Improving and Simplifying a Variant of Prêt à Voter*

Ralf Küsters, Tomasz Truderung, and Andreas Vogt

University of Trier, Germany
{kuesters,truderun,vogt}@uni-trier.de

Abstract. Recently, Xia et al. proposed a variant of Prêt à Voter which enjoys several attractive properties. Their protocol is among the few verifiable and receipt-free paper-based voting protocols resistant against randomization attacks. Trust is distributed among several authorities and the voter interface is relatively simple. Also, approval and ranked elections are supported.
In this paper, we improve and simplify the protocol by Xia et al. Among others, we propose a simpler way of producing ballots, which only involves the encryption and re-encryption of candidate names; homomorphic encryption and proxy re-encryption are not needed. Also, no machine involved in the production of ballots needs to store a secret key. Moreover, unlike the protocol by Xia et al., in our protocol all authorities can be held accountable in case they misbehave in an observable way.

1 Introduction

In the last few years many paper-based voting protocols have been proposed that are designed to achieve (various forms of) verifiability [10] and receipt-freeness/coercion resistance [4], with protocols by Chaum [7], Neff [18], and Prêt à Voter [22, 9, 24, 23, 16] being the first such protocols; other protocols include Scratch&Vote [3], PunchScan [6, 19], ThreeBallot, VAV, and Twin [21], Split Ballot [17], BingoVoting [5], a protocol by Riva and Ta-Shma [20], and Scantegrity II [8]. Intuitively, verifiability means that the voter is assured that her vote is actually counted as cast. A voting protocol is coercion resistant if it prevents voter coercion and vote buying. In other words, a coercer should not be able to influence the behavior of a voter.

However, only very few of the paper-based protocols proposed so far are resistant against a specific kind of coercion, namely so-called randomization attacks [13] (see, e.g., [21, 8, 26]). In such an attack the adversary forces the voter to vote in a random way. For example, the voter could be forced to always mark the ballot at a specific position and the coercer may be able to check whether

* This work was partially supported by the *Deutsche Forschungsgemeinschaft* (DFG) under Grant KU 1434/5-1 and 1434/4-2, and the Polish Ministry of Science and Education under Grant 3 T11C 042 30. The second author is on leave from University of Wrocław, Poland.

the voter marked the ballot as instructed by looking at the voter's receipt. The candidate corresponding to such a mark may change from ballot to ballot. So, the vote may be "random". But this is still a serious manipulation of the election as this attack could be carried out on a large scale and the coercer can check after the election whether or not voters followed his instructions.

The recently proposed protocol by Xia et al. [26], which is a variant of Prêt à Voter that resists randomization attacks, is particularly interesting as it distributes trust among several authorities while at the same time allows for a relatively simple voter interface and supports several electoral systems.

Contribution of this Paper. In this paper, we improve and simplify the protocol by Xia et al. In a nutshell, our protocol works as follows. Ballots are produced by a sequence of printers. This process merely involves the encryption and re-encryption of candidate names under the joint public key of tallying tellers. Ciphertexts printed on the ballot have to be covered by scratch strips. The final ballot has only one covered ciphertext per candidate, plus the printed name of the candidate. The voter interface is very simple. To vote, a voter enters a voting booth, chooses one or more candidates and marks or ranks them, by putting crosses or numbers next to the candidate names, and separates the left and the right-hand sides of the ballots. This does not involve a machine. Scanning of ballots and printing receipts can then be done outside of the voting booth in public, with the assistance of the clerks, if necessary. Tallying the ballots is done by the tallying tellers using a mixnet and distributed decryption. The main features of our protocol are the following:

1. Distributed trust. In all stages of the election, trust is distributed among several authorities.
2. Simple production of ballots. The production of ballots involves only basic cryptographic tasks, namely public key encryption and re-encryption. The machines (printers) involved in this process are rather simple and do not have to store secret keys. This reduces failures and security leaks.
3. Simple voter interface. The voter interface does not involve any complex task. Tasks such as uncovering scratch strips, scanning ballots, and printing receipts are done, under the assistance of clerks, using quite simple machines. This, again, increases the reliability and usability of the protocol. Just as the protocol by Xia et al., our scheme is best suited for elections with a small number of candidates.
4. Supporting several electoral systems. Several electoral systems, including approval and ranked elections, are supported.
5. Coercion resistance. Coercion resistance, including resistance against randomization attacks, kleptographic attacks, and chain voting, is guaranteed under weak assumptions: We only require one honest member in every group of authorities. The voting terminals for scanning ballots/printing receipts and posting them on the bulletin board may be dishonest. Dishonest parties may cooperate with the coercer. In fact, we identify them with the coercer. Also, the voter may freely communicate with the coercer during the whole voting

process, even in the voting booth. We only assume that the voter may lie about what she sees and does. (So, no pictures or videos may be taken by the voter. But talking on the phone would not be a problem.)

6. Verifiability. Verifiability can be ensured (with high probability), by the proofs authorities have to provide and the audits that are performed by voters, clerks, and auditors.
7. Accountability. Our protocol guarantees accountability of authorities, i.e., single authorities can be held accountable for their misbehavior.

The protocol by Xia et al. enjoys some of the above features. However, there are crucial differences. First, the production of ballots in the protocol by Xia et al. is much more complex, and hence, less reliable and harder to implement: Besides performing re-encryptions, it also needs homomorphic encryption and proxy re-encryption. Moreover, machines are required to decrypt ciphertexts obtained by proxy re-encryption. In particular, these machines need to store secret keys.

