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

Jens-Matthias Bohli[1]*, Jörn Müller-Quade[2], and Stefan Röhrich[2]

[1] NEC Laboratories Europe, Network Research Division,
Kurfürsten-Anlage 36, 69115 Heidelberg, Germany
[2] 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[3]. 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.

Traditional voting schemes using paper and ballot boxes cannot be trusted to guarantee all these security properties. Ballot stuffing, miscounting, and the

* Work done while the author was at Universität Karlsruhe (TH).

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

manipulation or destruction of votes during tallying are possible. Current voting machines cannot be considered to be a secure solution as studies about machines used in practice [Uni07, GHB+06] 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[4].

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 apart from the voting machine. 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, comparable to 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.

Security properties like anonymity or eligibility (i.e., one vote per eligible voter) are, in contrast to purely electronic voting schemes, easily obtained by traditional methods. The authorization is handled in front of the voting booth and an eligible voter may enter once to cast his vote. The voting machine reorders the votes and has to be trusted in order to guarantee anonymity.

The voting scheme offers a very high usability. Only very limited capabilities on the side of the voter are required. The voting process corresponds to the voting with today's voting machines: the voter has to press the button that is assigned to the intended candidate. To ensure the correctness of her vote, the voter only 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 as long as the attacker cannot predict which voter actually will be verifying.

The security properties listed above are achieved relative to very realistic assumptions:

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

– a non interactive commitment scheme with some homomorphic properties is needed, e.g., Pedersen commitments [Ped91]. If general zero-knowledge protocols are used in the post-voting phase, then these homomorphic properties are not needed. Furthermore, if one is willing to use check samples instead of giving proofs then even physical commitments, e.g., using strong boxes, become possible.[5]
– A trusted random number generator is needed.
– 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 [Nef04, Rey05, MN06, Cha06] 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 [Nef04, Rey05, MN06] 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. The MarkPledge scheme of [AN06] does not associate the human generated randomness to a specific candidate and is not vulnerable to these attacks. However, voters have to commit to random choices which introduces new assumptions or makes the scheme impractical. ThreeBallot voting [Riv06] 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 [Cha06, PH06a] and point out an attack if the choice of a layer (step E.1 in [PH06b]) is not done before the voter receives her ballot (as it was the case in earlier versions of Punchscan [PH06a]). This shows possible ways of coercion, when the voter has to make decisions after getting unpredictable input. Examples for further protocols having the usage of cryptographic paper ballots in common are Prêt à Voter [CRS05], Scratch & Vote [AR06], and Benaloh's simple verifiable elections [Ben06].

2.1 Voting Schemes Using the Order of User Inputs

Neff [Nef04] proposed a voting scheme which is based on the temporal order of the interaction between the voting machine and the user. Other schemes use

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

the same idea, e.g., the scheme by Reynolds [Rey05]. Moran and Naor present a scheme which ensures everlasting privacy [MN06] 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.

This attack might seem unrealistic and only a special case of using a video camera to document the complete voting procedure of a coerced voter. But because an ear piece can be very small and requires only one-way communication (or no communication, if devices like a sealed audio-player with no user control are used) we think this attack shouldn't be ignored. Especially, the babble attack points out, that coercion-free voting schemes should clearly state the amount of isolation which is assumed in the voting booth, if there must be more protection than preventing the attacker to see what happens there.

2.2 Punchscan

Paper ballot scheme Punchscan [Cha06, PH06a, PH06b] 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 in a random permutation. The upper sheet has several holes through which letters from the lower sheet (in randomized order) are visible. To vote, the voter will mark the letter assigned to her candidate 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 [Cha06].

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.

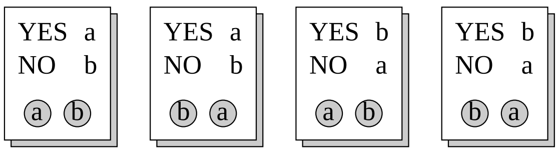

Fig. 1. Possible Punchscan ballots for a two-candidate race.

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 $1/4$.

