

eTegrity and ePunchScan

David Chaum, Stefan Popoveniuc, Poorvi Vora*

September 28, 2009

Abstract

We examine the problem of an electronic end-to-end voting system that does not require paper. As a first step, we define two protocols based on the end-to-end protocols of Scantegrity and PunchScan respectively. Using either protocol, the voter votes on an electronic voting machine in a polling booth. We also assume the voter may bring with him a (personally trusted) voter computing device that aids him in ensuring his vote is recorded correctly; this device performs checks equivalent to the ballot audits and commitment checks performed in the Scantegrity and PunchScan protocols. We assume that the computing device does not collude with the voting system to change the tally, but can, on the other hand, be inspected by a coercer who wishes to determine if the voter followed instructions. (The voter may, for example, bring with him a SmartPhone programmed by a public interest organization trusted by him.) We informally examine the security properties of the systems based on these protocols.

1 Introduction

Following the recent problems with electronic voting machines, there has been renewed interest in cryptographic voting protocols, first invented by Chaum in 1981 [7]. Recent cryptographic protocols are *software independent* as defined by Rivest and Wack [13, 12]: any errors in the tally—whether caused by intentional or unintentional errors in the voting system software—are detectable with overwhelming probability through a cryptographic post-election audit, whose results can be verified by anyone. Note that the audit does not verify that the software computing the tally is correct, only that the declared outcome of the particular election is correct; this is sufficient for the purposes of the election. The newer protocols are practical, have been prototyped and tested in real elections. For example, *PunchScan* [11] has been used in several binding elections, including those of the Graduate Student Association of the University of Ottawa; *Helios* [1], a remote electronic voting system, was recently used for the Recteur election at the Universite Catholique de Louvain (UCL), Belgium; and *Scantegrity* [5, 8] will be used for municipal elections in the city of Takoma Park, MD, USA. The use of the newer protocols in public elections would lead to a paradigm shift—from the expensive and incomplete verification of voting system software, to the individual verification of each election outcome with very high certainty. Thus, voting technology research is poised to make one of its most significant contributions ever, allowing voters to verify elections without requiring them to trust election officials or voting system technology.

Voting protocols should provide *end-to-end* independent verifiability of the tally; that is, the tally verifiability should be a property linking system input (votes) to system output (tally) without requiring (a) the trustworthiness of voting system components, including voting system software (b)

*This author supported in part by the National Science Foundation, Grant CNS-0831149

a secure chain of custody of the votes, or (c) trustworthiness of election officials. Protocol should simultaneously achieve incoercibility (an adversary cannot determine how a voter voted even if the voter colludes with the adversary). Currently, protocols with the strongest verifiability and incoercibility properties require the use of paper, either for ballots, or voter receipts, or both. Paper-based protocols are difficult to administer; this could considerably diminish the impact of their benefits to democracies worldwide. In this paper, we examine whether the benefits of two specific paper-based cryptographic protocols—PunchScan [11] and Scantegrity [5, 8]—can be made available in electronic protocols that do not require the use of paper.

In the literature on classical voting protocols, a voter used a trusted computational device to encrypt his vote. The collection of all vote encryptions was made public (for example, on a website) and voters could check that their encryptions were present in the collection. Thereafter, the computation of the election outcome from the collection of encrypted votes could be verified by anyone. The use of the trusted computational device to encrypt the vote led to problems with coercibility, as the voter generated a trail that could be used by an adversary to determine how he voted. An important contribution of the paper-based protocols is the design of paper ballots that enable voters to encrypt their own votes, thus avoiding, to some degree, the risk of coercion. Paper ballots may be audited to ensure that votes are being correctly encrypted. Because the problem of tally-verifiability has been addressed satisfactorily in the literature if votes are correctly encrypted, this paper will focus on the problem of incoercible vote encryption without the use of paper ballots. It will borrow a notion from the paper-based protocols—that of a trusted computational device that does not encrypt votes, but does perform cryptographic checks on behalf of the voter (such as those to check that an audited ballot is correctly constructed). This device is not a voting system device, but is a voter device. Thus, in this model, the voter casts a vote on a voting machine in the polling place; additionally, he has access to a personal computational device while casting his vote. (Such a device could be, for example, a smart phone.) The purpose of this device is to audit the voting machine, to ensure it encrypts the vote correctly. As the personal device performs the audit, it is expected to be independent of the voting system. The protocol should be designed so that a coercive adversary, who controls the computational device of a voter, but does not collude with the voting system, is not able to determine whether the voter voted in a certain manner.