In the work by Xia et al., accountability has not been considered. However, it is clear that their protocol does not achieve the same level of accountability as ours. For example, if in their protocol the audit in the production of ballots reveals that the re-encryption has not been performed correctly, then this cannot be traced back to specific participants, only to a group of participants. For this reason, some participants may have a higher tendency to misbehave, and hence, spoil the outcome of the election.

Unlike in this paper, Xia et al. do not provide a security analysis of their protocol. It seems that arguments for verifiability and coercion resistant similar to the ones presented here carry over to the protocol by Xia et al. However, the exact security guarantees are not clear. For example, while our protocol has an explicit mechanism to prevent chain voting attacks, this is not the case in the protocol by Xia et al., although, presumably, such a mechanism could be added.

2 Our Protocol

In this section, we describe our protocol. It consists of four stages: initialization, preparation of ballots, voting phase, and tallying phase. For simplicity of presentation, we first assume that a voter votes for exactly one candidate. Choosing more or no candidate would result in an invalid ballot. We will see in Section 2.8 that this assumption is not necessary. In fact, as already mentioned in the introduction, our protocol supports a wide range of electoral systems.

Before going into the details of the protocol, we provide a brief description of how the election looks like from the voter's point of view.

2.1 Voting from the Voter's Point of View

When a voter enters the polling place, clerks provide her with what we call a *multi-ballot*. A multi-ballot contains exactly one *simple ballot* for each of the possible candidates. An example of a multi-ballot for an election with two candidates is depicted in Figure 1.

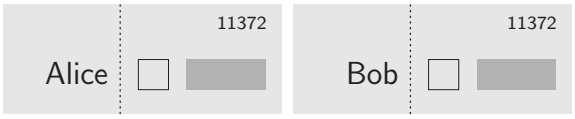

Fig. 1. A multi-ballot consisting of two simple ballots for a two candidate election.

A simple ballot consists of two parts which can be easily separated. On the left-hand side of a simple ballot a candidate name is printed. This side does not contain any other information. On the right-hand side a serial number and a box is printed, which the voter can mark. There is also a scratch strip which covers some information, namely the encryption of the candidate name, as explained later. Every simple ballot in one multi-ballot has printed on it the same serial number.

Once the voter is provided with a multi-ballot, she enters the voting booth. She marks the box on exactly one of the simple ballots and for *every* simple ballot, including the ones not marked, she separates the left-hand side from the right-hand side and discards the left-hand sides. Note that up to this point, the voter does not have to use any device or machine, which makes this process less error-prone and relatively easy to perform. This is a big advantage in practice.

At this point the voter steps out of the voting booth with the right-hand sides of all simple ballots in her hand. The rest of the process is done in public, under the eyes of and possibly with the help of the clerks. Basically, the scratch strip is removed (by hand or using a machine) and all (right-hand sides of the) simple ballots are scanned and posted on the bulletin board. The voter gets a copy of all of these ballots as her receipt.

2.2 Cryptographic Primitives

We start the detailed description of our protocol with a brief introduction of the cryptographic primitives that we use.

We will use an encryption scheme that allows for random re-encryption and distributed decryption (see, e.g., [11, 25, 1]). In such a scheme, a group of agents can collectively generate a public key K which can be used to encrypt messages. To decrypt a ciphertext, the participation of all parties involved in generating K is necessary and the parties are required to provide proofs of compliance.

We will also use a universally verifiable re-encryption mixnet. A re-encryption mixnet consists of a set of mix servers $\mathsf{T}_1, \ldots, \mathsf{T}_m$, where T_1 gets as input an ordered set $\{c_1, \ldots, c_r\}$ of messages encrypted under the public key K and then successively each of $\mathsf{T}_1, \ldots, \mathsf{T}_m$ in turn applies a random re-encryption to every ciphertext c_i. These re-encryptions are then forwarded in a random order to the next mix server. The output of the mixnet is the output of T_m. This output is a permutation of re-encryptions of $c_1, \ldots, c_r$, if every mix server behaved correctly. As long as the mixnet contains at least one honest server, it should be infeasible to trace a message from the input to the output. In a universally verifiable

mixnet the mix servers publish additional information so that for any observer it is possible to ensure that the servers behaved correctly (see, e.g., [12, 2]).

2.3 Participants

Beside the voters, the following principals/machines participate in the protocol. Their tasks will be explained in more detail later on.

1. *Bulletin board* BB: This is a kind of write-only, publicly accessible memory. We assume that every message posted by a principal on the bulletin board is signed by that principal.
2. *Auditors* $\mathsf{A}_1, \ldots, \mathsf{A}_{m_a}$: They perform several kinds of audits.
3. *Printers* $\mathsf{P}_1, \ldots, \mathsf{P}_{m_p}$: These machines print the ballots. They may be run by different institutions at different locations.
4. *Bundlers* $\mathsf{B}_1, \ldots, \mathsf{B}_{m_b}$: They perform the final step in compiling multi-ballots.
5. *Clerks* $\mathsf{C}_1, \ldots, \mathsf{C}_{m_c}$: They conduct the voting process, including issuing multi-ballots to voters and ensuring the correctness of the procedure.
6. *Voting terminals* VT: These machines scan ballots cast by the voters, make copies (receipts) for the voters, and, after the voting phase is finished, post the scanned ballots on the bulletin board.
7. *Tellers* $\mathsf{T}_1, \ldots, \mathsf{T}_{m_t}$: They tally the ballots in a way specified later.