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 [PH06b] in contrast to [PH06a]: 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 [PH06b] does not seem obvious.

This attack may seem similar to a vote randomization attack in which a voter should always mark the left bullet and which is still possible in the modified version. But the attack described here increases the probability to vote for a specific choice and therefore is much stronger.

3 Bingo Voting

3.1 Basic Idea

A 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.

However, the voter were able to deny her vote if she could fabricate and present a fake ballot that is published in the list of all votes. Such a fake ballot can be imagined as the vote of another voter or a *dummy vote*—a vote that appears in the list of votes but does not count.

To avoid problems that arise from the approach of swapping receipts with other voters our scheme makes enough dummy votes available to give to every voter a receipt for every (not elected) candidate in addition to the receipt for the elected candidate. All the serial numbers will be printed on the same receipt, each number printed next to the corresponding candidate in one line (see the Vote Receipt in Figure 2 for an example). Thus, a vote is simply represented by a number.

The tallying of the real votes taking into account the dummy votes will be assured by means of cryptography. To achieve this the dummy votes will be chosen out of a pool of random numbers that are committed before the election starts. 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 were previously determined and committed. In the polling booth, the voter has only to check, that the fresh number appears on the receipt as intended. The cryptographic proof that every ballot contains only one real vote and that dummy votes do not count is published with the tally and can be checked afterwards.

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. To ease the description of the scheme we restrict to the case of a single voting machine, but an extension to multiple machines is straightforward. In the polling booth is a trusted random number generator (RNG) and the voting machine available.

3.3 Pre-Voting Phase

Before election day, the voting machine will generate for every candidate P_i n random numbers N_j^i, yielding dummy vote pairs (N_j^i, P_i) for the candidate. Together, $m = n \cdot \ell$ dummy votes are created for the candidates $P_1, \dots, P_\ell$. Unconditionally hiding commitments[6] $C_1, \dots, C_m$ to the dummy vote pairs are computed using random coins r_j^i. The commitments $C_j^i = \mathsf{commit}((N_j^i, P_i); r_j^i)$ are shuffled and published on a bulletin board before the election starts. Additionally, the equal distribution of the committed dummy votes to the candidates is proven without opening the commitments[7].

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 2:

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 2 contains a sample receipt.
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.[8]
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.

[6] Unconditionally hiding commitments yield everlasting privacy. To achieve unconditional correctness unconditionally binding commitments would have to been used.
[7] This can be done via randomized partial checking [JJR02] if the part of the commitment containing the candidate can be opened without revealing the random number.
[8] See also the discussion in Section 4.

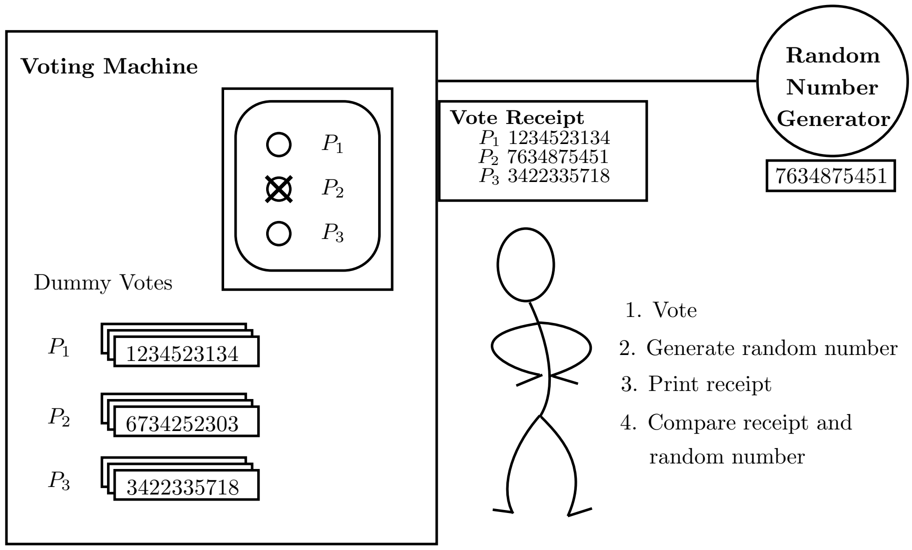

Fig. 2. Voting phase

3.5 Post-Voting Phase

After the election the voting machine calculates the result and sends it together with a proof of correctness to a public bulletin board. The published data consists of four sections:

