B_i = (e_i, s_i, \tilde{g_i}^{sk_i})$ such that s_i corresponds to the encrypted and signed vote.

Tallying. Let $B = (e, s, \tilde{y})$ be a ballot. For a vote to be considered in the tally, the following conditions have to be satisfied: $\tilde{y}$ is a valid pseudonym, s is a valid signature for e and B is the only valid entry for $\tilde{y}$ in **BB**. The talliers decrypt all valid votes individually, and determine the final election result by applying the counting function to the resulting plaintexts. They also provide proofs of correct decryption.

Formal Analysis. The main security property related to vote protocols is *Vote Privacy*. We informally recall the definition proposed by Delaune *et. al* [21]. In a nutshell, considering two voters V_1 and V_2 and their votes c_1 and c_2, respectively, a voting protocol respects vote privacy whenever a process where V_1 votes c_1 and V_2 votes c_2 is observationally equivalent to a process where V_1 votes c_2 and V_2 votes c_1. This means that an attacker is not able to detect whether arbitrary honest voters V_1 and V_2 swapped their votes or not. The result for Vote Privacy is depicted in Table 5. Once again, in the absence or in the presence of weak ZKPs, there is an attack; in the presence of strong ZKPs, the property is verified.

The attack found by ProVerif linking the pseudonyms generated by Mix-Net (which is used by the voters to sign their votes) and the identities of the voters is quite interesting as it also relies on our refined model of ElGamal encryption and is slightly different from the attack described in Section 3. More precisely, let us consider two voters V_i and V_j with secret keys skV_i and skV_j with corresponding public keys $pkV_i = g^{skV_i}$ and $pkV_j = g^{skV_j}$ respectively. An attacker who wants to track V_i would choose as public key $pkV_i{}^s$. The outputted list of the pseudonyms generated by the Mix-Net contains $pkV_i{}^{r_m}$ (the pseudonym of V_i), $pkV_j{}^{r_m}$ (the pseudonym of V_j) and $pkV_i{}^{sr_m}$, the attacker's pseudonym. To test whether a pseudonym $\tilde{g}$ belongs to the voter V_i, the attacker generates a plaintext m and encrypts it using their pseudonym h as the public key and using $\tilde{g}$ as the basis. If the attacker succeeds in decrypting the ciphertext using their secret key, then the pseudonym used as the basis for ElGamal encryption corresponds to V_i's pseudonym. We can see as follows why this attack works. Let c be the ciphertext for the plaintext m. Then, $c = (c_1, c_2)$ such that $c_1 = \tilde{g}^r = (pkV_k{}^{r_m})^r$ and $c_2 = m \cdot h^r = m \cdot (pkV_i{}^{sr_m})^r$. Using their secret key s the attacker tries to decrypt c. If $k = i$ then $c_2 \cdot c_1^{-s} = m$. ProVerif was only able to find such an attack because of our refined model of ElGamal encryption.

4 Re-Encryption Mix-Nets

Re-Encryption Mix-Nets were proposed by Park *et al.* [37]. In re-encryption Mix-Nets, both the input and output lists are ciphertexts encrypted within the same public-key encryption scheme. These Mix-Nets rely on homomorphic operations that allow the servers to re-randomize ciphertexts, thereby re-encrypting the corresponding plaintexts. Consider the usual ElGamal setup. We have a list of n ciphertexts $\langle C_i \rangle = \langle C_1, \ldots, C_n \rangle$ such that $C_i = (g^{r'_i}, m_i h^{r'_i})$, where m_i a plaintext, and m mix servers M_j. The first mix server M_0 takes the original list of ciphertexts $\langle C_i \rangle$, and for each ciphertext C_i generates a fresh random coin $r_i \in \{0, \ldots, q-1\}$ and re-randomizes the ciphertext with r_i to obtain $C'_i = (g^{r'_i} g^{r_i}, m_i h^{r'_i} h^{r_i})$. Then M_0 permutes the resulting terms using a secret random order and sends the new list to the next mix server. The following mix servers repeat these same steps. The last mix server M_{m-1} outputs a list of ciphertexts which encrypt the permuted input messages, initially chosen by the senders, encrypted using different and secret random values.

