Bingo Voting: Secure and coercion-free voting using a trusted random number generator

Jens-Matthias Bohli, Jörn Müller-Quade, and Stefan Röhrich

Institut für Algorithmen und Kognitive Systeme / E.I.S.S.,
Universität Karlsruhe (TH)
76128 Karlsruhe, Germany
{bohli,muellerq,sr}@ira.uka.de

Abstract. It is debatable if current direct-recording electronic voting machines can sufficiently be trusted for a use in elections. Reports about malfunctions and possible ways of manipulation abound. Voting schemes have to fulfill seemingly contradictory requirements: On one hand the election process should be verifiable to prevent electoral fraud and on the other hand each vote should be deniable to avoid coercion and vote buying.
This work presents a new verifiable and coercion-free voting scheme *Bingo Voting*, which is based on a trusted random number generator. As a motivation for the new scheme two coercion/vote buying attacks on voting schemes are presented which show that it can be dangerous to let the voter contribute randomness to the voting scheme.
A proof-of-concept implementation of the scheme shows the practicality of the scheme: all costly computations can be moved to a non time critical pre-voting phase.

Keywords: secure electronic voting, coercion-free, receipt-free

1 Introduction

Elections have to meet a lot of requirements, e.g., the German constitution speaks about the selection of the members of German House of Representatives in general, direct, free, equal, and secret elections[1]. For security considerations of voting protocols, mainly the last three properties are of interest: An election should be free, i.e., nobody can be coerced to cast a certain vote, it should be equal, i.e., nobody can influence the result more than with her own vote, and it should be secret: no one is able to learn the votes of other people.

[1] Grundgesetz Art. 38(1): "Die Abgeordneten des Deutschen Bundestages werden in allgemeiner, unmittelbarer, freier, gleicher und geheimer Wahl gewählt."

Traditional voting schemes using paper and ballot boxes cannot be trusted to guarantee all these security properties. Ballot stuffing, miscounting, and the manipulation or destruction of votes during tallying are possible. Current voting machines cannot be considered to be a secure solution as experiences with malfunctioning machines in recent elections in the US showed.

These problems led to an increasing interest in voting schemes which allow the voter to verify that her vote was counted. However, such a proof should be meaningful only for the direct recipient, because otherwise coercion and vote buying become substantially simplified. Such schemes are called coercion-free or receipt-free[2].

An additional important requirement for voting schemes is usability. A scheme must be convincing in a very direct way and one cannot expect all voters to use electronic devices. This makes the design of a voting scheme even more difficult, because many cryptographic techniques cannot be used to directly convince humans.

Our Contribution

In this work we propose a new voting scheme, called Bingo Voting due to the use of a random number generator, like a bingo cage. The new scheme achieves:

- ballot casting assurance and universal verifiability, i.e., the voter can check if her own vote is cast and counted as intended, and everyone is able to verify that all votes are correctly counted as recorded on a bulletin board without learning the content.
- depending on the binding property of the commitments used the scheme offers either everlasting privacy or unconditional correctness.
- coercion-freeness, i.e., even if the voter deviates from the protocol she does not gain any evidence which allows her to prove anything about the contents of her vote.

These security properties are achieved relative to very realistic assumptions:

- a non interactive commitment scheme with some homomorphic properties is needed, e.g., Pedersen commitments [1]. If general zero-knowledge protocols are used in the post-voting phase, then no homomorphic properties are needed and if one is willing to use check samples

[2] The term receipt-free might be misleading as the voter indeed obtains a receipt.

instead of giving proofs then even physical commitments, e.g., using strong boxes, become possible.[3]
- A trusted random number generator is needed.
- The human capabilities assumed on the side of the voter are very limited. Basically the voter needs to check equality of two random numbers and check if her paper receipt has been posted to a bulletin board. The scheme remains secure if not all voters actually verify the process. It is enough if the attacker cannot predict who will be verifying.
- If a voter should not only be able to detect cheating attempts, but also to prove an electoral fraud, then the printed receipts should be difficult to forge.

To show the practical applicability of the new scheme it has been implemented in Java as a proof-of-concept.

Furthermore we present two new coercion/vote-buying attacks on existing schemes [2–5] which strongly suggest, that the voter should not be trusted to contribute her own randomness. This gives an additional motivation for the use of trusted random number generators.