2.4 Initialization

In the initialization phase, all participants generate their private and public keys, as far as necessary, and put their public keys on the bulletin board, possibly along with a proof that they know the corresponding private key and signatures by some certification authority.

As mentioned in Section 2.2, the tellers $\mathsf{T}_1, \ldots, \mathsf{T}_{m_t}$ will perform distributed decryption in the tallying phase. Their joint public key, which we denote by K, can be computed publicly from the public keys every teller put on the bulletin board.

Finally, the set of allowed serial numbers is posted on the bulletin board as well as the list of candidates. This can be done by some election supervisors, which we do not specify in detail here.

2.5 Preparation of Ballots

We now describe how multi-ballots are produced. This is done in two steps. First, ballots are printed by the printers $\mathsf{P}_1, \ldots, \mathsf{P}_{m_p}$. Then, bundlers $\mathsf{B}_1, \ldots, \mathsf{B}_{m_b}$ do the final compilation. The whole process is audited by the auditors $\mathsf{A}_1, \ldots, \mathsf{A}_{m_a}$.

Printing of Multi-Ballots. As already mentioned in Section 2.1, a multi-ballot consists of a set of simple ballots, one simple ballot per candidate. Each multi-ballot has a unique serial number. This number is printed on every simple ballot within a multi-ballot.

Initially, the sheets of paper on which simple ballots are printed are a bit larger than in the final ballots in order to accommodate additional information needed for the production of the ballots. However, the additional parts of the (extended) simple ballots are cut off in the last step of the ballot preparation.

Each multi-ballot is prepared by the printers $\mathsf{P}_1, \ldots, \mathsf{P}_{m_p}$, which iteratively process the multi-ballots. Typically different printers would belong to different institutions in different locations in order to distribute trust. Let us look at the production of one multi-ballot with the serial number sid.

First, P_1 prints, for each candidate name name, an (extended) simple ballot which contains the serial number sid, the candidate name name, and an encryption s_1 of the candidate name name under the joint public key K of the tellers, as depicted in Figure 2, (a); the randomness used for the encryption is chosen freshly for every encryption. Then, P_1 covers the candidate name with a scratch strip (see Fig. 2, (b)).

In this way, P_1 prepares an (extended) simple ballot for each candidate. Together they form a multi-ballot. Before giving this multi-ballot to the next printer P_2, P_1 shuffles the simple ballots within the multi-ballots. In praxis, P_1 would of course not only send one but the set of all multi-ballots prepared by P_1 to P_2.

Now, when P_2 receives a multi-ballot from P_1, P_2 computes re-encryptions of the ciphertexts printed on the simple ballots contained in the multi-ballots, everytime using fresh randomness. For each simple ballot, the corresponding re-encryption is then printed next to the previous ciphertext. In addition, the old ciphertext is covered by a scratch strip. Figure 2, (c) shows a simple ballot after the re-encryption has been printed, where s_1 is the old ciphertext and s_2 is its re-encryption. Figure 2, (d) shows this simple ballot after the old ciphertext s_1 has been covered.

As in the case of P_1, P_2 shuffles the simple ballots of a multi-ballot before giving the multi-ballot to the next printer P_3. Then, P_3 processes the multi-ballot in the same way, and so on.

The last printer P_{m_p} obtains multi-ballots containing simple ballots of the form depicted in Figure 2, (e). It re-encrypts s_{m_p-1}, obtaining s_{m_p}, prints s_{m_p} next to s_{m_p-1} (see Fig. 2, (f)), and then covers both s_{m_p-1} and s_{m_p}, resulting in a simple ballot as depicted in Figure 2, (g).

We assume that independent auditors $\mathsf{A}_1, \ldots, \mathsf{A}_{m_a}$ make sure that in the process of printing ballots, scratch strips on ballots are never removed. They also make sure that ballots do not get lost or are replaced. However, the auditors do not get to see how exactly the ballots are shuffled. Of course, printers also keep the randomness they use for the encryption (in case of P_1) and re-encryption (in case of $\mathsf{P}_2, \ldots, \mathsf{P}_{m_p}$) secret.

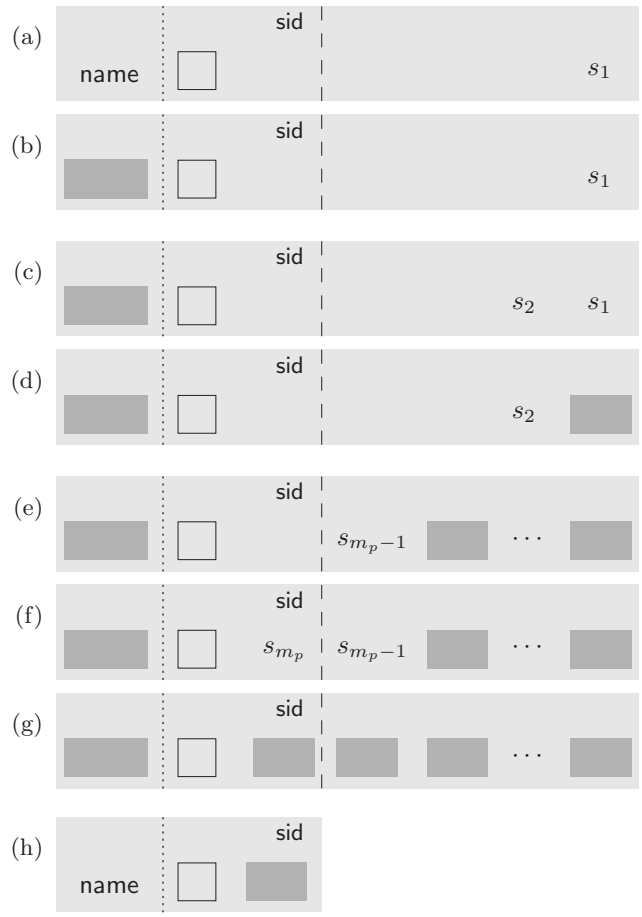

Fig. 2. Preparing ballots.