4.1 Attacks against Re-Encryption Mix-Nets

Despite the fact that re-encryption Mix-Nets were specifically designed to guarantee anonymity, Pfitzmann [39] described

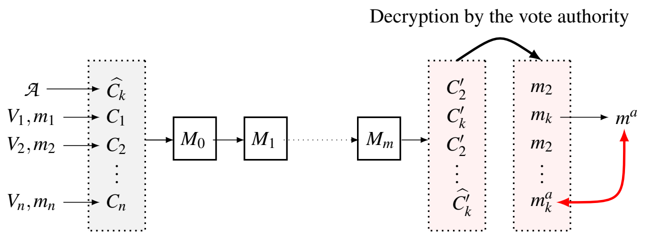

Figure 6: Attack against Re-encryption Mix-Nets, where $\widehat{C}_k = (g^{r_k a}, (m_k h^{r_k})^a)$.

the attack depicted in Figure 6 in the context of electronic voting. In this case encrypted ballots are sent to a Mix-Net. Once the ballots are mixed, the authority decrypts all the mixed encrypted ballots, and publishes the plaintext values in order to ensure verifiability properties. The attack considers a malicious voter $\mathcal{A}$ participating in the protocol. The attacker has access to the list of the Mix-Nets' input $\langle C_i \rangle$ of all participants. The attacker wants to learn the plaintext corresponding to a specific ciphertext $C_k \in \langle C_i \rangle$ such that $C_k = (g^{r_k}, m_k h^{r_k})$. $\mathcal{A}$ can generate $a \in \{1, \ldots, q-1\}$, raise both components of C_k to the power a, and send the ciphertext $\widehat{C}_k = (g^{r_k a}, (m_k h^{r_k})^a)$ to the Mix-Net as its vote. When the output list of encrypted messages is sent to the authority, all plaintexts m_i are published, and $\mathcal{A}$ knows that there is one output value m_i that is equal to m^a_j. Hence, $\mathcal{A}$ looks for this collision by raising each plaintexts to the power of a and looking for a match with the others. This way $\mathcal{A}$ can identify the plaintext m_i of the victim.

Adding zero knowledge proof of possessing the randomness would fix the attack described above, as the attacker only knows a, but not r_k. However, when the weak variant is used, since the first component of the honest ciphertext is g^{r_k} and $(g^{r_k})^a$ for the malicious one, the attacker can construct an invalid ZKP for a fixed a as it is shown in Section 3.5: let (c, f) be the ZKP for g^{r_k} with the commitment $B = g^b$, then he can compute $c' = \mathcal{H}(B^{c^{-1}})$ and $f' = c^{-1} f$, and choose $a = c'^{-1}$, and submit the ciphertext $((g^{r_k})^{c'^{-1}}, (m_k h^{r_k})^{c'^{-1}})$ with the proof (c', f'). This enables them to bypass the verification of the ZKP as $\mathcal{H}(g^{f'} \cdot ((g^{r_k})^{c'^{-1} c'})^{-1}) = \mathcal{H}(g^{c^{-1}(b+cr_k)} \cdot (g^{r_k})^{-1}) = \mathcal{H}(g^{c^{-1} \cdot b + r_k} \cdot g^{-r_k}) = c'$.

4.2 Formal model and analysis using ProVerif

For our formal mode of re-encryption mixnets, we define a Reenc-mixnet instance, analogous the exp-mixnet instance.

Definition 4.1 (Reenc-mixnet instance) *A reenc-mixnet instance is a closed process $RP = \nu \tilde{n} \cdot (S\sigma_{id_1}\sigma_{m_1} \mid \ldots \mid S\sigma_{id_j}\sigma_{m_j} \mid M\sigma_{r_1} \mid \ldots \mid M\sigma_{r_k} \mid A\sigma_{sk_A})$ where $\tilde{n}$ is the set of all restricted names, which includes the set of the protocol's private channels; $S\sigma_{id_i}\sigma_{m_i}$ are the processes run by the senders with the substitutions specifying* *the identity and the plaintext of the i^{th} sender respectively; $M\sigma_{m_j}$ is the process executed by the Mix-Net servers with the substitution specifying their secret random; A is the process run by the decryption authority and σ_{sk_A} specifies the decryption key.*

This definition allows us to apply the same definition of *Anonymous Shuffling* (Def. 3.3). Applied to re-encryption Mix-Nets, Anonymous Shuffling requires that the process where id_1 encrypts m_1 and id_2 encrypts m_2, is equivalent to the process where id_2 encrypts m_1 and id_1 encrypts m_2. This prevents the attacker from having any information about which plaintext belongs to which sender.