2 Related Work

Several voting schemes have been proposed over the last years.

Protocols that require the user to provide random choices additional to the actual vote [2–4] will be analyzed in the following section. These protocols allow a practical attack which is overlooked in cryptographic models as it exploits the limited memory of humans involved in the voting process. ThreeBallot voting [6] also demands additional random choices from the voter and, as already outlined in the paper, a coercion attack becomes possible.

In Section 2.2 we will take a closer look at Punchscan [5] and point out an attack if the choice of a layer (step E.1 in [7]) is not done before the voter receives her ballot (as it was the case in earlier versions of Punchscan). This shows possible ways of coercion, when the voter has to make decisions after getting unpredictable input. We also describe an adaption of Punchscan to voting machines which overcomes the problem of checking the correct destruction of one of the ballot layers. However, we show that the voter's effort to check correctness of her vote remains high.

[3] However, samples instead of proofs would weaken the coercion-freeness as a small fraction of votes is opened.

Some other protocols use encryptions which make it difficult for the voter to directly check the correctness of the vote and therefore might not be trusted by some voters. Instances of such protocols are Prêt à Voter [8], Scratch & Vote [9], and Benaloh's simple verifiable elections [10].

2.1 Voting Schemes Using the Order of User Inputs

Neff [2] proposed a voting scheme which is based on the temporal order of the interaction between the voting machine and the user. Other schemes use the same idea, e.g., the scheme by Reynolds [3]. Moran and Naor present a scheme which ensures everlasting privacy [4] and they prove the coercion-resistance of their scheme in a simulation-based model.

The basic concept behind these schemes is, that the voting machine commits itself to some random values, e.g., by printing it on a receipt without showing it to the voter. After the machine is committed, the voter casts her vote and enters some randomness into the machine. This randomness is then used to generate a proof that the vote will be counted for the chosen candidate. In order to avoid coercion, the voter can also input random values for the other candidates, but this is done *before* the voting machine is committed, so the voting machine is able to produce fake proofs for these candidates. On a receipt every candidate is printed with the corresponding user choices. Crucial for these schemes is the order of the interaction, first the user enters dummy values for the other candidates, then the voting machine commits, after this the voter enters the random value for the real candidate, so that the machine cannot fake the proof for the real vote.

A "Babble" Attack Because of the limited memory of the human brain there may be the possibility of a vote-buying/coercion attack, if the scheme doesn't restrict the length of the random choices by the voter to very short lengths or there is a large number of candidates. Suppose the attacker provides the voter with an ear piece, i.e., a small radio receiver which allows the voter to listen to the attacker in the voting booth. The adversary now remotely babbles a long stream of random choices through the ear piece to the voter. The attacker coerces the voter to vote for a special candidate and to enter the random values she hears through the ear piece in the correct chronological order. Because the random choices for the real vote are entered last (and this is required by the schemes) the attacker can check if the random choice for the chosen candidate really was sent after the choices used for the other candidates.

This attack requires that there are sufficiently many random choices such that the voter can't memorize and rearrange the random values. Even if the voter had access to a pen and a paper in the booth the attack remains dangerous as the attacker could observe the time the voter spends in the voting booth: For very many or very long random values and an appropriate timing when this information is sent over the ear piece it takes time to write down all the random choices before starting to vote.

2.2 Punchscan

Paper ballot scheme Punchscan [7] is a paper based voting scheme that recently attracted attention. A Punchscan ballot consists of two paper sheets that are attached one upon the other (see Figure 1 for sample ballots). On the top page, the list of candidates is given, assigned to each candidate is a letter with the letters being in a random permutation. The upper sheet has several holes through which letters from the lower sheet are visible. Which letter shows in which hole is again randomized. To vote, the voter will look for the letter assigned to her candidate, find the letter in one of the holes, and mark the hole such that the mark is visible on both sheets. The voter can choose one layer of her ballot as a receipt to take home. A system of published commitments and reveals allows the voter to verify the tally without being able to prove what she voted for. For more details, we refer to the Punchscan webpage [5].