In case a simple ballot in a multi-ballot is tampered with, this ballot is destroyed and the serial number of this ballot is marked invalid. This fact would then be reported by the auditors on the bulletin board. Potentially, a printer accountable for this tampering could be excluded from the group of printers.

Auditing of Printed Ballots. Beside this "physical auditing" of the printing process, auditors $\mathsf{A}_1, \ldots, \mathsf{A}_{m_a}$ (or a group of auditors different to the group mentioned above), check, after the printing process is finished, whether the information printed on the ballots is as specified.

For this purpose, $\mathsf{A}_1, \ldots, \mathsf{A}_{m_a}$ jointly and randomly (every auditor contributes her own randomness) pick a fraction of multi-ballots from the set of all printed multi-ballots.

The auditors remove all scratch strips on these ballots and ask the printers to reveal the randomness that they have used to perform the encryption (in case of P_1) and the re-encryption (in case of $\mathsf{P}_2, \ldots, \mathsf{P}_{m_p}$) in the production of these ballots.

The auditors can then check (in public) whether these ballots have been produced as specified. Of course, they cannot see whether the randomness that the printers used is really random. However, as we will see in Section 3, as long as one printer is honest, and hence, uses real randomness, this is no problem. What the auditor *can* see, though, is whether the first encryption s_1 in Figure 2 in fact encrypts the candidate name on the left-hand side of the ballot and whether the rest of the ciphertexts is a re-encryption of s_1, and hence, s_{m_p} is an encryption of the correct candidate name.

All ballots audited by the auditors are destroyed and their serial numbers are marked invalid on the bulletin board.

Final Preparation of Multi-Ballots. The remaining multi-ballots are then shuffled, in turns, by all the bundlers $\mathsf{B}_1, \ldots, \mathsf{B}_{m_b}$. Then the right-hand sides of the ballots containing the covered ciphertexts $s_{m-1}, \ldots, s_1$ are cut off, say by B_{m_b}. This bundler also removes the scratch strip from the left-hand sides of the ballots, uncovering candidate names. The resulting ballots are depicted in Figure 2, (h). Each multi-ballot is then put into an envelope. The parts cut off from the ballots are destroyed.

Auditors make sure, by physical inspection, that all steps are performed correctly. No multi-ballot should get lost or be replaced and all scratch strips, including those on the eventually destroyed right-hand sides of the ballots should not have been tampered with.

2.6 Voting Phase

The voting phase, which takes place in polling places, consists of the following steps, which are carried out by the voters, the clerks, and the voting terminals, in conjunction with the bulletin board.

V1. Register and obtain multi-ballot. The voter first registers at the polling place. The clerks then provide the voter with a multi-ballot, making sure that the serial number is valid. The voter and the clerks also check whether the multi-ballot has not been tampered with and that there is exactly one simple ballot for every candidate. Otherwise, the voter gets a new multi-ballot and the old one is destroyed and marked invalid on the bulletin board. (The incident is reported and investigated.)
Clerks also record the serial number of the multi-ballot issued to a voter. (Later they make sure that the voter casts a ballot with the same serial number.)

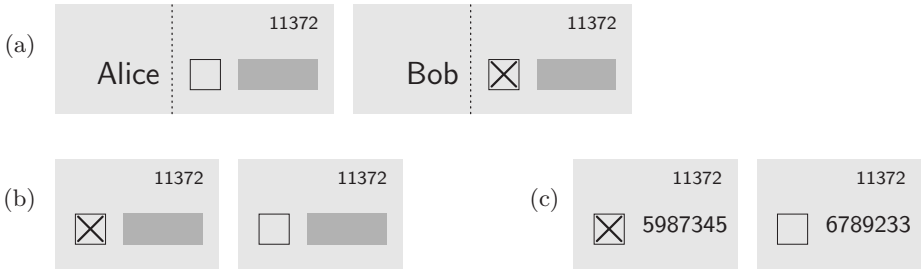

Fig. 3. An example of a multi ballot consisting of two simple ballots for a two candidate election: (a) after step V2, (b) after step V3, (c) after step V5. Note that a copy of (c) is given to the voter as a receipt. The number "5987345" is supposed to be a randomized encryption of "Bob".

V2. Entering the voting booth and choosing a candidate. The voter enters a voting booth and marks the box of exactly one candidate. All other boxes are left blank. In an election with two candidates, the multi-ballot would now look like the one depicted in Figure 3, (a).

V3. Separating left-hand and right-hand sides. The voter then separates the two parts of every simple ballot, not just the one that she marked (this can be done manually or using a simple cutter). She keeps only the right-hand sides. The left-hand sides can be thrown away. The right-hand sides should be shuffled. The result of this step is depicted in Figure 3, (b).

Now, the voter leaves the voting booth. We emphasize that up to this point the voter did not have to use any machine or device. All steps performed by the voter thus far were relatively basic and simple, and hence, less error-prone and less susceptible to manipulation.