Equational theory. Re-encryption Mix-Nets relies on the homomorphic properties of the encryption scheme since Mix-Net servers need to re-encrypt the list of the inputted ciphertexts. Therefore, in addition to the encryption equation, we require an additional equation that models a re-encryption operation. We use the following equational theory:

$$\begin{aligned}
&dec(enc(m,X,exp(X,x),r),X,x) = m \\
&reenc(enc(m,X,exp(X,x),r),r',X,exp(X,x)) = \\
&\qquad\qquad enc(m,X,exp(X,x),sum(r,r')) \\
&EXP(enc(m,X,exp(X,x),r),a) = \\
&\qquad\qquad enc(exp(m,a),X,exp(X,x),mult(r,a))
\end{aligned} \tag{6}$$

The first equation corresponds to our refined model of ElGamal described in Section 3.4. Let $(c_1 = g^r, c_2 = m \cdot (g^x)^r)$ be an ElGamal ciphertext. Re-encrypting the latter ciphertext consists in generating a new random r' and multiplying c_1 with $g^{r'}$, and c_2 with $(g^x)^{r'}$. The resulting ciphertext is of the form $(g^{r+r'}, m \cdot (g^x)^{r+r'})$. To model the re-encryption operation, we introduce two function symbols *sum* and *reenc*. The binary function *sum* represents a sum of two exponents (albeit in a limited way, as *sum* has no further equations – ideally *sum* should be associative and commutative, but this is not possible in ProVerif). The function *reenc* takes as arguments a ciphertext $enc(m,X,exp(X,x),r)$, a coin r', a basis X and a public key $exp(X,x)$. It returns the ciphertext $enc(m,X,exp(X,x),sum(r,r'))$ with a coin representing the sum of r and r'. The first and the second equations are needed for the protocol to work as it uses encryption and re-encryption, but they are insufficient to capture attacks against re-encryption Mix-Nets. To capture the attack depicted in Figure 6, we need to augment our equational theory with an equation that allows for the exponentiation of the ciphertext with respect to the induced homomorphic properties on the corresponding plaintext. The third equation uses the binary function EXP (different from *exp*) to apply a known exponent a over the plaintext. We introduce the function *mult* to represent a multiplication between random coins (again, in a limited way). When analyzing re-encryption Mix-Nets using ZKPs to show knowledge of the randomness used for the encryption, we also added Equations (5) and (4) to the equational theory.

Model. The process for each Mix-Net server *M* is similar to the one defined for exponentiation Mix-Net (as depicted in Figure 2), but it uses different parameters. Each Mix-Net server inputs the list of all generated ciphertexts, checks they are distinct, applies a random coin to each ciphertext in order to re-randomize it and outputs the list of all new ciphertexts in random order. The sender simply outputs its ciphertext. Thus, the sender process is: let S(m_S, r_S, pk) = out(ch, (enc(m_S, g, pk, r_S))).

Analysis. The result of the analysis of re-encryption Mix-Nets is given in Table 4. ProVerif concludes that *Anonymous Shuffling* is not satisfied and finds the attack depicted in Figure 6 when the protocol is modeled without ZKP. In the case of the weak ZKP, ProVerif finds (nearly) the same attack, except that the attacker need to fake the ZKP as in the attack on the exponentiation Mix-Net with weak ZKP. In contrast, the property is verified when the strong ZKP is used instead.

4.3 Application: IVXV Internet Vote Protocol

In this section, we analyze the Estonian internet vote protocol which uses a re-encryption Mix-Nets [1]. It is used to conduct legally binding political elections in Estonia. We use the equational theory described in Equation (6). The description of the protocol is based on the one given by Müller in [36].

