Marked Mix-Nets

Olivier Pereira[1] and Ronald L. Rivest[2]

[1] UCLouvain (Louvain-la-Neuve, Belgium)
olivier.pereira@uclouvain.be
[2] MIT (Cambridge, MA)
rivest@mit.edu

Abstract. We propose a variant mix-net method, which we call a "marked mix-net" . Marked mix-nets avoid the extra cost associated with verifiability (producing a proof of correct mixing operation), while offering additional assurances about the privacy of the messages, compared to a non-verifiable mix-net.
With a marked mix-net, each mix-server adds an extra secret mark in each ciphertext, and the input ciphertexts are made non-malleable but still re-randomizable (RCCA).
Marked mix-nets appear to be a good fit for the mix-net requirements of voting systems that need a mix-net for anonymity but where correctness is guaranteed through independent mechanisms. Our work investigates applications to STAR-Vote, but other applications could be explored, e.g., in Prêt-à-Voter, Selene or Wombat.

1 Introduction

1.1 Mix-nets

Mix-nets were originally proposed by Chaum [12], then extended and elaborated by many others: additional details can be found in Adida [2], Sampigethaya et al. [24], and Wikström [31], for instance. They are a central tool for anonymizing a set of messages, like votes for instance, by breaking any observable connection between the messages it receives, and those it outputs.

More precisely, a mix-net server, or *mixer*, takes a sequence of encrypted messages and outputs them in permuted order, according a secret permutation that only it knows (sometimes the message encryption and decryption process are also included in the mix-net definition). Since the objective is to hide the permutation, the inputs and outputs to the mix-net must be not only encrypted somehow, but the outputs should be re-encrypted or encrypted differently than the inputs, so that the outputs can not be trivially matched with the corresponding inputs. This places additional requirements on the encryption methods used.

A mix-net can be just a single server, or a sequence of k mixers for some $k > 1$, each permuting its inputs according to its own secret permutation before sending its outputs along to be the inputs to the next mixers.

In some cases one may worry about malicious mixers who do not actually permute their inputs, but perform some other operation instead. For example, in a voting context, a mixer could replicate some inputs and delete others, causing a change in the vote tally. Because of the encryption, such manipulations may

be hard or impossible to detect; the only thing an observer can really tell is that the number of inputs is equal to the number of outputs.

For such applications one may use a *verifiable mix-net* [23,31,21,4,11]. Here each mixer produces an additional output, which is often a non-interactive zero-knowledge (NIZK) proof that it has operated correctly (i.e., that the set of messages one obtains by decrypting the inputs is the same as the set of messages one obtains by decrypting the outputs; the mixer only permuting things around). Anyone can verify the published NIZK proofs from each mixer.

The design of these NIZK proofs considerably improved over the years, and they certainly are among the most sophisticated cryptographic protocols deployed in real-world applications: they started to be increasingly used in private elections and trialed in public ones [9,14,27,28]. This sophistication also comes with computational complexity. For instance, Bayer and Groth [4] report in 2012 timings between 2 and 5 minutes for two types of state-of-the-art proofs computed for 100,000 ElGamal ciphertexts, computed in an order q subgroup of $\mathbb{Z}_p^*$ where $|q| = 160$ and $|p| = 1024$. More recently, the authors of Verificatum report timings around 12 minutes for the shuffle of 100,000 ElGamal ciphertexts, again in an order q subgroup of $\mathbb{Z}_p^*$, but with the more contemporary security parameters $|q| = 3247$ and $|p| = 3248$ [29].

While remarkably efficient, such numbers can become a potential obstacle when running a large-scale public election. Considering for instance an election in a mid-size city with 500,000 voters and 100 races that need to be mixed independently (in order to avoid pattern attacks), the computation of a proof would take between 16 and 100 hours. In such a context, and given the typical time-frame of elections, organizers will require each mixer to use several powereful workstations in order to improve speed and parallelism. But such workstations increase the management load, the attack surface, and are likely to require IT staff, which need to be chosen independently for each mixer in order to avoid the creation of a single point of corruption. Overall, these requirements of hardware and experts can be expected to create important costs and organizational challenges.

Marked mix-nets aim at offering a considerably faster alternative that would be useful in some practical settings. Going back to our previous example, the 100 hours required when computing a proof of shuffle modulo a 3248 prime modulus become an online effort of around 13 minutes on a single laptop when using our marked mix-net, while maintaining all previous parallelism possibilities.

This speed improvement comes with a relaxation of the security guarantees offered by a fully verifiable mix-net.

1. The marked mix-net is only verifiable for privacy/anonymity, but not for correctness. If the marked mix-net verification procedure succeeds, then the link between the input and output ciphertexts is broken as soon as one mixer is honest, and assuming that correctness is verified independently (possibly in a statistical sense). There is indeed no guarantee that the output ciphertexts are a permutation of those at the input: a malicious mixer could remove some ciphertexts and insert new ciphertexts of his own while remaining undetected by the statistical correctness test, due to luck.

2. A marked mix-net targets security in front of a covert adversary [3]. This means that the marked mix-net aims at making any privacy/anonymity violation likely to be visible to an auditor, and it is the expectation that the sanctions that would result from any evidence of malicious behavior would be high enough to deter any such behavior. (Note that organizational measures can be taken so that a single judge audits the data before they are released, so that any privacy violation would only be visible to that judge, and not to the malicious party.)

We believe that these security properties are satisfactory for some applications, and that the operational simplifications resulting from the lower computational requirements makes marked mix-net an interesting option for some voting systems.