The rest of the voting process takes place in public.

V4. Auditing by clerks. The clerks check that the voter in fact has separated the left- and the right-hand sides of all simple ballots and that the scratch strips are untouched. The clerks also make sure that the ballots are separated in such a way that there are no visible signs that would allow an observer to match corresponding right- and left-hand sides. Otherwise, the ballots are destroyed, the serial number is marked invalid, and the voter may start the voting process all over again (beginning with step V1) or leave the polling place without voting.

V5. Removing scratch strips. The voter, possibly with the assistance of the clerks or some machine, may now remove the scratch strips from all (right-hand sides) of the simple ballots in her multi-ballot, resulting in a multi-ballot (with only right-hand sides) of the form depicted in Figure 3, (c).

V6. Casting the ballot and obtaining a receipt. The voter, in presence of the clerks, now casts her ballots, i.e. she inserts all the (right-hand sides of) the simple ballots into the voting terminal VT. The clerks make sure that the serial number of these ballots is the same as the one recorded in step V1.

The terminal scans the ballots and checks whether the markings made by the voter are correct, i.e. comply with the election system. If the markings are correct, the terminal provides the voter with a copy of the ballots. Otherwise, the voter is informed that the marking was invalid, without being given a receipt.
As mentioned at the beginning of Section 2, for now we assume that a voter marks exactly one simple ballot. So the terminal would only accept the multi-ballots with exactly one marked box. In this particular case, a voter could cast only the marked simple ballot.

2.7 Tallying Phase

After the voting phase is finished, the voting terminals VT post all the scanned multi-ballots with valid markings on the bulletin board. At this point or later, a voter can check, whether all her ballots (i.e. all the ballots she has receipts of) appear on the bulletin board. Also, at each pooling place, the clerks check whether the ballots posted on the bulletin board correspond to the actual (physical) ballots kept by the voting terminals and those issued to the voters. This could be done manually or with the help of another, independent machine.

Then, the tellers $\mathsf{T}_1, \ldots, \mathsf{T}_{m_t}$ shuffle and decrypt the set of published ballots, as described below. Recall that each of the published ballots contains a serial number, a ciphertext, and possibly a marking, where the markings of multi-ballots comply with the electoral system used.

Mixing. From the published ballots, the serial numbers are discarded. The ballots are then grouped according to their markings. In our simple setting, we have two groups, one group of ballots with markings and one without. If the marks were numbers, the ballots would be grouped according to these numbers. Every group of ballots is now put through a universally verifiable re-encryption mixnet (see Section 2.2), where the tellers $\mathsf{T}_1, \ldots, \mathsf{T}_m$ serve as mix servers. (In our simple setting, ballots without marks can be discarded. They do not need to be shuffled or decrypted.) The result of the whole mixing procedure is a list of shuffled and re-encrypted ballots posted by the last teller, one list for each group.

Decrypting. Finally, the tellers collectively decrypt the shuffled and re-encrypted ballots within each group of ballots. The results as well as proofs of compliance of the decryption are posted on the bulletin board.

2.8 Supporting Other Electoral Systems

Up to now, we have assumed, for simplicity of presentation, that every voter marks exactly one of the simple ballots within a multi-ballot. However, it is straightforward to support many other electoral systems, where voters may vote for many candidates or even rank candidates by assigning numbers.

We only need to assume that the pattern of marks (crosses or numbers) chosen by a voter does not reveal any information, e.g., to the clerks in the voting phase, about how a voter voted. If, for example, voters may make either one or two marks, then a clerk can see how many marks a voter made. Therefore, for our protocol to be coercion resistant, we assume that the electoral system is such that only one pattern of markings is allowed.

In case the electoral system allows for different marking patterns one can simply add to a multi-ballot enough of what we call dummy simple ballots, which do not belong to any candidate. For example, if the electoral system allows to vote for zero, one, or two candidates, a multi-ballot could contain two dummy simple ballots and every voter is required to mark exactly two simple ballots. To vote for one candidate, for example, a voter would mark her candidate and one dummy simple ballot.

We assume also that for tallying simple ballots the context of the multi-ballot it belonged to is not relevant. For some electoral systems that might not be true. In this case, the mixing and decryption of ballots should keep simple ballots grouped in multi-ballots. This could for example be done by leaving the serial numbers on the ballots, but encrypting and re-encrypting them in the mixing phase. However, we omit the details.

3 Security Analysis

In this section, we argue that our protocols enjoys verifiability, accountability, and coercion resistance.

3.1 Accountability

We say that a protocol is *accountable* if the following is true: If there is an observable deviation from the protocol (i.e., a participant does not send messages as expected, proofs of compliance are invalid, or an audit step fails), then this deviation can be traced back to every single party who misbehaved, not just an anonymous group of parties. This property is important. Without it there would be a higher tendency to misbehave, and hence, a higher probability for the result of the election to be spoiled. Accountability justifies the assumption that authorities, even if dishonest, do not misbehave in an observable way, if this is likely to be noticed by (honest) auditors/clerks or external observers who can check proofs of compliance of authorities.

To argue about accountability we assume the following:

A1. The bulletin board is honest and there is at least one honest participant in the group of auditors and clerks, respectively.