Protocol Description. The Voting System consists of the Election Organizer EO, the Vote Collector VC, the Registration Service RS, the I-Ballot Box Processor $IBBP$, the Talliers T and Mix-Nets MS. The protocol runs in four phases: *Registration*, *Vote Casting*, *Preparation* and *Tabulation*.

Registration. This scheme requires a Public Key Infrastructure. Each eligible voter v_i is therefore equipped with a key pair (sk_{v_i}, pk_{v_i}) and a public certificate $Cert_{v_i}$ binding their public key pk_{v_i} to their identity v_i. The EO publishes the set $C = \langle c_j \rangle$ of possible candidates. The talliers T jointly generate a public key pk_T using a threshold encryption scheme.

Vote Casting. To vote for a candidate c_j, a voter v_i encrypts first c_j with the Talliers' public key pk_T and a generated coin rv_i. The voter signs the encrypted ballot Bc_i with their private key skv_i and sends their signed encrypted ballot Bcs_i along with their certificate $cert_{v_i}$ and their identifier v_i to VC. Upon receiving a valid $(Bcs_i, cert_{v_i}, v_i)$, VC responds with a unique generated identifier vid_i and the RS confirmation reg_{vid_i}.

Preparing for Tabulation. After the online voting phase, the VC has a set of digitally signed votes and RS has the set of registration queries and responses. Both of these sets are transferred to the $IBBP$ responsible for auditing the voting phase and pre-processing the votes for tabulation/ $IBBP$ composes a new list of signed vote envelopes. This list only contains the latest vote for each voter v_i since the protocol allows voters to revote. $IBBP$ then anonymizes the votes in the list by extracting only the encrypted ballots. $IBBP$ can then pass the

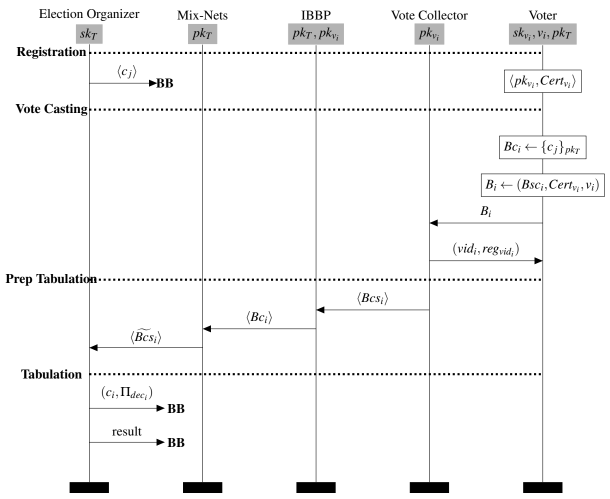

Figure 7: Estonian voting protocol sequence diagram.

list $\langle Bc_i \rangle$ to the re-encryption Mix-Nets MS. The output set of the Mix-Nets $\langle \widetilde{Bc_i} \rangle$ is then sent to EO for tabulation.

Tabulation. The EO uses the election private key to decrypt each choice c_i and to compute the result. EO also provides a proof of correct decryption Π_{dec_i} for every plaintext.

Formal Analysis. We analyze *Vote Privacy* as described in the previous protocol analysis. The input list to the Mix-Nets in IVXV protocol is not public, but is given to a third party performing audits. The re-encryption Mix-Net was added to the protocol to preserve the privacy of voters with respect to this third party performing audits. Hence, vote privacy in the IVXV protocol relies on the anonymity provided by the shuffling procedure performed by the re-encryption Mix-Net.

We verify vote privacy using our new refined model of the Mix-Net. The result of the analysis is depicted in Table 5: ProVerif finds an attack on Vote Privacy. Assuming a malicious third party, the attacker can retrieve the encrypted ballot Bc_i of his victim v_i from the ballot list and perform the attack described in Section 4.1. At the end of the election, when the results are published, the attacker knows to which candidate c_j the voter v_i voted. In [36], Müller found that the IVXV protocol is vulnerable to similar attacks exploiting the malleability of the ElGamal encryption when investigating the protocol manually. To fix this issue, he added ZKPs proving that the voter knows the randomness and the plaintext of the encrypted ballot. We also verified the protocol with an added weak or strong ZKP showing knowledge of the randomness (only), using the modeling described in Section 3.5. In the case of a weak ZKP the attack persists, whereas with a strong ZKP ProVerif succeeds in proving the property (Table 5).