1.2 Applications of Marked Mix-nets

We were motivated to develop a marked mix-net by the need for a suitable mix-net in the STAR-Vote design [5]. However, other systems, including Selene [22], Wombat [6], or some variants of Prêt-à-Voter that use a human verifiable paper audit trail [16] could possibly adopt a marked mix-net as well.

In STAR-Vote, human-readable paper ballots are produced by ballot marking devices, together with an electronic and encrypted record of the votes. As ballots are collected, this electronic record is replicated and hash chained in various ways, in order to improve robustness and reliability. Furthermore, the content of some hash-chains is included in the voter take-home paper receipts, in order to support end-to-end verifiability. These features make it quite simple to identify which voter produced what ciphertext – and this is not meant to be a secret information.

STAR-Vote is designed to be end-to-end verifiable. Still, it is also designed to accommodate failures in the end-to-end verification process. For instance, the cast-as-indended audit process could fail to be performed on election day, or there might be a soundness issue in the zero-knowledge proofs that are used to prove the validity of the ballots. Therefore, STAR-Vote also supports an efficient ballot-level risk-limiting audit (RLA) [7], illustrated in Figure 1, which proceeds by comparing paper ballots randomly picked from the urns against electronic ballot records. In order to perform a matching between electronic and paper ballots, the encrypted votes need to be decrypted, but they need to be made anonymous first. STAR-Vote requires the use of a mix-net for that purpose.

The original specification for STAR-Vote indicates that the mix-net should be verifiable. But is a verifiable mix-net *really* needed for STAR-Vote? Perhaps not, as the RLA that it serves compares electronic records against paper records in order to detect if there is any significant malfeasance that would cause divergence between these records, and as such would detect a divergence coming from an incorrect mixing process. So, for integrity purposes, a verifiable mix-net may not be needed.

Privacy is a different concern, though: STAR-Vote still relies on the electronic process, and on the mix-net in particular, to guarantee that the ballots that are decrypted are anonymous. And the use of a non-verifiable mix-net can raise

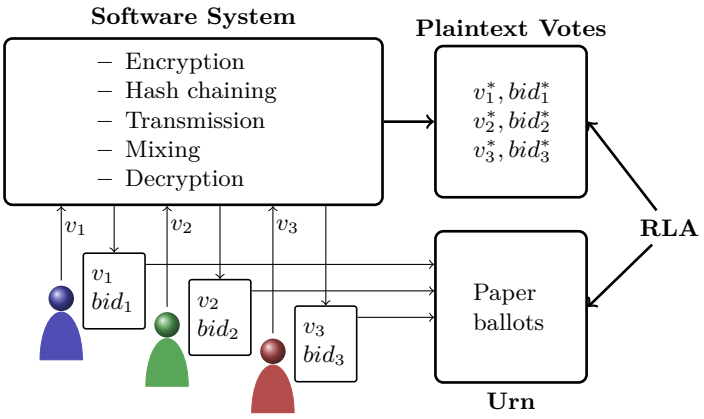

Fig. 1. Overview of the preparation of the inputs of the STAR-Vote risk limiting audit. Voters interact with the software system, which prints paper ballots. After voter verification, the paper ballots are placed into an urn. At the end of the election, the software outputs an anonymized list of plaintext votes. The risk limiting audit compares paper ballots and electronic ballots, referencing them by their ballot id.

important privacy issues. For instance, a corrupted last mixer could ignore the ciphertexts handed by the penultimate mixer, and mix those that were the input of the first mixer instead. As a result, this corrupted mixer would be able to deanonymize all cleartext votes after decryption, and this would be completely undetectable. Hence the initial recommendation of a verifiable mix in the STAR paper. But, as explained above, this is not an innocuous choice, both in terms of computational requirement and organizational complexity for the STAR-Vote implementers and auditors.

There are other systems that offer similar features, and for which the use of a marked mix-net could be envisioned.

For example, Wombat [6] is another system that uses both encrypted electronic records that are mixed, and human-readable paper records that can be used to verify the electronic tally. As such, using a marked mix-net in Wombat could increase the speed of the tally, as long as a (ballot polling) risk-limiting audit process is used to confirm the electronic tally based on the paper ballots. Note that the security model would be a bit different between Wombat and STAR-Vote: in STAR-Vote, the mix-net is only a component of the RLA that is applied on the result of a system that is end-to-end verifiable independently of it; while in Wombat, the verifiable mix-net is really a component of the end-to-end verifiability of the system, and moving to a non-fully verifiable mix makes the verification of the correctness of the election result rely on the RLA. Besides, the "delayed effect" attack detailed in Section 3.3 could apply.

Another example would be Selene [22], which uses two mix-nets: one for assigning tracker numbers to voters, and one for making the votes anonymous. While the use of a marked mix-net seems difficult for the tracker number as-

signment phase (e.g., duplication may be hard to detect), the use of a marked mix-net for the vote anonymization phase might be an interesting option. Similar observations can be made about a variant of Prêt-à-Voter proposed by Lundin and Ryan [16], that offers a human readable paper trail.

We do not make any claim about the exact consequences of using a marked mix-net in these systems, and leave these questions open for the moment, while focusing on STAR-Vote in this paper.

The following sections give details. Section 2 provides some background information on ElGamal encryption and an overview of the basic mix-nets techniques. Section 3 explains the design of our marked mix-net. Section 4 describes the risk-limiting process of STAR-Vote and how it could be adapted to use our marked mix-net, and Section 5 concludes.