A Vote Buying Attack One way of vote buying is possible even though the receipts do not reveal the actual vote. A vote buyer may offer a reward for a certain top and bottom layer, respectively. Considering a contest with two choices, say, YES or NO, a vote buyer interested in convincing people to vote for NO might offer to pay for

- a top receipt where YES is assigned to the letter A and the left bullet is marked, or
- a bottom receipt where A appears in the left bullet and is marked.

Both layers together constitute a YES-vote. A voter who votes YES might hold both layers together, a top and bottom layer that would entitle her for the reward. However, she has to shred one of her papers. A voter for NO will have the benefiting layers on separate ballots, thus, being able to choose the corresponding receipt. To maximize the probability of the payment, voters are motivated to vote with NO. By enumerating all possible ballots, see Figure 1, this becomes obvious: only in the first

possible ballot the voter can qualify for the reward by voting YES. In the next two of the four possible ballots, the voter can qualify for the reward by voting NO, and for the last ballot, it is impossible to produce one of the rewarding layers. Voters following the attack unreservedly will vote in favour of the coercer with probability of 1/2, vote against the coercer with probability 1/4 and vote according to their own decision with probability 1/4.

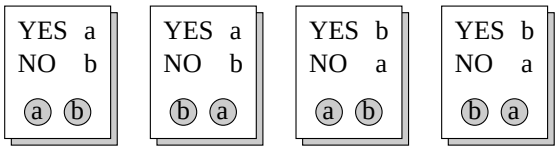

Fig. 1. Possible Punchscan ballots for a two-candidate race.

The success probability and the effect of this attack diminishes when more choices are available. Even though this attack is not published there is evidence that this attack is known to the designers of Punchscan, because a change was made which prevents this attack: In the first step of the election phase the voter is required to state her choice (top receipt or bottom receipt) before she can see the ballot. We think the above attack might be the motivation behind this step, as a connection to chain voting as indicated in [7] does not seem obvious.

2.3 Computer-aided Punchscan

Instead of using a trusted paper shredder[4], and to make the scheme more usable, we will next adapt Punchscan for usage on a voting machine. As in current voting machines a screen will offer the list with candidates and the voter casts her vote by pressing the according button. For Punchscan, the voter is finally asked to choose between the top or bottom layer. To avoid the above-mentioned vote buying attack, it is important that the voter announces which layer she intends to take home before seeing the ballot. Summing up, the voting process will be as follows:

1. The voting machine chooses a ballot.
2. The voter indicates her vote and choice of receipt.

[4] The trusted shredder is needed to check the destruction of the correct layer. Otherwise, a simple chain voting attack becomes possible.

3. The voter is presented both parts of the ballot to verify the accuracy of her vote.
4. The voter can take home the receipt of her choice. The other part of the ballot is destroyed.

The computer-aided Punchscan will be by far easier to use. However, it remains the rather complicated process of verifying the correctness of the vote. By the nature of Punchscan, it is necessary to follow through two permutations to check a vote, while there might be different presentations of the permutation.

The voting machine has to be trusted with the handling of the ballot during phases 1 and 2. I.e., the voter must be able to check that one ballot (identified by the serial number) is selected in phase 1. In phase 2, she indicates the vote and which layer to take home. The voter should not be able to see the layout of the ballot before the choice of the layer. On the other hand, the voting machine could unrecognizable modify the ballot layer that was not chosen. Therefore the machine has to be constructed in a way that makes obvious that the machine cannot alter the prepared punchscan ballots other than casting a vote. Both layers have to be shown to the voter in phase 3 to allow checking the correctness of her vote. This will require following two permutations, as it was the case with the paper ballot scheme. Finally, the voter receives the layer she has chosen. The second layer has to be destroyed inside the voting machine.

3 Bingo Voting

3.1 Basic Idea

The basic idea for a verifiable vote is to have ballots with a unique serial number. Each voter will cast her vote on the ballot and take a copy of the ballot home. When the election result is published, all ballots are published such that every voter can check that her vote is indeed counted for the intended candidate. Unfortunately, this simple voting protocol paves the way for vote buying or coercion. The voter can easily prove her vote by showing her ballot.