We describe two protocols, each with similar properties. We see that the protocols are not incoercible if the computing device performs a denial of service attack, by either refusing to carry out the protocol steps, or carrying them out incorrectly. On the other hand, if the computing device can be trusted to carry out the steps of the protocol, it appears that we do have two protocols with the properties of verifiability and incoercibility. This is work in progress; we do not provide precise protocol definitions or security proofs of any kind, though we do provide some security arguments.

2 Related Work

Recent approaches to the voting problem have focused on paper-based protocols, which either use paper ballots, or paper receipts, or both. The protocols rely on the write-once property of paper, and enable voters to encrypt their votes without access to a trusted computational device [4, 9, 6, 3, 10, 14, 2, 5]. Some of these protocols use cleverly-designed paper ballots with the property that a voter can encrypt her vote simply by filling the ballot. Other protocols rely on encryption machines which print out the encrypted vote on a piece of paper. In either case, voters take the encrypted vote home; this forms the voter receipt. To check if the encryption is correct, receipts may be audited at random; votes corresponding to such receipts are not cast. Thus, using the

newer protocols, the voter need not have access to trusted software in the polling booth. She does need access to software to check the correctness of audited receipts, however, this software need not be provided by the voting system manufacturers or election officials; the voter may obtain it from any—and, in fact, several—sources.

While computing the tally, the voting system makes available, to the public, a sequence of cryptographic commitments to the computations performed to determine the tally. After the election outcome is announced, the tally computation is audited: the voting system is challenged by election auditors to open some of the commitments, chosen at random. Anyone can write software to check the opening of the commitments. The commitments are designed so that, if even a few votes are tallied incorrectly, at least one commitment will not open correctly (with very high probability).

Helios is a recent example of a voting system that does not use paper and has similar properties to the protocols we present in this paper.

3 eTegrity

We first describe an electronic version of the Scantegrity protocol; this version replaces the Scantegrity paper-based front-end with an electronic front-end. Note that the electronic front-end is far easier to integrate with various user interfaces, and hence results in a voting system that is far more accessible to voters who have difficulties with marking or seeing/reading ballots. We use the model proposed earlier: the voter votes on a voting machine in a polling booth. He also has access to a personal computational device that does not collude with the voting system to change the vote; however, this device may be examined by a coercer.

3.1 Simplified Protocol Sketch

The participants in the vote encryption protocol consist of (a) a voting computer that we call a **DRE** (b) a personal computational device that communicates with the **DRE**—we term this device the **Assistant**, and (c) the **Voter**.

Electronic Ballot Preparation

DRE will commit to several standard Scantegrity ballots, consisting of Scantegrity serial number(s) and several races; each race consisting of a set of choices, each with a corresponding Scantegrity confirmation code. For example, “Alice: X8T; Bob: MWQ; Carol: B7K”. Several such committed ballots will be displayed on a secure public bulletin board.

Electronic Ballot Presentation

Voter enters a polling booth to vote in private. He has with him his computational **Assistant**. **Assistant** requests and obtains a committed ballot, and checks that it is one of those on the secure bulletin board. **Assistant** generates a pseudo-random bit, and sends to **DRE** a commitment to it. This bit is also communicated to the voter. This bit determines whether **Voter** will audit or cast this ballot; alternately, the choice may be left to **Voter**. Further still, **Voter** may not commit to this choice. Whether it is a **Voter** choice, and whether it is committed to, will determine incoercibility vs. usability trade-offs.

Vote Casting

DRE presents the electronic ballot to **Voter**, who responds with his selections on the races. **Voter** also communicates to **Assistant** a valid selection, not necessarily the one made by **Voter**. **DRE** sends the corresponding confirmation numbers to **Assistant**, which confirms receipt to **Voter**. **Assistant**

then communicates to **DRE** whether **Voter** wishes to cast or audit the ballot (this may be a choice made by **Voter** at this instant, or might be a pseudo-random number generated by **Assistant** earlier; further, **Assistant** might have committed to it). If the vote is cast, **Assistant** and **DRE** sign the cast confirmation code, and **Assistant** checks its presence on the secure bulletin board. **Voter** then leaves the booth, and need not perform any further procedural steps with respect to ballot casting.