1. the final outcome of the poll;
2. a lexicographically sorted list of all receipts issued;
3. a list of all unused dummy vote/candidate pairs with the respective commitment and reveal information;
4. non-interactive zero-knowledge proofs[9] 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 number of remaining commitments is as expected (i.e., every vote for a candidate causes a fresh random number, so one dummy vote isn't needed. Therefore, if a candidate has a votes and b voters were absent, the candidate should have $n-a-b$ remaining commitments.) and the reveal information leads to the given tally. Finally, the non-interactive proof for the correctness of the receipts can be checked.

[9] 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.

In the following we describe a proof for the correctness of the receipts which is efficient and uses standard shuffling techniques[10] which are common in e-voting protocols. For this, we need commitments with a special homomorphic property: it should be possible to generate a new "masked" or re-randomized version of a commitment, which is a new commitment to the old value (using new randomness), and the correctness of this masking should be provable. Using Pedersen commitments [Ped91] it is easy to generate such masked commitments and to prove the equality of the old and new values.

The proof shows that all unopened (used) commitments are assigned to receipts. By a counting argument, the revealed unused commitments correspond to the election result (and a constant offset for absent voters). It has the following steps for every single receipt:

- A new commitment C to (R, P) is generated, where P is the really chosen candidate and R the output of the trusted RNG.
- The $\ell - 1$ commitments to dummy values used for this receipt and C are published (without revealing). This list of commitments is called $\mathcal{C}_{left}$, everyone can check that $\ell - 1$ previous published commitments are used.
- The commitments are masked to new commitments to the same value and shuffled. The new list $\mathcal{C}_{middle}$ is published.
- The last step is repeated with the commitments of $\mathcal{C}_{middle}$, yielding a new list $\mathcal{C}_{right}$ which is published.
- The commitments of $\mathcal{C}_{right}$ are revealed. The contents must equal the corresponding numbers on the receipt.
- A random bit s is chosen[11]. If the bit s is 0, the association between the commitments of $\mathcal{C}_{left}$ and $\mathcal{C}_{middle}$ is published and the correctness of the masking is proven without revealing the commitments of these lists[12]. If the ith bit of s is 1 the according relation between commitments of $\mathcal{C}_{middle}$ and $\mathcal{C}_{right}$ is proven.

Any manipulation in the proof for a single receipt affects at most ℓ votes and is detected with a probability of 50 % per manipulated receipt. If a higher assurance is necessary (e.g., if the vote counts of the candidates are very close to each other), the proof for the single receipts can be repeated to yield higher soundness.

[10] Commitments are shuffled and later revealed [Cha81]. To prove the correctness of the shuffle we apply the method of *randomized partial checking* [JJR02] to every single ballot.

[11] 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 [MN06]. 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.

[12] For Pedersen Commitments this is simply done by publishing the random choices used for the re-randomization.

Our scheme scales rather well, after the creation of $n\ell$ dummy commitments in the pre-voting phase, additionally q commitments are created for the real votes. During the proof, $q\ell$ commitments have to be masked twice to yield new commitments to the same value.

4 Attacks, Assumptions and Security

Different assumptions are necessary: The main security goal is certainly to ensure *correctness* of the election result. The key assumption to ensure correctness is a trusted random number generator attached to the voting machine. It should provide some clearly evident form of randomness, we even recommend a mechanical device, similar to devices used in lotteries or to a bingo cage, and read the random number by sensors.

In order to provide *coercion protection*, we must rely on the voting machine: The voting machine must not be tampered with and must guarantee the secrecy of votes. Likewise the voting booth has to be secured, e.g., no hidden cameras may be able to surveil the voting (this would already be a threat for classical elections).