Once the result and all ballots are published, however, it may be possible for the voter to deny her vote, by fabricating a fake ballot for any listed vote of her choice. Thus, a plausible enhancement of the simple scheme would be to hand out fake ballots to the voter immediately in the polling booth. This might be difficult to realize in a paper-only system but possible with a machine that keeps track of all votes and is able to

issue a receipt for formerly casted ballots. To be able to give a fake receipt to the first voters, however, requires some previously set up *dummy votes* for all candidates as an initial pool to choose from. If the pool of fake votes is small, the correctness is only marginally affected.

However, if an attacker is able to coerce a lot of voters, the reuse of votes can be utilized for a vote-buying/coercion attack. The attacker would check the receipts of the voters he coerces for numbers which appear more than once. These numbers have to be fake votes for every voter besides one, this gives enough evidence to the adversary to coerce voters. The scheme we describe in the following will therefore make enough dummy votes for all voters available. Every voter gets besides a receipt for the vote actually cast faked receipts certifying a (dummy) vote for all other candidates. All the serial numbers can be printed on the same receipt, each number belonging to the candidate printed in the same line (see the Vote Receipt in Figure 3 for an example).

The tallying of the real votes taking into account the fake votes can be assured by means of cryptography.

On the receipt, every candidate has assigned a number representing the serial number of the ballot. The voter must be assured, that her actual vote—the number of the real ballot—is not a dummy vote. This is the case if the voter can actually witness the random generation of the fresh number that will represent her vote, while the dummy votes are previously determined and committed. In the following, we describe the protocol in a more formal and detailed way.

3.2 Preliminaries

The scenario we assume in the following is a poll with ℓ candidates and n eligible voters. In the polling booth is a trusted random number generator (RNG) and the voting machine available. The voting machine's printer—or the paper where the receipt is printed on—has to be trusted only if evidence for electoral fraud needs to be transferable. To actually alter the outcome of an election many votes have to be changed and it is therefore enough if the printer prevents large-scale forgery of receipts.

3.3 Pre-Voting Phase

Before election day, the voting authority will generate a list with $m = n \cdot \ell$ random numbers $N_1, \ldots, N_m$ representing the *dummy votes*. These votes are assigned equally to the candidates $P_1, \ldots, P_\ell$. The assignment is described by a mapping $\phi : \{1, \ldots, m\} \to \{1, \ldots, \ell\}$. An example for

candidate	random number
P_1	1234523134
P_2	6665876824
P_3	3422335718
P_1	3953692256
P_2	6875210932
P_3	3940959973
P_1	4516989092
P_2	5253736656
P_3	0671069428
P_1	1894180419
P_2	0838226760
P_3	2539035690

Fig. 2. Example list of dummy votes

such a list is given in Figure 2. Unconditionally hiding commitments[5] $C_1, \ldots, C_m$ to the pairs $(N_i, P_{\phi(i)})$ are computed using random coins r_i. The commitments $C_i = \mathsf{commit}((N_i, P_{\phi(i)}); r_i)$ are published such that every voter can convince herself of the commitments.

The pairs $(N_i, P_{\phi(i)})$ with random numbers and assigned candidates are distributed to the electronic voting machines, such that every machine has for each candidate one dummy vote per eligible voter on this machine.

3.4 Voting Phase

In the voting booth, a voter has to perform the following steps to cast a vote, which are also shown in Figure 3:

1. The voter will first indicate her vote by pressing the according candidate's button on the voting machine.
2. Next the random number generator generates a fresh random number R.
3. The random number R is transferred to the voting machine and assigned to the candidate of the voter's choice.
   For each other candidate, the voting machine will draw randomly one number out of the pool of dummy votes for the respective candidate. The machine will print out a receipt listing candidates and numbers: The candidate that was voted is assigned the fresh random number R, for the other candidates the respectively chosen dummy vote is shown. Figure 3 contains a sample receipt.

[5] Unconditionally hiding commitments yield everlasting privacy. To achieve unconditional correctness unconditionally binding commitments would have to been used.