Ballot Audit

If the ballot is to be audited, **DRE** opens the committed ballot to **Assistant**, who checks that the opened ballot is correct (all confirmation numbers are not identical) and that the confirmation number obtained earlier from **DRE** is one of those on the ballot. Finally, **Assistant** communicates to **Voter** the selection corresponding to his confirmation code if it is not that which the **Voter** provided after making his selection. **Voter** hence knows if **DRE** cheated on this ballot.

Tally verifiability may be obtained by using the verifiable tally-computation component of Scantegrity.

3.2 Protocol Details

The above protocol may be extended so that **Voter** may request many ballots at a time. Assume we have a contest with c candidates and the voter can choose one of them.

The protocol is as follows:

1. **Voter** determines a security parameter n , representing the number of ballots in a session.
2. **Voter** send the number n to **DRE**, via **Assistant**.
3. For all the n ballots, **DRE** randomly assigns confirmation codes to candidates on each ballot (on ballot 4711, the confirmation code for Alice is B7K, for Bob X8T, for Carol MWQ). On different ballots, the same candidate may have different confirmation codes.
4. **DRE** sends to **Assistant** commitments to all the assignments for all ballots.
5. **Assistant** computes n random bits, such that at least one of the bits is a 1. A zero bit represents a ballot that is going to be audited and a 1 bit represents a ballot that might be cast.
6. **Assistant** sends **DRE** a commitment to the n bits (i.e. a commitment to which ballots are going to be audited).
7. **Assistant** tells **Voter** which bits are 1 (which of the n ballots cast be used for casting a vote)
8. **Voter** sends **DRE** (via a channel that is not tappable by **Assistant**) exactly n votes (in clear text, e.g. on the first ballot a vote for Carol, on the second ballot a vote for Alice, etc), such that at least one of the ballots corresponding to a bit with value 1 contains the candidate she wants to vote for. This is set1 of votes
9. **Voter** sends **Assistant** n votes. This is set2 of votes. Set1 and set2 may have the same votes on none, some, or all positions.
10. **DRE** sends **Assistant** the confirmation codes for all the votes it got from the voter in step 8

11. **Assistant** reveals to **DRE** the ballots that are used for auditing (the n bits generated in step 5).
12. **DRE** verifies the commitment to the n bits.
13. For each audited ballot (each bit with a value of 1 in the n bits), **DRE** sends **Assistant** all the confirmation codes corresponding to the votes received from **Voter** in step 8 and opens the commitments to them.
14. **Assistant** checks that confirmation codes on all audited ballot match the commitments.
15. **Assistant** checks that the confirmation codes received by **Assistant** in step 10 correspond to the votes **Assistant** got from **Voter** in step 9. For each vote that does not check, **Assistant** informs **Voter**
16. If **Assistant** did not report any error in the previous step, or if the errors reported are consistent with what **Voter** sent to the **DRE** in step 8, **Voter** chooses a ballot to cast (one of the ballots that were not audited) and sends **Assistant** instructions to cast it. Otherwise, **Voter** aborts the protocol and notifies the election authorities.
17. **Assistant** signs the confirmation code (received in step 10) on the ballot indicated by **Voter** in the previous step and sends the signed confirmation code to **DRE**.
18. **DRE** signs the confirmation code and publishes it on the bulletin board.
19. **Assistant** checks the correct posting on the bulletin board and repeats step 17.