2 Cryptographic Background

For concreteness, we present the new marked mix-net design using a variant of the ElGamal encryption scheme [15]. Our ideas should be portable to mix-nets based on other encryption schemes.

2.1 ElGamal Encryption

Assume that G is a group of prime order q, with generator g. The description of G, including q and g, are public parameters. An ElGamal secret key x is selected by drawing x uniformly at random from $F_q - 0$, and the corresponding public key is computed as $y = g^x$.

The encryption of a message m encoded as an element of G is computed as $E(y, m) = (mg^r, y^r)$ for a uniformly random element r from F_q. The decryption of a ciphertext (a, b) is easy using x: we can indeed define $D(x, a, b) = ab^{-1/x} = a(y^r)^{-1/x} = (mg^r)(g^{xr})^{-1/x} = m$ Note that x is invertible in F_q since x is nonzero. The security of the ElGamal encryption scheme relies on the hardness of the Decision Diffie-Hellman problem [8] in the group G.

There are various techniques for mapping bit strings into ElGamal messages in the group G, but these techniques depend on the choice of G. For the purposes of this paper, the specifics of this mapping do not matter and, when the context is clear, we will not make any distinction between m as a sequence of bits and m's encoding as an element of G.

ElGamal is (multiplicatively) *homomorphic*—the (componentwise) product of ciphertexts is a ciphertext for the product of the messages: $E(y, m_1) * E(y, m_2) = E(y, m_1 m_2)$. (Technically, the above means equality of sets of ciphertexts.) Because ElGamal is homomorphic, a ciphertext can be re-randomized knowing only the public key, by multiplying (componentwise) by an encryption of 1: $E(y, m) * E(y, 1) = E(y, m)$. Finally, ElGamal encryption is *malleable*. In particular, you can multiply the plaintext by a factor of b merely by multiplying one component of the ciphertext by b $(b, 1) * E(y, m) = (b, 1) * (mg^r, y^r) = ((bm)g^r, y^r) \in E(y, bm)$. We will make an intensive use of these two features in

our marked mix-net. (These features are also present in many other encryption schemes, but ElGamal is probably the simplest and most common example.)

2.2 Mix-Nets

In *re-encryption mix-nets* (our focus here), each mixer receives a sequence of n ciphertexts as input, to which it applies a random permutation, after which it re-randomizes each ciphertext in order to make ciphertexts unlinkable, and then outputs the result for the next mixer.

The inputs of a re-encryption mic-net can simply be encrypted with a single public key y and, as explained above, the mix-servers do not need to know the corresponding secret key x in order to re-randomize. The outputs of the last mix-server can be decrypted by a party who knows x. (In some cases, x may be secret-shared [25] by several parties, and a threshold number of such parties cooperate to decrypt the mix-net outputs [18].)

Looking at the re-randomization process of ElGamal, we can observe an interesting feature: the re-randomization is essentially the multiplication of two ciphertexts that are independent of each other. This means that a mixer can, in an offline phase, before he sees any ciphertext, compute a collection of encryptions of "1" $\in G$. Then, the online phase can simply consist in ciphertext multiplications, which is orders of magnitude faster than computing a ciphertext (the exact factor being strongly dependent on the choice of G). This is the property we aim at preserving when designing our marked mix-net. In particular, this excludes so-called decryption mix-nets, in which each mixer would perform a partial decryption, as this would cause a latency of at least one modular exponentiation per ciphertext and per mixer.

Notation: We let k denote the number of mix servers. We let m_i denote the ith input message, for $1 \leq i \leq n$. We let $c_i^{(j)}$ denote the ith input to the jth mix server, for $1 \leq i \leq n$ and $1 \leq j \leq k$, and let $\hat{c}_i^{(j)}$ denote the ith output of the jth mix server. Since the outputs of server j are the inputs to server $j+1$, we have $\hat{c}_i^{(j)} = c_i^{(j+1)}$ for all i and $1 \leq j < k$. Since there is no server $k+1$, the values $\{\hat{c}_i^{(k)}\}$ are the mix-net outputs.

3 Marked Mix-Nets

3.1 Privacy issues with non-verifiable mix-nets

Independently of the integrity properties, the use of the re-encryption mix-net outlined above raises several privacy concerns.

Mixer bypassing. First, when using k mixers in order to avoid the need to trust any particular one, there is no way to be sure that the mixers do not bypass each other during the mixing process. In particular, M_k could ignore the $\hat{c}_*^{(k-1)}$ ciphertexts and use the $c_*^{(1)}$ instead. As a result, M_k alone would be able to choose

the permutation between the $c_*^{(1)}$ and the $\hat{c}_*^{(k)}$ ciphertexts, which is precisely what the use of k mixers is expected to avoid. And there would be no way to detect this manipulation.

One way to avoid this would be to require each mixer to apply, to each ciphertext, a *mark* that shows that it processed this ciphertext, and that can be removed after the end of the mixing process.

Ciphertext replication. At the other end of the chain of mixers, M_1 could also violate the privacy of some voters, by exploiting the homomorphic property of ElGamal. The ciphertexts in $c_*^{(1)}$ can be expected to have a known structure and, in some cases, they could contain elements that cannot be verified easily, or cannot be verified at all, like a random padding used in the message encoding. For instance, in STAR-Vote, each $c_i^{(1)}$ is actually made of two ElGamal ciphertexts, one of them encrypting a hash that can only be matched if the corresponding paper ballot is picked, which is extremely unlikely in a large-scale election. By relying on this, M_1 could target a ciphertext $c_i^{(1)}$ by replacing $c_j^{(1)}$ with a ciphertext $\bar{c}_j^{(1)} = Enc(y, d) \cdot c_i^{(1)}$ where d is a message carefully crafted by M_1 so that $\bar{c}_j^{(1)}$ looks like a perfectly plausible ciphertext, but can still be recognized after decryption of the outputs of M_k by looking for two messages with difference d.