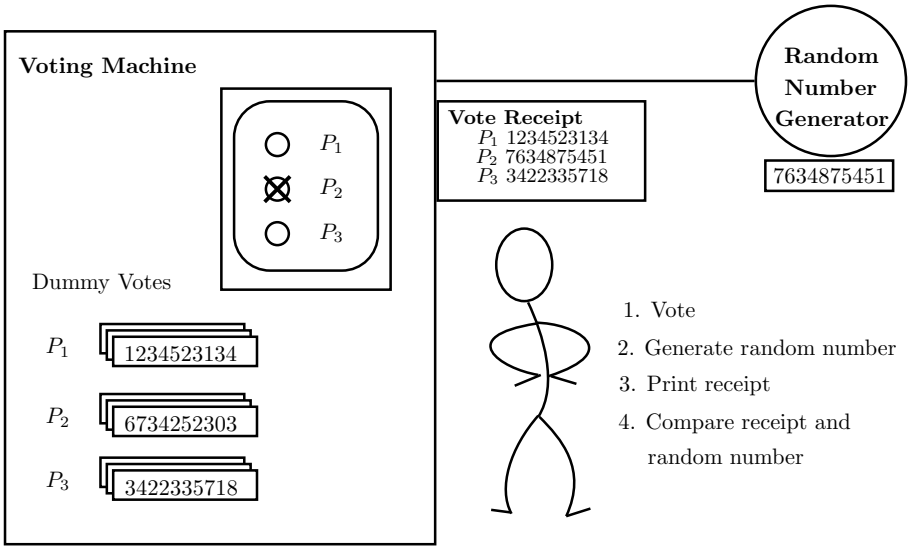

Fig. 3. Voting phase

4. The voter has to verify that the number shown on the random number generator is assigned to the party she intended to vote for. If this is not the case, the voter has to protest immediately.
5. The voter leaves the booth and takes out the receipt. For any outsider it is impossible to recognize the fresh random number and therefore the vote this ballot implies.

3.5 Post-Voting Phase

The voting machines will send a tally, copies of the vote receipts, and the remaining dummy vote/candidate pairs to the authority (or, if bandwidth is a concern, only the receipts and the election authority calculates the remaining data). The vote receipts should be mixed in the voting machine, so that the position of the receipt does not allow to draw conclusions to a specific voter, this limits the coercion abilities of the election authority. Sorting the receipts, say, in lexicographic order would be one way to mix the receipts. The election authority will publish the result consisting of four sections:

- the final outcome of the poll;
- all receipts issued;

– a list of all unused dummy vote/candidate pairs with the respective commitment and reveal information;
– non-interactive zero-knowledge proofs[6] for the correctness, i.e., that the dummy vote of every unopened commitment was indeed used on one receipt.

Now every voter can easily verify that her receipt is included in the list and therefore was counted for the tally. Every voter can verify that the commitments with unused dummy votes are correctly revealed and the number of remaining commitments is as expected. And finally the non-interactive proof can be checked.

The non-interactive proof certainly needs more explanation. After all unused commitments were revealed in the previous step, a list with the unopened commitments $C_i = \mathsf{commit}(N_i, P_{\phi(i)})$ for $i \in \mathcal{I}$ with a set $\mathcal{I}$ remains. If q out of n people voted, the set $\mathcal{I}$ is of size $n\ell - q(\ell - 1)$. It has to be proven that every committed dummy vote N_i for $i \in \mathcal{I}$ appears on a published receipt and every receipt contains $\ell - 1$ dummy votes. However, directly revealing the commitments would expose the votes.

Although the problem initially looks different to standard proofs in e-voting protocols, the common approach to apply a shuffle and then reveal [11] can be used. At first new commitments $\tilde{C}_j = \mathsf{commit}(R_j, P_{\mathsf{vote}(j)})$ on the fresh values R_j (the actual votes) and the voted party $P_{\mathsf{vote}(j)}$ on each receipt $j \in \{1, \ldots, q\}$ are computed. This could be done by the voting authority or already in the voting machine and be part of the data transmitted by the voting machine.

For every receipt j a list of commitments $\mathcal{C}_0^j$ is assigned containing $\tilde{C}_j$ and the dummy votes which were used for this receipt. It can easily be verified that $\ell - 1$ dummy vote commitments and only one fresh commitment is used. The union of these lists must contain all $[C_i]_{i \in \mathcal{I}}$ and every $[\tilde{C}_j]_{j \in \{1,\ldots,q\}}$, i.e., it contains commitments to all values that appear on the receipts. Every list $\mathcal{C}_0^j$ is now shuffled and revealed such that everyone can check, that indeed all values on a receipt appear in one commitment without being able to link the value to a primary commitment.