3.3 Brief security analysis

1. The public bulletin board contains data that does not allow the public to find out how **Voter** voted. Indeed, from the confirmation code and the audited ballot, it is not possible to infer the candidate selected.
2. **Assistant** does not learn the vote cast by **Voter**. Indeed, in step 8 **Voter** sends all the votes to **DRE** using a channel that **Assistant** cannot read. It follows that, even if a coercer forces a voter to use a specific **Assistant**, the coercer cannot find out how the voter voted.
3. During one execution of the protocol, **DRE** can cheat one voter with a probability of $\frac{1}{n}$. Indeed, **Voter** can choose $n-1$ ballots to audit and one ballot to cast. **DRE** does not know which of the n ballots is going to be used for casting to the chance of correctly guessing it is one out of n . It is important the **Voter** does the checks in step 16 and aborts the protocol if any inconsistencies are detected.
4. **DRE** cannot ignore **Voter**'s input. Justification: **Assistant** sends **DRE** a signed encoded vote and all the receipts posted on the public bulletin board must also be signed by **Assistant**.

Undesirable properties:

1. **DRE** can ignore **Voter**'s input in step 8 and choose the candidate for some or all of **Voter**'s votes. **Voter** will detect this in step 16, and can then abort the protocol, but does not have proof of cheating.

2. **Voter** has to communicate to **DRE** her choices on many ballots, not only the candidate she wants to vote for. **Voter** also has to remember all these choices on all the ballots for the checking in step 16. Small values of n reduce the probability with which a cheating **DRE** will be detected by **Voter**.
3. **Assistant** needs to be able to digitally sign messages.
4. If the coercive adversary can program **Assistant**, this protocol is not incoercible. For example, **Assistant** can refuse to tell **Voter** which votes will be audited. Hence, either **Voter** casts all his votes for the coercer's candidate, or casts some or all votes for his own candidate, risking detection by the coercer, as well as risking, in general, that the correct vote will not be communicated to **DRE**. For another example, **Assistant** can communicate incorrectly to **Voter** which votes will be audited, thus determining how he intended to vote. On the other hand, if **Assistant** were not allowed to determine which votes would be audited or cast, this risk is mitigated.

4 EPunchScan

We now present an electronic protocol that combines the front-end of PunchScan with the back-end of Scantegrity. A key aspect of this protocol is its use of oblivious transfer.

4.1 Protocol Sketch

The participants in the vote encryption protocol consist of **DRE**, **Assistant** and **Voter**.

Electronic Ballot Preparation

The **DRE** will commit to several two-part ballots. The first part of each ballot will consist of a permutation of candidate ordering. This will be presented as a mapping between candidates and a number, such as: “for Alice press 2, for Bob press 3, for Carol press 1”. The second part will be a map between the numbers of the first ballot part and Scantegrity confirmation numbers, such as: “for 1 the confirmation code is B7K, for 2 X8T, for 3 MWQ”, and so on. Several such committed ballots will be displayed on a secure public bulletin board.

Electronic Ballot Presentation

The **Voter** enters a polling booth to vote in private. He has with him his computational **Assistant**. The **Assistant** requests and obtains the committed ballot, and checks that it is one of those on the secure bulletin board. **Assistant** and **DRE** perform an oblivious transfer protocol so that **Assistant** obtains one of the two ballot parts, and **DRE** does not know which one it is. **Assistant** checks that the ballot part corresponds to its commitment.

Vote Casting

The **DRE** presents the permuted voter list on the first half of the electronic ballot to **Voter**, the **Voter** makes a choice and communicates the number corresponding to the choice. **DRE** sends the confirmation number on the second half of the electronic ballot to **Assistant**. If, in the oblivious transfer, **Assistant** obtained the first ballot half, it presents this to **Voter** at the same time as does **DRE**. So, for example, **DRE** might present the ballot half in one ear and **Assistant** in the other. If these are not identical, **Voter** will notice it and know that **DRE** is cheating. If, in the oblivious transfer, **Assistant** obtained the second ballot half, it checks the choice corresponding

to the confirmation number. It then presents the choice to **Voter** at the same time as does **DRE**. If these are not identical, **Voter** will notice it and know that **DRE** is cheating.

Tally verifiability may be obtained by using the verifiable tally-computation component of Scantegrity.

4.2 Protocol Detail

Assume we have a contest with c candidates and **Voter** can choose one of them.