Accountability of auditors and clerks follows from the assumption that they watch each other and that there is at least an honest member in every group. Misbehavior of bundlers and voting terminals is observed by the checks that clerks and auditors do. Individual tellers can be held accountable due to the proof of

compliance each teller has to provide. Misbehavior of printers is detected by auditors. For example, auditors make sure, at least for a fraction of the ballots, that printers can provide the randomness they have used for the encryption or re-encryption of candidate names. Since the encryption and re-encryption performed by every printer is recorded on the (extended) simple ballot, misbehavior can be traced back to every single printer. We note that in the protocol by Xia et al. [26], this is not the case.

3.2 Verifiability

We now show that our scheme is *verifiable* in the following sense. First, a voter, under some weak assumptions (see below), can make sure that her vote is included in the final tally as intended (*individual verifiability*). Second, under slightly stronger assumptions, it can be made sure that the outcome of the election corresponds to the intended votes of all legitimate voters; we call this property *complete verifiability*. Complete verifiability involves that (1) only legitimate voters can cast ballots, (2) these ballots are tallied as expected by the tellers, and (3) no other ballots are tallied.

The assumption we need to obtain individual verifiability is the following:

S1. The multi-ballots checked by the auditors in the ballot production phase are chosen randomly and ballots are not altered or replaced before being issued to the voters.

Let us emphasize that this assumption is rather weak. It is, for example, implied by the assumption that there is one honest member in the group of auditors and clerks, respectively.

By assumption S1, the voter can be almost sure, that her ballot is formed correctly. (If t ballots are not formed correctly and 10% of the ballots are audited, then the probability that none of the ill-formed ballots is detected is $\leq 0.9^t$.) Now, universal verifiability of the mixnet plus the proofs provided by the tellers in the decryption phase guarantee that all ballots on the bulletin board are decrypted correctly. So, with the voter's receipts, individual verifiability follows.

To avoid the above assumptions, one could change the protocol in such a way that the voters themselves check whether multi-ballots are formed correctly: They are given two multi-ballots, say, possibly with still extended simple ballots, and randomly choose one of the two multi-ballots to be audited in a publicly verifiable way.

To obtain complete verifiability of our protocol, we, in addition to S1, assume:

S2. There is at least one honest clerk at each polling place.

From this assumption, (1) and (3) from above follow easily by the task that clerks have to perform. Condition (2) follows with S1 as before.

3.3 Coercion Resistance

Intuitively, a voting protocol is coercion-resistant if it prevents voter coercion and vote buying. In other words, a coercer should not be able to influence the behavior of a voter. Below we present a more accurate, but still informal definition of coercion resistance, inspired by the (formal) definition given in [15].

This definition has two parameters. The first one is the goal γ of the coerced voter. This goal is what the voter would try to achieve in absence of coercion. Typically the goal is to vote for some particular candidate (or some particular ranking of candidates). The second parameter of the definition is a set of runs α, which contains almost all runs of the system, except for those that are unlikely to happen and would reveal to the coercer whether or not the coerced voter followed his instructions. For instance, if a particular candidate did not obtain any vote, it is clear that the coerced voter did not vote for this candidate, even though the coercer might have instructed the coerced voter to do so. Therefore, such (unlikely) runs are excluded from the set α.

In the definition of coercion resistance we imagine that the coercer provides the coerced voter with a *coercion strategy* or *instructions* v which the coercer wants the coerced voter to carry out. We do not restrict the set of possible coercion strategies in any way. The coercion strategy may, for instance, simply require the voter to follow the instructions given by the coercer over some communication channel, e.g., a phone. However, the setup is such that the coercer does not get direct access to the interface of the voter in the voting booth. Hence, the voter can lie about what she sees and does in the booth. Now, coercion resistance is defined as follows.

Definition 1 (informal). A voting protocol is *coercion resistant* in α with respect to γ, if for each coercion strategy v that can be carried out by the coerced voter there exists a counter strategy v' that the coerced voter can carry out instead such that

(i) the coerced voter, carrying out v', achieves his own goal γ (with high probability) regardless of the actions of the remaining participants,
(ii) the coercer cannot distinguish (or only with negligible probability) whether the coerced voter carries out v or v', for runs in α.

To prove coercion resistance for our protocol, we make the following assumptions:

C1. The bulletin board is honest and there is at least one honest participant in each of the following groups: auditors, clerks, printers, bundlers, and tellers. We do not assume the voting terminals to be honest. Dishonest participants may cooperate with the coercer.
C2. A voter can freely communicate with the coercer, even in the voting booth. But, in the voting booth, she can lie about what she sees and does.
For example, our protocol is still coercion resistant if a voter talks on the phone with the coercer, even in the voting booth. But a voter should not be able to take pictures or make videos in the voting booth (unless she could manipulate them on the fly, which, however, is unrealistic).

Assumption C2 is justified if electronic devices are forbidden in the voting booth. The use of such devices may even be detectable from outside the voting booth. However, some kind of communication or non-interactice procedure, e.g., by means of scratch cards [14], can still be possible. Our security analysis shows that this kind of communication does not undermine the coercion resistance of our protocol as it is captured by the free communication between the voter and the coercer we allow even in the voting booth. For example, the coercer can just tell the coerced voter what she would see on the scratch card.

Now, we will define γ and α for our protocol and show that for these parameters Definition 1 is satisfied. While we consider only one coerced voter, from theorems shown in [15], the results easily carry over to the case where multiple voters are coerced.

To define γ, we first observe that our protocol, as basically any other paper-based protocol, is prone to forced abstention attack: The coercer may instruct the coerced voter not to vote. To enforce this, the coercer could, for instance, observe the polling place or cooperate with dishonest clerks. Hence, it would be too strong to formulate the goal γ of a voter simply as "the voter successfully votes for a candidate of her choice". The protocol can at most guarantee that *if* the coerced voter casts a valid multi-ballot, she votes for a (ranking of) candidate(s) of her choice, even though the coercer might have instructed her to vote for a different candidate.