This problem could be avoided by making the ciphertexts somehow non-malleable, so that any duplication or malicious manipulation of a ciphertext would become visible at time of audit. This requirement may seem to be contradictory with the requirement of being able to mark ciphertexts (outlined above), and the reconciliation of these two features is at the core of the design of our marked mix-net.

3.2 A Marked Mix-Net

Our marked mix-net aims to address these privacy issues (and others, which will be discussed later). Still, like the original mix-nets, it does not provide any correctness guarantees: mixers remain able to add or delete ciphertexts during the mixing process. However, contrary to the original mix-nets, these additions and deletions must be independent of the honest mix-net inputs, hence preventing the leakage of information about these inputs.

In order to obtain an extremely fast protocol, we address these issues in a covert adversary model [3], in which attacks can succeed with non neglible probablity, but will also be detected with a non negligible probability. Our assumption here (which is in line with the motivation of the covert adversary model) is that the mixing will be executed by well-defined public parties, and that any cheating detection would immediately trigger police investigation and important penalties, that would be sufficient to deter any such malicious behavior in the first place. Aumann and Lindell discuss a strong version of the covert adversary model in which, when an attack is detected, the adversary does not learn any undue infrormation (so, it is punished on top of having no benefits from

his attack). Variants of our marked mix-net could offer this flavor of security, e.g., by using two layers of encryption, but the resulting protocol would be more expensive.

Non-malleability The resistance to replication attacks suggests the use of an encryption scheme offering some form of non-malleability property that would guarantee that any unauthorized ciphertext manipulation would trigger an alert. However, we still need to be able to re-randomize ciphertexts, in order to be able to perform the mixing operations.

These properties are reminiscent of the notion of re-randomizable RCCA encryption, proposed by Canetti et al. [10], which requires the possibility to re-randomize ciphertexts while preventing any other homomorphic transformation. RCCA security of course does not prevent a mixer to make exact ballot copies (possibly re-randomized), and we will need a mechanism in order to detect these.

One efficient solution for building re-randomizable RCCA encryption has been proposed by Phan and Pointcheval [19], and consists in applying a transformation, called OAEP 3-Round (OAEP3 for short), to messages before encrypting them with a probabilistic encryption scheme like ElGamal.

This transformation assumes the availability of 3 independent random oracles, H_1, H_2 and H_3 (in practice, these can be implemented with a single hash function and 3 distinct prefixes). The OAEP3 transform processes a message m, represented as a bit-string, using a fresh random bit string r as follows:

$$s = m \oplus H_1(r) \quad t = r \oplus H_2(s) \quad u = s \oplus H_3(t)$$

and outputs the pair (t, u).

The OAEP3 transformed message can then be encrypted with ElGamal as usual, and the resulting scheme is shown to be RCCA secure [19], assuming the hardness of the gap Diffie-Hellman problem in G [17], a problem that consists in solving an instance of the computational Diffie-Hellman problem in the presence of a Decisional Diffie-Hellman oracle. The intuition underlying this result is common to the OAEP-style transformations: any change into t or u leads to message changes that are unpredictable without querying the H_i oracles first. So, a modification of a ciphertext can result in a recognizable modification of the corresponding plaintext only if this plaintext is known in the first place.

Still, it does not guarantee that mixers operate on the expected ciphertexts in the first place, without bypassing other mixers. This concern will be addressed below.

Marking The mixer bypassing problem discussed above can be solved by requiring all mixers to add a secret mark on each of the ciphertexts that they process, on top of the OAEP3 transform.

Concretely, each of the k trustees proceeds as follows:

1. The encryption process is modified by requiring each party willing to submit an encryption of a message m to the first mixer to encrypt the message OAEP3$(m\|0^\mu)$ instead, where μ is a security parameter.

2. Each mixer $M_i \in \{M_1, \ldots, M_k\}$ chooses a secret mark $a_i \in \mathbb{Z}_q$, and broadcasts $E(y, \text{OAEP3}(a_i))$. (It may be appropriate to use a key y that is different of the one used to encrypt the messages.)
3. At mixing time, instead of simply re-randomizing each ciphertext by multiplying it with a ciphertext of the form (g^r, y^r), each mixer performs multiplications with ciphertexts of the form $(a_i g^r, y^r)$, hence multiplying the encrypted message by a_i.
4. When the whole mixing procedure is complete, the secret marks are decrypted, and the decrypted outputs of the last mixer are all divided by the product of all a_i's. It is then verified that the 0^μ sequence is present in all the plaintexts, and that the randomness used in OAEP3 is unique. An investigation is triggered otherwise.

The 0^μ string provides a baseline on top of which all marks are applied. If a decrypted ciphertexts fails to have been properly marked, there is a probability $2^{-\mu}$ that, after the removal of all expected marks, it would still end with the 0^μ sequence. So, in our covert adversary setting, a very short μ may be sufficient, e.g., $\mu = 1$ for having a probability around $1/2$ of detecting this kind of malfeasance.

In a similar way, if an adversary tries to perform a copy of a ciphertext, possibly rerandomized or modified in some way, the non-malleability property obtained from OAEP3 will either cause the repetition of the randomness used in the OAEP3 transform, which we require to be unique, or remove the 0^μ string with noticeable probability, leading to an invalid ciphertext.