Correctness	RNG is uncorrupted
Coercion protection	RNG is uncorrupted **and** (voting machine is uncorrupted **or** (voting machine is only passively corrupted **and** actions on the voting machine/receipts are not linked to voters))

Table 1. Assumptions of Bingo Voting.

Before we discuss the security in more details, Table 1 summarizes the assumptions.

4.1 Correctness

At the start of the election it is proven that each candidate has the same number of dummy votes. For each vote a fresh random number is generated and associated with the candidate voted for. Hence for each vote for a specific candidate one dummy vote for this candidate is left unused. A corrupted voting machine could generate a fresh number to a candidate of its choice if the fresh random number equals a dummy vote, but the probability for this equality is negligible for a trusted random number generator[13]. The fresh number cannot be associated to another candidate, because this is checked by the voter in the booth.

[13] For the correctness the random numbers need not be uniformly distributed. It is sufficient if the attacker cannot predict the numbers.

Also no additional fresh numbers can be introduced as the zero-knowledge proof guarantees $\ell - 1$ dummy votes on each ballot.

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. Additional care has to be taken to avoid denial of service attacks by an always complaining voter.). 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 in the list, she must be able to complain. We intend therefore a trusted printer, which uses unforgeable paper to allow the voter to prove that she indeed has a valid receipt.

According to the definitions in [AN06] 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 a non-immediate notion of *ballot casting assurance*. For a really direct and immediate verification the public list of receipts has to be updated with new votes and their proofs in real time. This is possible (probably with the addition of some helper organisations like the proposal for MarkPledge in [AN06]), but it would ease vote coercion attacks by an attacker with access to a corrupted voting machine.

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. Therefore, the random number generator has to be trusted for coercion protection, too, so that it doesn't generate numbers which can be recognized by an attacker.

In order to gain privacy and coercion protection you also have to trust the voting machine. It mustn't be actively corrupted, e.g., the addition of a secret camera together with the recording of the votes would clearly violate privacy. As with most voting machines, coercion attacks are also possible if you only change the software of the machine, like adding a secret command to the software of the voting machine, which is activated by some special keystrokes which a voter is coerced to press, and which votes in favour of the attacker and records this.

However, another relevant attack can to a certain extend be prevented, even if the adversary has access to the secret commitment information or passively corrupted the voting machine (i.e., he learns every information the voting machine knows, but doesn't change the voting machine or its software). The secrecy of a vote can be maintained as long as the attacker cannot obtain the specific receipt of the voter or otherwise link the voting process to an individual voter. To protect voters from this some non-government organisations (NGOs) could collect paper receipts to check on the correctness of the vote. 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.

The concrete voting machine has to be designed very carefully in order to avoid attacks comparable to side-channel attacks, e.g., the state of the voting machine (like if it's waiting for input or printing a receipt) should be hidden from persons outside of the voting booth and the output of the random number generator and the presentation of the receipt to the voter should be very close in time. This avoids a kind of "reverse" babble attack where the voter is forced to use a microphone and read the numbers presented to him.

5 Implementation

We implemented a proof-of-concept implementation in Java [Sun06]. Pedersen commitments [Ped91] 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.

The prototype is mainly a demonstration for the feasibility of computation time and size of the proofs, a more detailed system, especially with a real hardware number generator for the voting booth, still has to be realized in order to analyze the whole voting process from a systems perspective as done for other schemes in [KSW05, RP05].

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 [MN06] and our Bingo Voting, would be an improvement in this respect.

The combined scheme would have a 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. 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 [MN06].

This new scheme avoids the babble attack and even requires a slightly weaker assumption for the printer compared to [MN06], because the commitment does not have to be hidden from the voter. Still, it must be assumed, that the voting machine cannot exchange or generate the commitment after receiving the trusted random number. Compared to Bingo Voting, this assumption avoids the administrative effort of generating and storing the dummy votes. However, the voter has to be more careful to detect *message reordering attacks*[14] [KSW05] which may affect the correctness property, while Bingo Voting isn't vulnerable to this kind of attacks to forge elections (but a coercion attack might be possible through this corruption of the voting machine).