However, this is still too strong. If a teller would misbehave in an observable way, i.e., is not able to provide information that shows the compliance of the teller, then voters cannot hope for their votes to be counted correctly. The same is true if a voting terminal misbehaves in an observable way, i.e., if clerks or voters discover that not all scanned ballots have been posted on the bulletin board.

These observations lead to the following specification of γ. For a valid choice (ranking) of candidates z, we say that γ_z is achieved in a given run, if this run satisfies the following condition: If the coerced voter, having cast her multi-ballot, obtains a receipt (which implies that the marking on her multi-ballot is correct) and if the tellers and voting terminals do not misbehave in an observable way as explained above, then she successfully votes for z (i.e. her ballots are published and, when decrypted, show the choice z).

The set α consists of all the runs such that for each valid choice z there exists an honest voter who casts a multi-ballot according to z.

Now, we can prove the following theorem.

Theorem 1. *The protocol proposed in this paper is coercion resistant in α with respect to γ_z, for any valid choice z.*

This theorem implies, for example, that if the coercer wants the coerced voter to vote for some candidate c, then the coerced voter, by performing the counter strategy, can nevertheless vote for her candidate z without being caught by the coercer, given that the run belonged to α. Conversely, the voter cannot prove to the coercer that she voted as intended by the coercer (resistance to vote buying). The theorem also implies that the protocol is not prone to randomization,

chain voting, and kleptographic attacks. If these attacks were possible, the coerced voter could not vote for her choice without being detected by the coercer, contradicting the definition of coercion resistance.

Proof sketch of Theorem 1. Let v be a coercion strategy. We construct a counter strategy v' such that the conditions (i) and (ii) of Definition 1 hold for v and v'. The counter strategy v' works as follows: When in the voting booth, the coerced voter follows v in her head. If following v would result in an invalid ballot (e.g., the marking is invalid or the coerced voter is instructed to leave the booth without separating the two parts of the ballots), then the coerced voter performs v. Otherwise, the coerced voter fills in the multi-ballot according to *her own choice* z. (We assume that this can be done quickly.)

Condition (i) of Definition 1. To prove this condition, let us consider a run in which the coerced voter carries out v'. If the coerced voter does not obtain a receipt or if a teller or a voting terminal misbehaves in an observable way, then the goal γ_z is clearly achieved. Otherwise, according to the definition of v', the coerced voter prepares a (valid) multi-ballot according to her own choice z. This multi-ballot must be the one obtained from the clerks, since otherwise the serial number would be wrong (with high probability). Moreover, since by assumption C1 there is at least one honest member in the group of auditors and clerks, respectively, it follows, from the results shown in Section 3.2, that this multi-ballot is formed correctly (with high probability), i.e., candidate names printed in clear and encrypted on simple ballots coincide. Since the voting terminal and the tellers do not misbehave in an observable way, the multi-ballot cast by the coerced voter appears on the bulletin board and is counted correctly.

Condition (ii) of Definition 1. We argue that the information available to the coercer does not allow him to distinguish whether the coerced voter carries out v or v'. First, note that the information available to the coercer, including the information available from dishonest parties, before the coerced voter enters the voting booth is the same in both cases, as up to this point, the coerced voter follows the instructions of the coercer. The same is true for the information the coercer (including dishonest clerks) has right after the coerced voter leaves the booth: If the multi-ballot is not prepared correctly, v' is the same as v, and hence, the information the coercer has is the same independently of whether v or v' is performed. Clearly also all remaining steps will be identical in this case. So, we may from now on assume that the ballot is prepared correctly. Then, right after leaving the voting booth, there is no visible difference between the two cases, because, as we have assumed, all valid markings have the same pattern and all ciphertexts are still covered. Moreover, the serial number has to be the one issued to the coerced voter by the clerks at the beginning of the voting phase, as otherwise the multi-ballot would be invalid.

Now, consider the information available to the coercer (including dishonest clerks and the voting terminal) after the scratch strips are removed from the (valid) multi-ballot of the coerced voter. Due to the auditing in the ballot preparation phase, by (honest) auditors, and the voting phase, by (honest) clerks, the

multi-ballot issued to the coerced voter was not manufactured by the coercer and the coercer did not get a chance to uncover scratch strips. Moreover, the honest printer, together with the way candidate names and ciphertexts are covered, make sure that the coercer looses the connection between a candidate name and the ciphertext printed on a simple ballot. Also note that the coercer cannot simply decrypt ciphertexts, as this requires the participation of the honest teller, which we assume in C1.

In the tallying phase, the coercer looses the connections between the ciphertexts on the simple ballots cast by the coerced voter (and printed on the receipts) and the decrypted candidate names, again due to the honest teller.

It is easy to see that information obtained in audits also does not help the coercer to distinguish between carrying out v and v'.

Finally, consider the information sent directly to the coercer by the coerced voter. By the construction of the counter strategy v', this information corresponds to the one provided when carrying out v. Here we use that the coerced voter can lie about what she is actually doing in the voting booth (Assumption C2).