To prove the correctness of the shuffle we apply the method of *randomized partial checking* [12] to every single list. Starting from the list $\mathcal{C}_0^j$, each commitment is transformed to a new "masked" commitment to

[6] For efficiency reasons we will use "proofs" with 50 % soundness in this paper as many votes need to be manipulated to change the outcome of an election. To actually have zero-knowledge proofs one would need several iterations.

the same value.[7] Thereafter, the entries in the list are permuted to obtain a new list $\mathcal{C}_1^j$. This list is published and the same process is repeated to obtain a list $\mathcal{C}_2^j$. In an audit, for each entry in $\mathcal{C}_1^j$ depending on a random bit[8] either the first or second transformation is revealed. E.g., for revealing the first transformation, the element of $\mathcal{C}_1^j$ is linked to the element in $\mathcal{C}_0^j$ according to the permutation, and the correct masking of the commitment is proven without revealing the content of the commitment. Finally, the entries in list $\mathcal{C}_2^j$ are not linkable to the commitments in $\mathcal{C}_0^j$ but are with very high probability commitments to the same values. The commitments in $\mathcal{C}_2^j$ are revealed to check that the committed values match with the numbers on the voting receipts. An adversary cheating in this proof will be detected with a probability of 50 % per faked vote.

Our scheme scales rather well, after the creation of nl dummy commitments in the pre-voting phase, additionally q commitments are created for the real votes. During the proof, ql commitments have to be masked two times to yield new commitments to the same value.

4 Attacks, Assumptions and Security

The main security goal is certainly to ensure *correctness* of the election result. This should be verifiable by the voters without sacrificing the *secrecy* of the election. To enable verification, the voter gets a receipt, but this cannot be used for vote buying or coercion. Depending on the commitments, computational correctness and unconditional privacy (everlasting security) or unconditional correctness and computational secrecy are possible.

Different assumptions are made to prevent the attacks described: For correctness, our main assumption is a trusted random number generator attached to the voting machine. One could even imagine to build such a device mechanical, say, like a bingo cage, and read the random number by sensors.

In order to provide coercion protection we must rely on the voting machine and the voting authority. The voting machine must not be tam-

[7] E.g., using Pedersen commitments [1] it is easy to generate such masked commitments and to proof the equality.

[8] The randomness and unpredictability for the voting authority is important for the correctness property. It can be achieved by using a trusted random number generator in public or by other means, some of them are discussed in [4]. The most practical method probably would be using the Fiat-Shamir heuristic to handle the secure hash of the whole transcript to that point as randomness, but solutions like using physical sources or a coin flipping protocol are also possible.

pered with and must guarantee the secrecy of votes, likewise the voting booth has to be secured, e.g., no hidden cameras must be able to surveil the voting (this would already be a threat for classical elections). A secure channel is needed for the transfer of the dummy votes and later the results from the voting authority to the voting machine and back. These assumptions may sound hard to achieve, but even with today's paper-based voting schemes the election authorities have to be trusted to achieve coercion protection: Covertly marked ballots or similar may be used to identify voters and bribe them.

honest voting authority	hidden receipt	coercion protection
+	+	yes
+	−	yes
−	+	yes
−	−	no

Fig. 4. Coercion protection in different situations.

In Figure 4 we give a short overview about the coercion protection security property of our scheme in different situations. We distinguish the assumption of an honest or corrupted voting authority. Additionally we regard the situation, if the individual receipt of the voter is hidden from the adversary (e.g., because the voter has destroyed the receipt) in contrast to an adversary who knows which receipt belongs to which voter. The voting machine is assumed to be uncorrupted, because with full control over the voting machine an adversary may identify a voter and record her behaviour. The correctness property of the election is omitted in the table, because it is always fullfilled, even if the voting machine would be corrupted (but the random number generator is always assumed to be working properly).

4.1 Correctness

A vote is counted correctly, because the number on the ballot associated with the candidate for which the user has voted is a freshly generated random number. This is achieved by the trusted hardware number generator. The voter has to check if the number output on the random number generator is associated to her candidate. No other votes can be counted because the zero-knowledge proof guarantees $\ell - 1$ dummy votes on each

ballot and the probability that the fresh random number is a number which was used for a dummy vote is negligible.