Note also that we demand the encrypted marks of the initial broadcasts to be OAEP3 processed too, essentially to make sure that the encrypted mark that is broadcast cannot be used by a corrupted mixer to apply another mixer's mark (using the homomorphic property of ElGamal encryption for instance).

Thanks to this marking process, any mixer that would try to bypass other mixers and process ciphertexts that did no go through the expected path will fail to contain the marks that it should. This will also leave evidences through the list of ciphertexts, and the cheating mixer can then be identified. The marking mechanism used here is similar to that used by Chaum in the "dining cryptographer's problem" [13]. Here marks are also members of the underlying abelian group, and can be added or removed from the message in any order.

The resulting marked mix-net is described in Figure 2.

3.3 Security analysis

We discuss how our marked mix-net defeats the traditional attacks on mix-nets, including those discussed in the in-depth review of Adida [2], and leave a rigorous specification and analysis of the security properties as an important and non trivial future work, that should be performed before any deployment.

1. **Related inputs.** In this attack, a corrupted party submits a ciphertext computed as a function of another one, in order to detect a know relation after decryption and de-anonymize targeted ciphertexts. As usual, we require the input ciphertexts to be submitted using a submission-secure scheme [30],

The marked mix-net proceeds as follows:

1. A public key y is made available to all message senders and mixers.
2. Each mixer M_i picks a random mark $a_i \in \mathbb{Z}_q$, and broadcasts $E(y, \text{OAEP3}(a_i))$.
3. In order to submit a message m to the first mixer, submit a ciphertext $E(y, \text{OAEP3}(m \| 0^\mu))$.
4. When all inputs have been submitted, each mixer M_i sequentially shuffles its input ciphertexts, multiplies each of them with a fresh ciphertexts $E(y, a_i)$, then passes the resulting ciphertexts to mixer M_{i+1}.
5. The ciphertexts produced by the last mixer are verifiably decrypted, producing a sequence $d_1, \ldots, d_n$.
6. The encrypted marks are verifiably decrypted, making the a_i's available.
7. The mixed messages $\hat{m}_1, \ldots, \hat{m}_n$ are retrieved by computing $(\hat{m}_i \| 0^\mu \| r_i) = \text{OAEP3}^{-1}(d_i/a)$ where $a = a_1 a_2 \cdots a_k$. If the 0^μ sequence is missing, or if any pair of ciphertexts shares the same r_i then an investigation is triggered in order to identify the cheating party.
8. The inputs and outputs of each mixer are tested to be made of elements of the group G, and the ZK proofs are verified.

Fig. 2. The marked mix-net

which prevents related submissions. Still, if the first mixers are corrupted, they would be able to remove one ciphertext (or more) and replace it with a ciphertext related to the one that is targeted. The relation between the two ciphertexts can be of two types: either a direct (re-encrypted) copy, or an encryption of a modified ciphertext. Copies would cause the presence of messages with identical randomness (r_i) after decryption. Modifications would, thanks to the properties of the OAEP3 transform, result in the decryption of a random message, which would then contain the 0^μ message tag with probability $2^{-\mu}$ and be declared invalid with high probability. Both these attacks would be detected and investigated, in contradiction with our covert adversary security model.

2. **Attacks based on lack of semantic security.** Our ciphertexts remain standard ElGamal ciphertexts, which prevents any leakage of partial information that could be used to track ciphertexts.
3. **Attacks based on partial decryption during mixing.** Our marked mix-net only performs decryption after the completion of the mixing, preventing any coalition of mixers to take advantage of the partial decryption of others.
4. **Mixing cancellation.** We prevent mixers from canceling the permutation applied by other mixers (by mixing their inputs instead of their outputs) thanks to our marking mechanism: skipping any mixer will cause a missing mark, which will be spotted at decryption time.
5. **Proof wrapping.** Some mixes use double encryption layers, which may cause issues when proofs are provided about the outer layer only (e.g., an adversary could add a third encryption layer and make proofs about that

last one). OAEP3 can be interpreted as an extra layer in the encryption mechanism. However, it is designed to prevent malleability, which is the core ingredient used in wrapping attacks.

6. **Subgroup tagging.** A corrupted mixer may lift some ciphertexts to another (possibly larger) group, and use this mechanism to circumvent the guarantees of semantic security and track a ballot throughout the mixing process. In order to prevent such attacks, we require the mix verification process to check that the outputs of each mixer indeed lie in the right group.
7. **Delayed effect.** Here, a corrupted first mixer uses the time between the closing of the polling places and the beginning of the mixing to replace some ciphertexts, possibly with the help of the parties who submitted them. This may make it possible to change or adapt votes after the closing of the votes, based on fresh information. Such a strategy would definitely pass our verification process and is the main reason that makes it non-verifiable in the traditional sense. Whether it matters is application dependent. In an application like STAR-Vote, in which the mixing-process is followed by a risk-limiting audit against the paper ballots that were submitted before the closing of the polling places, the RLA guarantees that this kind of attack will be limited to happen for a very small number of ciphertexts, small enough to make sure that it has no impact on the election outcome. Besides, any change would also create discrepancies with the results of the end-to-end verifiable tally.
8. **Input guessing.** This attack is similar to the previous one but, instead of colluding with a voter, the adversary (corrupting the first mixer, for instance) can try to guess someone's vote (or maybe just verify that a voter followed instructions) and replace the ciphertexts submitted by the targeted voter with fresh ciphertexts encrypting the expected vote intent. As before, this substitution cannot be detected as part of the mixing process. It is also benign in itself, as long as the adversary has no way to detect whether his guess was correct or not. This may not be the case however in an application like STAR-Vote, where the RLA may actually offer evidence of a ballot modification, which would happen as a the result of a wrong guess. But this communication channel, which may inform the adversary about the targeted voter intent, would at the same time offer evidence of malicious behaviour, and this attack strategy would therefore be excluded by our covert adversary model.
9. **Permutation guessing.** In another variation of the previous attacks, an attacker could make a guess on the mapping between an input ciphertext and an output ciphertext. If the guess is correct, then the division of these two ciphertexts would provide an encryption of the mark of that mixer, which could be used to bypass him. However, this attack will be visible in the likely case of an incorrect guess, and is therefore excluded by the covert adversary model.