References

1. M. Abe. Universally Verifiable Mix-net with Verification Work Indendent of the Number of Mix-servers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 1998)*, pages 437–447, 1998.
2. B. Adida. *Advances in Cryptographic Voting Systems*. PhD thesis, MIT Department of Electrical Engineering and Computer Science, 2006.
3. B. Adida and R. L. Rivest. Scratch & vote: self-contained paper-based cryptographic voting. In *Workshop on Privacy in the Electronic Society (WPES 2006)*, pages 29–40, 2006.
4. J. C. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing (STOC 1994)*, pages 544–553. ACM Press, 1994.
5. J.-M. Bohli, J. Müller-Quade, and S. Röhrich. Bingo Voting: Secure and Coercion-Free Voting Using a Trusted Random Number Generator. In A. Alkassar and M. Volkamer, editors, *Proceedings of the First International Conference on E-Voting and Identity (VOTE-ID 2007)*, volume 4896 of *Lecture Notes in Computer Science*, pages 111–124. Springer, 2007.
6. D. Chaum. `http://punchscan.org/`.
7. D. Chaum. Secret-Ballot Receipts: True Voter-Verifiable Elections. *IEEE Security & Privacy*, 2(1):38–47, 2004.
8. D. Chaum, R. Carback, J. Clark, A. Essex, S. Popoveniuc, R. L. Rivest, P. Y. A. Ryan, E. Shen, and A. T. Sherman. Scantegrity II: End-to-End Verifiability for Optical Scan Election Systems using Invisible Ink Confirmation Codes. In D. L. Dill and T. Kohno, editors, *USENIX/ACCURATE Electronic Voting Technology (EVT 2008)*. USENIX Association, 2008.
9. D. Chaum, P.Y.A. Ryan, and S. Schneider. A practical, voter-verifiable election scheme. In *Proceedings of the 10th European Symposium on Research in Computer*

Security (ESORICS 2005), volume 3679 of *Lecture Notes in Computer Science*, pages 118–139. Springer, 2005.
10. David Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Commun. ACM*, 24(2):84–88, 1981.
11. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. *J. Cryptology*, 20(1):51–83, 2007.
12. M. Jakobsson, A. Juels, and R. L. Rivest. Making Mix Nets Robust for Electronic Voting by Randomized Partial Checking. In D. Boneh, editor, *Proceedings of the 11th USENIX Security Symposium*, pages 339–353, 2002.
13. A. Juels, D. Catalano, and M. Jakobsson. Coercion-resistant electronic elections. In *Proceedings of Workshop on Privacy in the Eletronic Society (WPES 2005)*. ACM Press, 2005.
14. J. Kelsey, A. Regenscheid, T. Moran, and D. Chaum. Scratch off attacks on end-to-end voting systems. Rump Session of the 28th Annual International Cryptology Conference (CRYPTO 2008).
15. R. Küsters and T. Truderung. An Epistemic Approach to Coercion-Resistance for Electronic Voting Protocols. In *2009 IEEE Symposium on Security and Privacy (S&P 2009)*, pages 251–266. IEEE Computer Society, 2009.
16. D. Lundin and P. Y. A. Ryan. Human Readable Paper Verification of Prêt à Voter. In S. Jajodia and J. López, editors, *Proceedings of the 13th European Symposium on Research in Computer Security (ESORICS 2008)*, volume 5283 of *Lecture Notes in Computer Science*, pages 379–395. Springer, 2008.
17. T. Moran and M. Naor. Split-ballot voting: everlasting privacy with distributed trust. In *ACM Conference on Computer and Communications Security (CCS 2007)*, pages 246–255, 2007.
18. C. A. Neff. Practical High Certainty Intent Verification for Encrypted Votes. `http://www.votehere.com/old/vhti/documentation/vsv-2.0.3638.pdf`.
19. S. Popoveniuc and B. Hosp. An introduction to PunchScan. In *IAVoSS Workshop on Trustworthy Elections (WOTE 2007)*, 2007.
20. B. Riva and A. Ta-Shma. Bare-Handed Electronic Voting with pre-processing. In *USENIX/ACCURATE Electronic Voting Technology (EVT 2007)*, 2007.
21. R. L. Rivest and W. D. Smith. Three Voting Protocols: ThreeBallot, VAV and Twin. In *USENIX/ACCURATE Electronic Voting Technology (EVT 2007)*, 2007.
22. P. Y. A. Ryan. A variant of the Chaum voter-verifiable scheme. In *Water, Innovation, Technology & Sustainability (WITS 2005)*, pages 81–88, 2005.
23. P. Y. A. Ryan. Prêt à Voter with Paillier Encryption. Technical Report CS-TR 1014, University of Newcastle upon Tyne, 2008.
24. P. Y. A. Ryan and S. A. Schneider. Prêt à Voter with Re-encryption Mixes. In *Proceedings of the 11th European Symposium on Research in Computer Security (ESORICS 2006)*, volume 4189 of *Lecture Notes in Computer Science*, pages 313–326. Springer, 2006.
25. K. Sako and J. Kilian. Receipt-Free Mix-Type Voting Scheme — A practical solution to the implementation of a voting booth. In *Advances in Cryptology — EUROCRYPT '95, International Conference on the Theory and Application of Cryptographic Techniques*, volume 921 of *Lecture Notes in Computer Science*, pages 393–403. Springer-Verlag, 1995.
26. Z. Xia, S. A. Schneider, J. Heather, and J. Traoré. Analysis, Improvement, and Simplification of Prêt à Voter with Paillier Encryption. In D. L. Dill and T. Kohno, editors, *USENIX/ACCURATE Electronic Voting Technology (EVT 2008)*. USENIX Association, 2008.