Fraud should not only be detected, but there should be evidence of the fraud. Assume the voting machine will cast a vote for a candidate that was not intended by the voter, however, pretends to behave correctly. Our protocol makes this evident to the voter immediately in the polling booth: the number that is shown on the RNG is not printed on the right place. In this case the voter can immediately be given a possibility to cancel the previous vote and revote (this will, however, need suitably more commitments to be prepared in the pre-voting phase). To avoid the situation that a receipt does not show up on the tally, one could imagine trusted printers, which digitally sign their output or use special unforgeable paper to allow the voter a prompt protest. Each voter can check if her receipt appears on the list of ballots which is published after the election. If a voter does not find her receipt, she is able to complain and the special paper in the voting machine allows to prove that she indeed has a valid receipt.

According to the definitions in [13] we achieve *cast as intended* by comparing and checking the receipt and the output of the random number generator by the voter, the vote is *recorded as cast* can easily be checked through comparison with the public list of receipts, and the property *counted as recorded* is achieved by the (public) proof at the end of the election, so that *universal verifiability* is reached. Through the *end-to-end* and *direct verification* of "cast as intended" and "recorded as cast" we achieve *ballot casting assurance* (for a really direct and immediate verification the public list of receipts has to be updated with new votes in real time which would ease vote coercion attacks by the voting authority, see the next section).

4.2 Coercion protection

The vote receipt cannot be used to prove anything about the contents of a vote to a third party. The hiding property of the commitments in the published dummy votes together with the information of the zero-knowledge proof does not allow an adversary to distinguish the random number of the trusted random number generator from the dummy votes which were stored in the voting machine and published as a commitment.

Coercion protection against the voting authority can only be achieved if the voting machine is uncorrupted and either the commitments of the pre-voting phase were generated, deployed and later used for proving the result in a secure way, e.g., via secure multi-party computations with

enough uncorrupted parties, or the voting authority cannot access the receipt of the individual voters in order to match votes.

In case the election authority is corrupted (or corrupted parties somehow gain access to the unveil information of the commitments to the dummy votes) there is no way to prevent coercion, even if the voting machines are working properly: One can be forced to bring the printed receipt which, in the case described, reveals the party voted for. However, another relevant attack can to a certain extend be prevented, even if the adversary has access to the secret commitment information.

The voting machine mixes and thus anonymizes the votes, hence the secrecy of a vote can be maintained as long as the attacker cannot obtain the specific receipt of the voter. To protect voters from a coercion attack after having voted, some non-government organisations (NGOs) could collect paper receipts to check on the correctness of the vote and to help voters, who fear coercion by a corrupted election authority, to get rid of their receipts. If it is possible to anonymously dispose the paper receipt the voter cannot be coerced afterwards and the correctness of the vote will still be checked with a high probability unless many of the NGOs are corrupted.

5 Implementation

We implemented a proof-of-concept implementation in Java [14]. Pedersen commitments [1] were used as described for the underlying commitment scheme, therefore everlasting security is achieved. Our straightforward implementation in Java, which is not optimized for speed, needs about 0.6 seconds per potential voter (assuming a voter participation of 80 % and five candidates) for all three phases together on a standard 3 GHz Pentium 4 CPU and a bit length of 1000 for the commitments. Most of the time consuming work, generation of and calculations with the commitments, can be separated to the pre-voting phase, where speed is not a very important factor. Also the output size is acceptable, even including some debugging information, repeating many things and writing the commitments as ASCII encoded decimal numbers under 10 KBytes per potential voter are used. This enables a voter to download the whole information needed to check the proof of her electoral district.

6 Outlook

Further research is also required to enhance the usability and reduce the administrative requirements of voting schemes. Possibly, a combination

of existing schemes, like the scheme of Moran and Naor [4] and our Bingo Voting, would be an improvement in this respect.

The combined scheme would have an user interface similar to our Bingo Voting scheme. The voter's only input is the choice of a candidate. After the voter has chosen her candidate, a random number is generated by a trusted random number generator and transferred to the voting machine. The machine prints a receipt containing the name of each candidate with an associated random number generated by the voting machine itself for the not elected candidates and the number of the random number generator for the elected candidate. The voter has to check if the number of her candidate corresponds to the output of the random number generator. Additionally to Bingo Voting, the voting machine commits to proof information before the generation of the trusted random number by printing it on the receipt and the voter has to check that it is done before the random number generator is started. This information is used to generate proofs of the election result as in [4].