5 Discussion and Conclusion

In this paper we propose a novel and more precise modeling of exponentiation Mix-Nets which includes the details of the exponentiation. Previous models used a high-level abstraction of the functionality, which lead them to miss attacks based on a weakness where a user can submit a modified version of a key of another participant in an attempt to trace them.

Using three case studies (including a voting and an exam protocol) we show that we can use our improved modeling to analyze protocols using exponentiation Mix-Nets. In particular, we are able to (re-)discover known and unknown attacks on these protocols. Fixing these attacks requires the use of zero-knowledge proofs. We propose two models: a novel model for weak ZKPs vulnerable to certain attacks, and a model for strong ZKPs. We can show automatically that in our examples the use of weak ZKPs is insufficient, as all attacks persist. The use of strong ZKPs however fixes the attacks, and we can verify the protocols.

Moreover, we propose a novel and more precise modeling of ElGamal encryption where keys are actually the result of exponentiation operations, which is of independent interest and can be applied to other protocols, even if they do not use Mix-Nets. We also propose a refined modeling of re-encryption Mix-Nets, which allows use to rediscover a known attack on the Estonian voting protocol. Again, we can automatically show that the attack persists when adding weak ZKPs, but disappears if strong ZKPs are used.

As future work, we would like to apply our ElGamal and ZKP models to other protocols using these primitives, such as other voting protocols, private set intersection protocols or password based authentication protocols. Essentially, all existing ProVerif models of protocols using these primitives could be extended using our new refined modeling. We hope that in the future such a modeling will become commonplace.

Moreover, we would like to get rid of some of the remaining limitations of the current model in ProVerif. For example, in the current equational theories, we use a fixed generator, which in particular limits the number of exponentiations, and some attacks need more than three exponentiations [38]. In general, there are results showing that a fixed number can be sufficient (e.g., [35]), which are however not directly applicable in this case.

We would also like to propose models capturing other attacks on weak zero-knowledge proofs.

Finally, we do not consider the scenario of corrupted Mix-Nets. To model one or more corrupted Mix-Nets, additional details have to be considered within the model. The assumption of having at least one single honest server among the ones in Mix-Nets is sufficient to have an honest mixing process, as stated for example in [27]. However, this assumption is not always true, as shown in [44]. The fifth and the second attack required that the first mix-server is corrupted whereas the third and the fourth ones required that both the first and the last mix-servers are corrupted.

Acknowledgments

This work benefited from funding managed by the French National Research Agency (ANR) under the France 2030 programme with the reference ANR-22-PECY-0006. The authors are also grateful for the support provided by the ANR through the ANR SEVERITAS (ANR-20-CE39-0009) and the ANR MobiS5 (ANR-18-CE39-0019) projects.

References

[1] Specification of IVXV estonian voting protocols, 2023. https://www.valimised.ee/sites/default/files/2023-02/IVXV-protocols.pdf.

[2] Proverif codes. https://tinyurl.com/mw3syfmr, 2024.

[3] Martín Abadi, Bruno Blanchet, and Cédric Fournet. The Applied Pi Calculus: Mobile Values, New Names, and Secure Communication. *Journal of the ACM (JACM)*, 65(1):1 – 103, October 2017.

[4] Michael Backes, Fabian Bendun, and Dominique Unruh. Computational soundness of symbolic zero-knowledge proofs: Weaker assumptions and mechanized verification. In David Basin and John C. Mitchell, editors, *Principles of Security and Trust*, pages 206–225, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[5] Gilles Barthe, Benjamin Grégoire, Santiago Zanella-Béguelin, and Sylvain Heraud. Computer-Aided Security Proofs for the Working Cryptographer. In *Advances in Cryptology - {CRYPTO} 2011 - 31st Annual Cryptology Conference*, Santa Barbara, United States, 2011.