In terms of accountability, the use of a marked mix-net makes investigations more challenging than in the case of a fully verifiable mix-net: in the fully

verifiable case, it is enough to check all proofs of shuffle and decryption in order to find out who cheated during the mixing process. A marked mix-net does not provide such features anymore. However, we require all mixers to keep track of their secret permutations and reencryption factors until the end of the audit process (e.g., by storing their random seed). These can be used in order to obtain the necessary accountability in case of discrepancy happening during decryption (e.g., the 0^μ sequence is missing) or if the RLA detects a problem. A simple strategy for making mixers accountable would be to ask them to provide their re-encryption factors related to problematic ciphertexts, proceeding by following the mixers in backwards order. This is extremely simple but may raise privacy concerns in some cases. Still, if the penalty of a cheating mixer is high enough, it is sufficient to deter from any temptation. A privacy-friendly solution would be, in case of problem, to ask the mixers to produce a traditional ZK proof of their correct behavior. Again, even if this is not desired due to the computational burden that it would bring, the perspective that this will happen and result in cheater detection can be expected to deter mixers from adopting any malicious behavior. A malicious vote submission device could also submit an invalid ballot as an input to the mix-net, in order to trigger the full proof process and slow down the mixing. Whether this is realistic or not is application dependent: ballots submitted to a mix-net are typically identifiable, and this may be enough to deter anyone from adopting this strategy, which offers fairly low benefits. Besides, when ballots are encoded by a DRE (as in STAR-Vote for instance), an invalid ballot also becomes a sign of a serious hacking in the system, which would trigger deep investigation anyway.

4 STAR-Vote

4.1 STAR-Vote's risk limiting audit

As explained in Section 1.2, our motivation for marked mix-nets comes from the risk limiting audit process in STAR-Vote. We provide more details about the process proposed in the original paper, discuss how it can be adapted to use our marked mix-net, and the benefits that result from it. In order to simplify our discussion, we only focus on the aspects of STAR-Vote that are relevant to the RLA.

STAR-Vote ballots are prepared by ballot marking devices (BMD) all interconnected, inside each voting office, to a controller and an urn. When a voter prepares a ballot, two records are produced:

1. A paper record, which contains (among other things) the voter choices in human readable form, as well as a random, high-entropy, unique ballot id (or *bid*). This *bid* is printed on the paper, but not otherwise known to the voters, election officials, or mix servers.
2. An electronic record, which contains (among other things):
   - An encryption of the voter choices, under the form of a counter set to "0" or "1" encrypted for each option that the voter can select. As a Texas election can contain 100 races, this can make a few hundreds ciphertexts.

– An encryption of $H(bid\|r)$ for each race r to which the voter participates.

So, for each race r included in the ballot, ciphertexts of the form $E(y, H(bid||r)), E(y, v)$ are produced, in which $E(y, v)$ encrypts the content of the vote v for race r.

The paper record is placed into an urn, making it human readable but anonymous, while the electronic record is fully encrypted but cannot be considered to be anonymous (including because various logs can make it quite easy to match the timing at which a ciphertext is produced and the one at which a voter is seen to produce his ballot).

When the polls are closed, the electronic record is used to compute the election tally very fast, by decrypting the homomorphic aggregation of the ciphertexts.

The mix-net permutes the $E(y, H(bid||r)), E(y, v)$ tuples race by race (they are considered as a single big ciphertext from the mix-net point of view), and the outputs are then decrypted, revealing the plaintext items $H(bid||r), v$.

This structure defeats "pattern attacks", also called "Italian attacks" [20], since it isn't obvious from the output which votes are from the same ballot (for different races) as long as the *bid* is not known.

The risk-limiting audit examines randomly selected paper ballots one at a time until enough evidence has been gathered to confirm the nominal (initial, reported) election outcome, or until all paper ballots have been examined. In the latter case, a full recount has been performed by the RLA.

When a paper ballot is selected, its *bid* is read. This allows $H(bid||r)$ to be recomputed for each race, which allows the appropriate entries to be identified in the decrypted mix-net output, together with the corresponding vote v. The voter selections on the paper ballot for each race can then be examined for equality with the value v.

4.2 Using a marked mix-net

STAR-Vote prescribes the use of a verifiable mix-net, for privacy reasons more than for verifiability reasons, since the output of the mix-net is already verified both against the homomorphic tally (at the global level) and through the RLA (at the ballot level). We suggested above that the use of a marked mix-net could provide a more efficient and simpler choice.