For a specific ballot, here are the steps of the protocol

1. **DRE** randomly assigns the numbers $\{1,2,3,\dots,c\}$ to candidates (e.g. for Alice press 2, for Bob press 3, for Carol press 1). This is file1.
2. **DRE** assigns random confirmation codes to the numbers (for 1 the confirmation code is B7K, for 2 X8T, for 3 MWQ). This is file2.
3. **DRE** sends to **Assistant** commitments to file1 and file2.
4. **DRE** generates two keys $K1$ and $K2$ and encrypts file1 with $K1$ and file2 with $K2$.
5. **DRE** sends **Assistant** the encrypted versions of file1 and file2
6. Using an oblivious transfer protocol, **Assistant** asks **DRE** to reveal either $K1$ or $K2$. **DRE** does not know which key **Assistant** got.
7. **Assistant** checks that the result of the decryption matches one of the corresponding commitment. Otherwise **Assistant** has proof of malfeasance.
8. Synchronized, both **DRE** and **Assistant** play file1 to **Voter** at the same time, using separate channels. If **Assistant** got $K1$, it plays file1, otherwise it is silent in this step (e.g. **Voter** hears "for Alice press 2, for Bob press 3, for Carol press 1" in either one or both ears).
9. **Voter** checks that both **DRE** and **Assistant** played the same content in the previous step. Otherwise it aborts the protocol and notices the election officials.
10. **Voter** communicates to **DRE** the number assigned to her favorite candidate.
11. **DRE** encrypts the number with $K2$ and sends the encryption to **Assistant**.
12. **DRE** sends to **Assistant** the confirmation code corresponding to the number (in clear text). This is the receipt for the vote.
13. If **Assistant** got $K2$ in the oblivious transfer, **Assistant** decrypts the number using $K2$ and checks that file2 contains the number attached to the confirmation code. Otherwise, **Assistant** has proof of malfeasance.
14. Synchronized, both **DRE** and **Assistant** play the number to **Voter** at the same time using separate channels. If **Assistant** got $K2$, it plays the number, otherwise it is silent in this step (e.g. **Voter** hears "You entered 3" in either one or both ears)
15. **Voter** checks that both **DRE** and **Assistant** played the same content in the previous step and that it is the number that she indeed entered. Otherwise it aborts the protocol and notices the election officials.

16. Both **DRE** and **Assistant** play the confirmation code (e.g. **Voter** hears "The confirmation code is MWQ" in both ears).
17. **DRE** publishes the confirmation code on the public bulletin board.

The following steps can be added to mitigate improper influence attacks.

1. **Assistant** proves to **DRE** which key it got during the oblivious transfer protocol
2. If **Assistant** got K_1 , **DRE** plays to **Voter** "You can lie about which number you've entered" (**Assistant** is not able to decrypt the number **Voter** entered)
3. if **Assistant** got K_2 , **DRE** plays to **Voter** "You can lie about which candidate is associated with number 3" (**Assistant** is not able to decrypt the association between candidates and numbers)

As a concrete example, **DRE** is a smart phone programmed by the election authority and **Assistant** is a smart phone that a blind voter has. The two phones communicate via bluetooth. In step 5, **Voter** hears in one ear what **DRE** is playing and in the other ear what **Assistant** is playing. **Voter** will notice if she hears different things at the same time in the two ears.

4.3 Brief security analysis

Desirable security properties:

1. The public bulletin board contains data that does not allow the public to find out how **Voter** voted. Indeed, the bulletin board contains the confirmation code and either file1 or file2 (but never both), and the vote cannot be inferred from the two.
2. **Assistant** does not learn the vote cast by **Voter**. Indeed, **Assistant** only knows information that is going to be published on the bulletin board. It is important that **Assistant** does not intercept the communication between **DRE** and **Voter** in step 5. It follows that, even if a coercer forces a voter to use a specific **Assistant**, the coercer cannot find out how the voter voted.
3. During one execution of the protocol, **DRE** can cheat one voter with a probability of 50%. Justification: During the oblivious transfer, **DRE** does not know which part of the ballot **Assistant** gets; **DRE** has a 50% chance of correctly guessing the part that **Assistant** got and, in step 5 present the voter a modified version of the other part. It is important **Voter** notices if **DRE** and **Assistant** present her the contradicting information. An immediate consequence of this property is that **DRE** can cheat on b ballots without any of the b **Voters** noticing with a probability of $\frac{1}{2^b}$.
4. **Voter** does not need to follow any indirection for voting (she does not need to remember any information). **Voter** types in the number associated with her favorite candidate and remembers it for checking in step 15