This new scheme avoids the babbel attack and requires only a slightly weaker assumption as in [4] because the commitment does not have to be hidden from the voter. Still, it must be assumed, that the voting machine cannot exchange the commitment after receiving the trusted random number. Compared to Bingo Voting, this assumption avoids the administrative effort for generating and storing of the dummy votes.

7 Conclusions

We have shown, that many voting protocols where the user needs to make decisions beyond choosing one candidate are susceptible to coercion attacks. We could reveal new coercion attacks to recently proposed voting protocols. To avoid those attacks, we introduce the assumption of a trusted random number generator inside the polling booth. We have presented a protocol basing on a random number generator, that makes it easy for the voter to vote and check correctness of the vote and have demonstrated that implementing our scheme is practical.

An open problem remaining is to find a suitable and realistic model for the treatment of voting protocols. Attacks like the babble attack are not covered by any security model known to us, even enhancements of very strong simulation based models to handle coercion ignore such a threat.

References

1. Pedersen, T.P.: Non-interactive and Information-Theoretic Secure Verifiable Secret Sharing. In Feigenbaum, J., ed.: Advances in Cryptology – CRYPTO '91: Proceedings. Volume 576 of Lecture Notes in Computer Science., Springer (1991) 129–140
2. Neff, C.A.: Practical high certainty intent verification for encrypted votes. Draft at `http://www.votehere.net/vhti/documentation/vsv-2.0.3638.pdf` (2004)
3. Reynolds, D.J.: A method for electronic voting with Coercion-free receipt. FEE 2005 (2005) Presentation: `http://www.win.tue.nl/~berry/fee2005/presentations/reynolds.ppt`.
4. Moran, T., Naor, M.: Receipt-Free Universally-Verifiable Voting With Everlasting Privacy. In Dwork, C., ed.: Advances in Cryptology – CRYPTO 2006. Volume 4117 of Lecture Notes in Computer Science., Springer (2006) 373–392
5. Chaum, D.: Punchscan (2006) `http://punchscan.org/`.
6. Rivest, R.L.: The ThreeBallot Voting System (2006) Draft online available at time of writing `http://theory.lcs.mit.edu/~rivest/Rivest-TheThreeBallotVotingSystem.pdf`.
7. Popoveniuc, S., Hosp, B.: An Introduction to Punchscan. IAVoSS Workshop On Trustworthy Elections, WOTE 2006 (2006) `http://punchscan.org/papers/popoveniuc_hosp_punchscan_introduction.pdf`.
8. Chaum, D., Ryan, P.Y., Schneider, S.: A Practical Voter-Verifiable Election Scheme. In De Capitani di Vimercati, S., Syverson, P., Gollmann, D., eds.: Computer Security – ESORICS 2005. Volume 3679 of Lecture Notes in Computer Science., Springer (2005) 118–139
9. Adida, B., Rivest, R.L.: Scratch & Vote: Self-Contained Paper-Based Cryptographic Voting. In: WPES '06: Proceedings of the 5th ACM workshop on Privacy in electronic society, ACM Press (2006) 29–40
10. Benaloh, J.: Simple Verifiable Elections. In: EVT '06, Proceedings of the First Usenix/ACCURATE Electronic Voting Technology Workshop, August 1st 2006, Vancouver, BC, Canada. (2006) `http://www.usenix.org/events/evt06/tech/full_papers/benaloh/benaloh.pdf`.
11. Chaum, D.: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. Communications of the ACM **24** (1981) 84–88
12. Jakobsson, M., Juels, A., Rivest, R.L.: Making Mix Nets Robust For Electronic Voting By Randomized Partial Checking. In: USENIX Security Symposium. (2002) 339–353
13. Adida, B., Neff, C.A.: Ballot Casting Assurance. In: EVT '06, Proceedings of the First Usenix/ACCURATE Electronic Voting Technology Workshop, August 1st 2006, Vancouver, BC, Canada. (2006) `http://www.usenix.org/events/evt06/tech/full_papers/adida/adida.pdf`.
14. Sun Microsystems: Java Platform, Standard Edition (2006) `http://java.sun.com/`.