The specific structure of the messages mixed in STAR-Vote suggests further tweaks. First we may question the benefits of mixing tuples of ciphertexts instead of single ciphertext. STAR-Vote has distinct ciphertexts in order to be able to reuse the ciphertexts used in the homomorphic tally as part of the inputs of the mix-net, which provides some guarantees of consistency between the inputs of the mixnet and homomorphic tally. Having two ciphertexts per ballot and per race however potentially doubles the computational power that is required to decrypt the outputs of the mix-net, a task that we would like to keep short. The consistency guarantee may also not be critical since, in case of investigation, evidence of the discrepancies can still be collected.

The reuse of the ciphertexts produced for the homomorphic tally as inputs for the mix-net also raises difficulties for our marked mix-net described above, since plaintexts need to be OAEP3 processed before encryption. We therefore suggest to modify the STAR-Vote design to have one single ciphertext per ballot and per race, which can then be conveniently processed through the OAEP3 transform, as specified in Figure 2.

The use of the OAEP3 transform does not raise any difficulty when the message space of the encryption scheme is large enough (for instance, when G is the subgroup of quadratic residues modulo a large p). However, the use of elliptic curve might raise some difficulties. Indeed, while the OAEP3 expansion is quite low (it does not require any strong redundancy, like OAEP for instance), there is still a length increase that comes from the randomness that is added, and which may look unnecessary in the context of an encryption scheme like ElGamal, which is already randomized. This randomness is however useful for at least two reasons: in general, it guarantees the non-malleability, by preventing an adversary to compute a table of all possible OAEP3 outputs in case of small message space and, in the context of a mixnet, it prevents the invisible inclusion in the mixing process of ciphertext copies.

We may however observe that the messages that we need to encrypt all contain a single unique component that is indistinguishable from a sequence of random bits: $H(bid\|r)$. We therefore suggest that the inputs of the mixnet could be computed as: $E(y, \text{OAEP3}(v\|0^\mu; H(bid\|r)))$, where the message part of the inputs of OAEP3 is $v\|0^\mu$ and $H(bid\|r)$ is used as fresh randomness. It can be observed, as did Abe, Kiltz and Okamoto for instance [1], that the OAEP3 transformation does not guarantee non-malleability with respect to the random part of the OAEP3 inputs (such non-malleability is provided by their OAEP-4X transformation). However, we really need the non-malleability with respect to the 0^μ part, which is sufficient to detect malicious behaviors. As a result, the message expansion resulting from the input pre-processing of the marked mix-net is only of μ bits, where μ can be just a small constant.

The modifications that we propose for the STAR-Vote RLA are summarized in Figure 3.

1. The ballot marking device produces a submission secure encryption $E(y, \text{OAEP3}(v\|0^\mu; H(bid\|r)))$ instead of the encryption $E(y, H(bid\|r))$.
2. The marked mix-net of Figure 2 is used to process these ciphertexts (except for Step 3, which is performed as above), instead of a verifiable mix-net.

Fig. 3. Proposed changes in the STAR-Vote RLA.

4.3 Benefits of the approach

The proposed approach considerably simplifies the risk-limiting process from an algorithmic point of view: it avoids the need to run a full verifiable mixnet for the sole purpose of the risk-limiting audit.

It also considerably decreases the computational power that is required from the mixers: their task can now be almost entirely precomputed and, in particular, no online modular exponentiation is required. This may enable to close the election audit process faster, reduce the computing infrastructure costs, and also reduce the organizational burden.

The latency before the beginning of the paper comparison phase of the RLA may not change, however: if a covert adversary setting is accepted, the mixers of the verifiable mix-net may simply shuffle the ciphertexts and pass them along, which would cost as much as our marked mix-net, and only start computing their proofs of shuffle after that, knowing that they will eventually need to produce them for audit. Note that, in a traditional adversary setting, proofs would need to be provided and verified before the beginning of the decryption phase, and the strong latency would be back.

5 Conclusions

We have described a new type of mix-net that offers weaker verifiability properties than a traditional verifiable mix-net, yet preserves privacy even when all public keys are known and when all but one mix server may act maliciously.

Our marked mix-net is considerably more efficient than a fully verifiable mix-net in terms of computational load: the online working load of each mixer is only 2 multiplications per ElGamal ciphertext, while recent proof of shuffle (e.g., [26,4]) require several online exponentiations per ciphertext. We can then expect to decrease the computational load by a factor at least 100. We believe that this is an appealing feature in large scale elections.

We suggest that STAR-Vote, and possibly other voting schemes, need not use a verifiable mix-net and may use the marked mix-net presented here instead. However, before any deployment, a rigorous analysis of the security properties of the marked mix-net would be needed.

Acknowledgement

We thank the anonymous reviewers for their helpful comments and suggestions.

The first author is grateful to the Belgian Fund for Scientific Research (F.R.S.-FNRS) for its financial support provided through the the SeVoTe project. The second author gratefully acknowledges support for his work on this project received from the Center for Science of Information (CSoI), an NSF Science and Technology Center, under grant agreement CCF-0939370, and from the Department of Statistics, University of California, Berkeley, which hosted his sabbatical visit during this work.