Undesired properties:

1. Suppose that, in either step 9 or 15, **Voter** notices that **Assistant** and **DRE** present her with conflicting information. All that **Voter** can do is to abort the protocol. Even though she caught **DRE** cheating, she does not have any proof that cheating occurred. This is because **Assistant** is not allowed to listen to the communication between **DRE** and **Voter**. If **Voter** aborts and restarts the protocol from step 1, now **DRE** has another chance of cheating in 5. This is similar to a vulnerability of eTegrity.
2. If the coercive adversary can program **Assistant**, this protocol is not incoercible. For example, **Assistant** can be programmed to refuse to prove to **DRE** which key it obtained. Hence, either **Voter** casts his vote for the coercer's candidate, or has a probability of $\frac{1}{2}$ of being detected by the coercer, when queried about his choices.

5 The Use of Blind Signatures

Blind signatures may be used to enable the voter to prove that a cheating **DRE** has cheated with ePunchScan.

5.1 Detailed Protocol

1. **DRE** creates a ballot with two parts. Part1 is of the form "For Alice remember 2, for Bob remember 3, for Carol remember 1.". Part2 is of the form "to vote for 1 press Y, to vote for 2 press X, to vote for 3 press Z".
2. **DRE** publishes on the public bulletin board signed commitments to part1 and part 2.
3. Using an oblivious transfer protocol, **Assistant** gets one of the two parts of the ballot.
4. For the part received, **Assistant** checks the signature and the compliance with the commitment.
5. Synchronized, both **DRE** and **Assistant** present the ballot to **Voter** at the same time. **DRE** presents both parts to **Voter** and **Assistant** presents the part it got during the oblivious transfer. It is important that **DRE** presents both parts to **Voter** such that **Assistant** is not able to listen.
6. If **Voter** notices that **Assistant** presents its part differently than **DRE** presents the same part, she aborts the protocol.
7. **Voter** enters a letter (either X, Y or Z) onto her **Assistant**. This letter corresponds to her vote (e.g. to vote for Bob, **Voter** enters Z)
8. **Assistant** signs the choice of **Voter** (letter Z) and sends it to **DRE**
9. **DRE** signs the received choice of **Voter** and posts this signed choice on a public bulletin board (this posted choice is signed by both **Assistant** and **DRE**). This is the receipt **Voter** gets.

If **Assistant** does not have the capabilities to produce a digital signature, **Assistant** can send the choice using a blind signature protocol instead of step 8:

1. **Assistant** blinds the choice of **Voter**.

2. **Assistant** sends the blinded choice to **DRE**.
3. **DRE** signs the blinded choice (**DRE** does not know the clear text vote, so it cannot selectively refuse to sign).
4. **DRE** sends the signed blinded choice to **Assistant**.
5. **Assistant** unblinds the choice.
6. **Assistant** sends choice of **Voter** signed by **DRE** to **DRE** and keeps a copy as a receipt.

5.2 Brief security analysis

1. The public bulletin board contains data that does not allow the public to find out how **Voter** voted. Indeed, from the encoded vote (Z) and one of the two parts of the ballot, it is not possible to infer the candidate selected.
2. **Assistant** does not learn the vote cast by **Voter**. Indeed, **Assistant** only knows information that is going to be published on the bulletin board. It is important that **Assistant** does not intercept the communication between **DRE** and **Voter** in step 5. It follows that, even if a coercer forces a voter to use a specific **Assistant**, the coercer cannot find out how the voter voted.
3. During one execution of the protocol, **DRE** can cheat one voter with a probability of 50%. Justification: During the oblivious transfer, **DRE** does not know which part of the ballot **Assistant** gets; **DRE** has a 50% chance of correctly guessing the part that **Assistant** got and, in step 5 present the voter a modified version of the other part. It is important the **Voter** notices if **DRE** and **Assistant** present her the contradicting information. An immediate consequence of this property is that **DRE** can cheat on b ballots without any of the b voters noticing with a probability of $\frac{1}{2^b}$.
4. **DRE** cannot ignore **Voter**'s input. Justification: **DRE** gets from **Assistant** a receipt that is already signed by either **Assistant** (if capable of signing digitally) or by **DRE** itself (if the blind signature protocol is used). All the receipts posted on the public bulletin board must be also signed by.