[6] David A. Basin, Jannik Dreier, and Ralf Sasse. Automated symbolic proofs of observational equivalence. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 1144–1155. ACM, 2015.

[7] David Bernhard, Olivier Pereira, and Bogdan Warinschi. How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios. pages 626–643, 12 2012.

[8] Bruno Blanchet. A computationally sound mechanized prover for security protocols. *Dependable and Secure Computing, IEEE Transactions on*, 5:193 – 207, 01 2009.

[9] Bruno Blanchet. Automatic Verification of Security Protocols in the Symbolic Model: The Verifier ProVerif. In Alessandro Aldini, Javier Lopez, and Fabio Martinelli, editors, *Foundations of Security Analysis and Design VII*, volume 8604 of *Lecture Notes in Computer Science*, pages 54–87. Springer, 2014.

[10] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, feb 1981.

[11] Vincent Cheval, Cas Cremers, Alexander Dax, Lucca Hirschi, Charlie Jacomme, and Steve Kremer. Hash gone bad: Automated discovery of protocol attacks that exploit hash function weaknesses. In Joseph A. Calandrino and Carmela Troncoso, editors, *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, pages 5899–5916. USENIX Association, 2023.

[12] Vincent Cheval, Raphaëlle Crubillé, and Steve Kremer. Symbolic protocol verification with dice: process equivalences in the presence of probabilities. In *2022 IEEE 35th Computer Security Foundations Symposium (CSF)*, pages 319–334, 2022.

[13] Vincent Cheval, Steve Kremer, and Itsaka Rakotonirina. The DEEPSEC prover. In Hana Chockler and Georg Weissenbacher, editors, *Computer Aided Verification - 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part II*, volume 10982 of *Lecture Notes in Computer Science*, pages 28–36. Springer, 2018.

[14] Véronique Cortier, Pierrick Gaudry, and Stéphane Glondu. Features and usage of Belenios in 2022. In *The International Conference for Electronic Voting (E-Vote-ID 2022)*, Bregenz / Hybrid, Austria, October 2022.

[15] Véronique Cortier and Ben Smyth. Attacking and fixing helios: An analysis of ballot secrecy. *Journal of Computer Security*, 21:297–311, 06 2011.

[16] Cas Cremers, Alexander Dax, Charlie Jacomme, and Mang Zhao. Automated Analysis of Protocols that use Authenticated Encryption: Analysing the Impact of the Subtle Differences between AEADs on Protocol Security. In *USENIX Security 2023*, Anaheim, United States, August 2023. USENIX.

[17] Cas Cremers and Dennis Jackson. Prime, order please! revisiting small subgroup and invalid curve attacks on protocols using diffie-hellman. In *2019 IEEE 32nd Computer Security Foundations Symposium (CSF)*, pages 78–7815, 2019.

[18] Chris Culnane, Peter Y. A. Ryan, Steve Schneider, and Vanessa Teague. Vvote: A verifiable voting system. *ACM Trans. Inf. Syst. Secur.*, 18(1), jun 2015.

[19] Quang Dao, Jim Miller, Opal Wright, and Paul Grubbs. Weak fiat-shamir attacks on modern proof systems. In *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*, pages 199–216. IEEE, 2023.

[20] S. Delaune, S. Kremer, and M. Ryan. Coercion-resistance and receipt-freeness in electronic voting. In *19th IEEE Computer Security Foundations Workshop (CSFW'06)*, pages 12 pp.–42, 2006.

[21] Stéphanie Delaune, Steve Kremer, and Mark Ryan. Verifying privacy-type properties of electronic voting protocols. *J. Comput. Secur.*, 17(4):435–487, dec 2009.

[22] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[23] Jannik Dreier, Rosario Giustolisi, Ali Kassem, Pascal Lafourcade, Gabriele Lenzini, and Peter Y. A. Ryan. Formal analysis of electronic exams. In *2014 11th International Conference on Security and Cryptography (SECRYPT)*, pages 1–12, 2014.

[24] Jannik Dreier, Hugo Jonker, and Pascal Lafourcade. Secure Auctions Without Cryptography (extended version). Technical report, ETH Zurich, April 2014.