References

1. Masayuki Abe, Eike Kiltz, and Tatsuaki Okamoto. Chosen ciphertext security with optimal ciphertext overhead. In *Advances in Cryptology - ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 355–371. Springer, 2008.
2. Ben Adida. *Advances in Cryptographic Voting Systems*. PhD thesis, MIT, 2006.
3. Yonatan Aumann and Yehuda Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. In *Theory of Cryptography TCC*, volume 4392 of *LNCS*, pages 137–156. Springer, 2007.
4. Stephanie Bayer and Jens Groth. Efficient zero-knowledge argument for correctness of a shuffle. In David Pointcheval and Thomas Johansson, editors, *Proc. EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 263–280. Springer, 2012.
5. Susan Bell, Josh Benaloh, Michael D. Byrne, Dana DeBeauvoir, Bryce Eakin, Gail Fisher, Philip Kortum, Neal McBurnett, Julian Montoya, Michelle Parker, Olivier Pereira, Philip B. Stark, Dan S. Wallach, and Michael Winn. STAR-vote: A secure, transparent, auditable, and reliable voting system. *USENIX Journal of Election Technology and Systems (JETS)*, 1(1), 8 2013.
6. Jonathan Ben-Nun, Niko Fahri, Morgan Llewellyn, Ben Riva, Alon Rosen, Amnon Ta-Shma, and Douglas Wikström. A new implementation of a dual (paper and cryptographic) voting system. In *E-VOTE*, 2012.
7. J. Benaloh, D. Jones, E. L. Lazarus, M. Lindeman, and P. B. Stark. Soba: Secrecy-preserving observable ballot-level audit. In *EVT-WOTE 2011*. USENIX, 2011.
8. Dan Boneh. The decision Diffie-Hellman problem. In *Proceedings of the Third Algorithmic Number Theory Symposium*, volume 1423 of *LNCS*, pages 48–63. Springer-Verlag, 1998.
9. Philippe Bulens, Damien Giry, and Olivier Pereira. Running mixnet-based elections with helios. In H. Shacham and V. Teague, editors, *Electronic Voting Technology Workshop/Workshop on Trustworthy Elections*. Usenix, 2011.
10. Ran Canetti, Hugo Krawczyk, and Jesper B. Nielsen. Relaxing chosen-ciphertext security. In *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *LNCS*, pages 565–582. Springer, 2003.
11. Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Verifiable elections that scale for free. In *Public-Key Cryptography – PKC 2013*, volume 7778 of *LNCS*, pages 479–496. Springer, 2013.
12. David Chaum. Untracable electronic mail, return addresses, and digital pseudonyms. *Comm. ACM*, 24(2):84–90, February 1981.
13. David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *J. Cryptology*, 1(1):65–75, 1988.
14. Chris Culnane, Peter Y. A. Ryan, Steve Schneider, and Vanessa Teague. vvote: A verifiable voting system. *ACM Trans. Inf. Syst. Secur.*, 18(1):3:1–3:30, June 2015.
15. Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inform. Theory*, IT-31(4):469–472, July 1985.
16. David Lundin and Peter Y. A. Ryan. Human readable paper verification of prêt à voter. In *Computer Security - ESORICS 2008*, volume 5283 of *LNCS*, pages 379–395. Springer, 2008.
17. Tatsuaki Okamoto and David Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. In *Public Key Cryptography, PKC 2001*, volume 1992 of *LNCS*, pages 104–118. Springer, 2001.
18. Torben P. Pedersen. A threshold cryptosystem without a trusted party (extended abstract). In Donald W. Davies, editor, *Advances in Cryptology - EUROCRYPT '91*, volume 547 of *LNCS*, pages 522–526. Springer, 1991.

19. Duong Hieu Phan and David Pointcheval. OAEP 3-Round - a generic and secure asymmetric encryption padding. In *Advances in Cryptology - Proceedings of ASIACRYPT '04*, number 3329 in LNCS, pages 63–78. Springer, 2004.
20. Stefan Popoveniuc and Jonathan Stanton. Undervote and pattern voting: Vulnerability and a mitigation technique. In *In Preproceedings of the 2007 IAVoSS Workshop on Trustworthy Elections (WOTE 2007)*, 2007.
21. Jian Ren and Jie Wu. Survey on anonymous communications in computer networks. *Comput. Commun.*, 33(4):420–431, March 2010.
22. Peter Y. A. Ryan, Peter B. Rønne, and Vincenzo Iovino. Selene: Voting with transparent verifiability and coercion-mitigation. In *Financial Cryptography and Data Security Workshops - VOTING 2016*, volume 9604 of *LNCS*, pages 176–192. Springer, 2016.
23. Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme. In *Advances in Cryptology–EUROCRYPT'95*, pages 393–403. Springer, 1995.
24. Krishna Sampigethaya and Radha Poovendran. A survey on mix networks and their secure applications. *Proc. IEEE*, 94(12):2142–2181, Dec 2006.
25. Adi Shamir. How to share a secret. *CACM*, 22(11):612–613, November 1979.
26. Björn Terelius and Douglas Wikström. Proofs of restricted shuffles. In Daniel J. Bernstein and Tanja Lange, editors, *Progress in Cryptology - AFRICACRYPT 2010*, volume 6055 of *LNCS*, pages 100–113. Springer, 2010.
27. Georgios Tsoukalas, Kostas Papadimitriou, Panos Louridas, and Panayiotis Tsanakas. From Helios to Zeus. *The USENIX Journal of Election Technology and Systems*, 1(1):1–17, 2013.
28. Verificatum. http://www.verificatum.org/, 2015.
29. Verificatum. Complexity analysis of the verificatum mix-net vmn version 3.0.2. http://www.verificatum.com/files/complexity-3.0.2.pdf, July 2016.
30. Douglas Wikström. Simplified submission of inputs to protocols. In *Security and Cryptography for Networks, 6th International Conference*, volume 5229 of *LNCS*, pages 293–308. Springer, 2008.
31. Douglas Wikström. Electronic election schemes and mix-nets. http://www.csc.kth.se/~dog/research/, 2015.