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.

[14] In such an attack a corrupted voting machine maliciously executes the protocol steps with the human in the wrong order to be able to fake a vote.

References

[AN06] Ben Adida and C. Andrew Neff. 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`.

[AR06] Ben Adida and Ronald L. Rivest. Scratch & Vote: Self-Contained Paper-Based Cryptographic Voting. In *WPES '06: Proceedings of the 5th ACM workshop on Privacy in electronic society*, pages 29–40. ACM Press, 2006.

[Ben06] Josh Benaloh. 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`.

[Cha81] David Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

[Cha06] David Chaum. Punchscan, 2006. `http://punchscan.org/`.

[CRS05] David Chaum, Peter Y.A. Ryan, and Steve Schneider. A Practical Voter-Verifiable Election Scheme. In Sabrina De Capitani di Vimercati, Paul Syverson, and Dieter Gollmann, editors, *Computer Security – ESORICS 2005*, volume 3679 of *Lecture Notes in Computer Science*, pages 118–139. Springer, 2005.

[GHB+06] Rop Gonggrijp, Willem-Jan Hengeveld, Andreas Bogk, Dirk Engling, Hannes Mehnert, Frank Rieger, Pascal Scheffers, and Barry Wels. Nedap/Groenendaal ES3 voting computer – a security analysis, 2006. `http://www.wijvertrouwenstemcomputersniet.nl/images/9/91/Es3b-en.pdf`.

[JJR02] Markus Jakobsson, Ari Juels, and Ronald L. Rivest. Making Mix Nets Robust For Electronic Voting By Randomized Partial Checking. In *USENIX Security Symposium*, pages 339–353, 2002.

[KSW05] Chris Karlof, Naveen Sastry, and David Wagner. Cryptographic Voting Protocols: A Systems Perspective. In *Proceedings of the Fourteenth USENIX Security Symposium (USENIX Security 2005)*, pages 33–50, August 2005.

[MN06] Tal Moran and Moni Naor. Receipt-Free Universally-Verifiable Voting With Everlasting Privacy. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 373–392. Springer, August 2006.

[Nef04] C. Andrew Neff. Practical high certainty intent verification for encrypted votes. Draft at `http://www.votehere.net/vhti/documentation/vsv-2.0.3638.pdf`, 2004.

[Ped91] Torben Pryds Pedersen. Non-interactive and Information-Theoretic Secure Verifiable Secret Sharing. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO '91: Proceedings*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, 1991.

[PH06a] Stefan Popoveniuc and Ben Hosp. An Introduction to Punchscan. Threat Analyses for Voting System Categories, A Workshop on Rating Voting Methods, VSRW 06, June 2006. `http://vote.cs.gwu.edu/vsrw2006/papers/9.pdf`.

[PH06b] Stefan Popoveniuc and Ben Hosp. An Introduction to Punchscan. IAVoSS Workshop On Trustworthy Elections, WOTE 2006,

2006. http://punchscan.org/papers/popoveniuc_hosp_punchscan_introduction.pdf, online version dated 2006-10-15.

[Rey05] David J. Reynolds. A method for electronic voting with Coercion-free receipt. FEE 2005, 2005. Presentation: http://www.win.tue.nl/~berry/fee2005/presentations/reynolds.ppt.

[Riv06] Ronald L. Rivest. The ThreeBallot Voting System, October 2006. Draft online available at time of writing http://theory.lcs.mit.edu/~rivest/Rivest-TheThreeBallotVotingSystem.pdf.

[RP05] Peter Y. A. Ryan and Thea Peacock. Prêt à Voter: a Systems Perspective. Technical Report CS-TR-929, School of Computing Science, University of Newcastle, 2005. http://www.cs.ncl.ac.uk/research/pubs/trs/papers/929.pdf.

[Sun06] Sun Microsystems. Java Platform, Standard Edition, 2006. http://java.sun.com/.

[Uni07] University of California. Reports of top-to-bottom review of voting machines, 2007. http://www.sos.ca.gov/elections/elections_vsr.htm.