[25] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proceedings on Advances in Cryptology—CRYPTO '86*, page 186–194, Berlin, Heidelberg, 1987. Springer-Verlag.

[26] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inf. Theory*, 31(4):469–472, 1985.

[27] Rosario Giustolisi, Gabriele Lenzini, and Peter Y. A. Ryan. Remark!: A secure protocol for remote exams. In *Security Protocols Workshop*, 2014.

[28] Kristian Gjøsteen. The norwegian internet voting protocol. In Aggelos Kiayias and Helger Lipmaa, editors, *E-Voting and Identity*, pages 1–18, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[29] Rolf Haenni, Reto Koenig, and Philipp Locher. Private internet voting on untrusted voting devices. In *7th Workshop on Advances in Secure Electronic Voting*. 2023.

[30] Rolf Haenni and Oliver Spycher. Secure internet voting on limited devices with anonymized DSA public keys. In *2011 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 11)*, San Francisco, CA, August 2011. USENIX Association.

[31] Thomas Haines, Rajeev Goré, and Bhavesh Sharma. Did you mix me? formally verifying verifiable mix nets in electronic voting. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*, pages 1748–1765. IEEE, 2021.

[32] Dennis Jackson, Cas Cremers, Katriel Cohn-Gordon, and Ralf Sasse. Seems legit: Automated analysis of subtle attacks on protocols that use signatures. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 2165–2180. ACM, 2019.

[33] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Formal analysis of chaumian mix nets with randomized partial checking. In *2014 IEEE Symposium on Security and Privacy*, pages 343–358, 2014.

[34] Daniele Micciancio and Bogdan Warinschi. Soundness of formal encryption in the presence of active adversaries. In Moni Naor, editor, *Theory of Cryptography*, pages 133–151, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[35] Sebastian Mödersheim. Diffie-hellman without difficulty. In Gilles Barthe, Anupam Datta, and Sandro Etalle, editors, *Formal Aspects of Security and Trust*, pages 214–229, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[36] Johannes Müller. Breaking and fixing vote privacy of the estonian e-voting protocol ivxv. In *Financial Cryptography Workshops*, 2022.

[37] Choonsik Park, Kazutomo Itoh, and Kaoru Kurosawa. Efficient anonymous channel and all/nothing election scheme. In *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*, volume 765 of *Lecture Notes in Computer Science*, pages 248–259. Springer, 1993.

[38] O. Pereira and J.-J. Quisquater. A security analysis of the cliques protocols suites. In *Proceedings. 14th IEEE Computer Security Foundations Workshop, 2001.*, pages 73–81, 2001.

[39] Birgit Pfitzmann. Breaking efficient anonymous channel. In *International Conference on the Theory and Application of Cryptographic Techniques*, 1994.

[40] Mohammadamin Rakeei, Rosario Giustolisi, and Gabriele Lenzini. Secure internet exams despite coercion. In *17th DPM International Workshop on Data Privacy Management*. Springer, 2022.

[41] A. Roscoe. *The Theory and Practice of Concurrency*. 01 2005.

[42] Peter Y. Ryan. Crypto santa. In *LNCS Essays on The New Codebreakers - Volume 9100*, page 543–549, Berlin, Heidelberg, 2015. Springer-Verlag.

[43] Efstathios Stathakidis, Steve Schneider, and James Heather. Robustness modelling and verification of a mix net protocol. pages 131–150, 12 2014.

[44] Douglas Wikström. Five practical attacks for "optimistic mixing for exit-polls". In Mitsuru Matsui and Robert J. Zuccherato, editors, *Selected Areas in Cryptography*, pages 160–174, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[45] Douglas Wikström. A sender verifiable mix-net and a new proof of a shuffle. In Bimal Roy, editor, *Advances in Cryptology - ASIACRYPT 2005*, pages 273–292, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[46] Burkhart Wolff, Oliver S, Hannes Federrath, Stefan Kispsell, and Andreas Pfitzmann. Towards a Formal Analysis of a Mix Network. Technical report, Institut für Informatik Albert–Ludwigs–Universität Freiburg, 2003.