Undesired properties:

1. Assuming that in step 5 **Voter** notices that **Assistant** and **DRE** present her with conflicting information. All that **Voter** can do is to abort the protocol. Even though she caught **DRE** cheating, she does not have any proof that cheating occurred. This is because **Assistant** is not allowed to listen to the communication between **DRE** and **Voter**. Assuming **Voter** aborts and restarts the protocol from step 1, now **DRE** has another chance of cheating in 5.
2. The ballot uses a double indirection. **Voter** needs to remember the number associated with her favorite candidate and then type in the letter associated with that number (same as in PunchScan). This requires voter effort.

6 Conclusions

We have presented electronic versions of the Scantegrity and PunchScan protocols, based on a new model of the voting process. In our model, the voter votes in a polling booth, and has access to a computational `Assistant` which can be examined by the coercer after the voter casts his vote. It appears that the protocols might be incoercible and tally-verifiable if the adversary cannot program the `Assistant` and does not collude with the voting system. This is work in progress, and we do not present rigorous definitions, descriptions, or proofs.

References

- [1] Ben Adida. Helios: Web-based Open-Audit Voting. In *Proceedings of the Seventeenth Usenix Security Symposium (USENIX Security 2008)*, June 2008.
- [2] 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 the Electronic Society*, pages 29–40, New York, NY, USA, 2006. ACM Press.
- [3] Josh Benaloh. Simple verifiable elections. In *EVT'06: Proceedings of the USENIX/Accurate Electronic Voting Technology Workshop 2006 on Electronic Voting Technology Workshop*, Berkeley, CA, USA, 2006. USENIX Association.
- [4] David Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security and Privacy*, pages 38–47, January/February 2004.
- [5] David Chaum, Richard Carback, Jeremy Clark, Aleksander Essex, Stefan Popoveniuc, Ronald L. Rivest, Peter Y. A. Ryan, Emily Shen, and Alan T. Sherman. Scantegrity II: End-to-end verifiability for optical scan election systems using invisible ink confirmation codes. In *EVT'07: Proceedings of the USENIX/Accurate Electronic Voting Technology on USENIX/Accurate Electronic Voting Technology Workshop*. USENIX Association, 2008.
- [6] David Chaum, Peter Y. A. Ryan, and Steve Schneider. A practical voter-verifiable election scheme. In *In Sabrina De Capitani di Vimercati, Paul F. Syverson, and Dieter Gollmann, editors, ESORICS, volume 3679 of Lecture Notes in Computer Science*, pages 118–139. Springer, 2005.
- [7] David L. Chaum. Untraceable electronic mail, return address, and digital pseudonym. *Communication of ACM*, February 1981.
- [8] Jeremy Clark Aleksander Essex Stefan Popoveniuc Ronald L. Rivest Peter Y. A. Ryan Emily Shen Alan T. Sherman David Chaum, Richard Carback and Poorvi L. Vora. Scantegrity: End-to-end verifiability for optical scan elections. *IEEE Transactions On Information Forensics And Security: Special Issue On Electronic Voting*. In Review.
- [9] C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *8th ACM Conference on Computer and Communications Security*, pages 116–125, 2001.
- [10] Stefan Popoveniuc and Ben Hosp. An introduction to PunchScan. In *IAVoSS Workshop On Trustworthy Elections (WOTE 2006)*, Robinson College, Cambridge UK, June 2006.
- [11] Punchscan. <http://www.punchscan.org>.
- [12] Ronald L. Rivest. On the Notion of “Software Independence” in Voting Systems. *Phil. Trans. Royal Society A*, 366:3759–3767, October 2008.
- [13] Ronald L. Rivest and John P. Wack. On the Notion of “Software Independence” in Voting Systems. <http://vote.nist.gov/SI-in-voting.pdf>, July 2006.
- [14] P. Y. A. Ryan and S. A. Schneider. Prêt à voter with re-encryption mixes. In *In D. Gollmann, D., J. Meier, and A. Sabelfeld, editors, ESORICS, volume 4189 of Lecture Notes in Computer Science*, pages 313–326. Springer-Verlag, 2006.